கணிப்பொறி இயல்

மேல்நிலை – இரண்டாம் ஆண்டு

பொருள்நோக்குத் தொழில்நுட்பம்

தொகுதி II

தமிழ்நாட்டுப்
பாடநூல் கழகம்

கணிப்பொறி இயல்

மேல்நிலை – இரண்டாம் ஆண்டு

தொகுதி 2 : பொருள்நோக்குத் தொழில்நுட்பம்

தமிழ்நாடு அரசு

தீண்டாமை ஒரு பாவச்செயல்
தீண்டாமை ஒரு பெருங்குற்றம்
தீண்டாமை மனிதத் தன்மையற்ற செயல்

தமிழ்நாட்டுப் பாடநூல் கழகம்
கல்லூரிச்சாலை, சென்னை - 600 006.

© தமிழ்நாடு அரசு
முதல்பதிப்பு 2005

தலைவர், பாடத்திட்டக் குழு
முனைவர் பாலகுருசாமி இ, முன்னாள் துணைவேந்தர்,
அண்ணா பல்கலைக் கழகம், சென்னை.

பாடப் புத்தக ஒருங்கிணைப்பாளர்
முனைவர் சங்கர நாராயணன் வி, இயக்குனர்,
தமிழ் இணையப் பல்கலைக் கழகம், சென்னை.

ஆங்கில மூல நூலாசிரியர்
முனைவர் கோபால் டி வி, அண்ணா பல்கலைக் கழகம், சென்னை.
திருமதி ருக்குமணி கே, பத்ம சேஷாத்திரி பாலபவன், சென்னை.
முனைவர் சங்கர நாராயணன் வி, இயக்குனர்,
தமிழ் இணையப் பல்கலைக் கழகம்,
சென்னை.

மீள்பார்வையாளர்
முனைவர் கோபால் டி வி, அண்ணா பல்கலைக் கழகம், சென்னை.
திருமதி வசந்தி கிருஷ்ணகுமார், சைஃபி லிமிடெட், சென்னை.

அமைப்புப் பதிப்பாளர்
திருமதி சுபா ரவி, இயக்குநர், டிஜிடாடி கன்சல்டன்ஸி பிரைவேட் லிமிடெட்,
சென்னை.

நூலாசிரியர்
சிவலிங்கம் மு, பொறியாளர், பிஎஸ்என்எல், சென்னை.

விலை ரூ:

பாடங்கள் தயாரிப்பு: தமிழ்நாடு அரசுக்காகப்
பள்ளிக் கல்வி இயக்ககம், தமிழ்நாடு

இந்நூல் 70 ஜி.எஸ்.எம். தாளில் அச்சிடப்பட்டுள்ளது.

முன்னுரை

மனிதனின் அன்றாட நடைமுறை வாழ்வில் தீர்க்கப்பட வேண்டிய பிரச்சினைகளை இனிக் கணிப்பொறியின் உதவியின்றித் தீர்க்க முடியாது என்ற நிலைமைக்குத் தள்ளப்பட்டுள்ளோம். நான் கணிப்பொறிகளைக் கண்டு அஞ்சவில்லை. அவை இல்லாவிடில் என்ன ஆகும் என்றே அஞ்சுகிறேன்.

–ஐசக் அசிமோவ்

கணிப்பொறிகள், சிக்கலான அறிவியல், வணிக, நிர்வாகப் பிரச்சினைகளைத் தீர்ப்பதில் நமக்கு உதவுகிற எந்திரங்கள் ஆகும். ஏராளமான தொழிலக, வணிக அமைப்புகளின் தானியங்கமாக்கலுக்கு அவை உதவியுள்ளன. எனினும், அவை மனிதர்களால் உருவாக்கப்பட்டு, மேலாண்மை செய்யப்படும் எந்திரங்கள் என்பதை நாம் நினைவில் கொள்ள வேண்டும்.

அவை தானே சிந்திக்கும் மூளையைப் பெற்றிருக்கவில்லை. அவை செய்கின்ற பணி எதுவாயினும், மனிதரின் ஆணைகளின்படியே செயல்படுகின்றன. நாம் தரும் ஆணைகளை, இருக்கின்ற மென்பொருள்களின் உதவிகொண்டு, நிறைவேற்ற முடியும் என்கிற நிலை இருக்கும்வரை, அவை மிகப் பணிவாக நமது ஆணைகளை நிறைவேற்றுகின்றன. அவை சரியானவையா, பிழையானவையா என்றுகூடப் பார்ப்பதில்லை. அதாவது, கணிப்பொறிகளுக்குத் தன்னறிவு கிடையாது.

எதைச் செய்ய வேண்டும், எப்படிச் செய்ய வேண்டும், எப்போது செய்ய வேண்டும் என்பதைக் கணிப்பொறிகளுக்குத் தெளிவாக ஆணையிட வேண்டும். கணிப்பொறிகளுக்கு அதுபோன்ற ஆணைகளைத் தருவதற்கான ஒரு வழிமுறையே ***நிரலாக்கம்*** எனப்படுகிறது. அத்தகைய ஆணைகளை உருவாக்கவும், பரிமாறிக்கொள்ளவும் பயன்படுத்தப்படும் மொழியே ***நிரலாக்க மொழி*** என அறியப்படுகிறது.

இருநூற்றுக்கு மேற்பட்ட நிரலாக்க மொழிகள் தற்போது பயன்பாட்டில் உள்ளன. அவற்றுள் சில அறிவியல் பயன்பாட்டுக்கு எனவும், சில வணிகப் பயன்பாடுகளுக்கு எனவும் வடிமைக்கப்பட்டுள்ளன. வேறு சில, பொதுப்பயன் மொழிகள் ஆகும். ஒரு நிரலாக்க மொழி, நிரலர்கள், தீர்வுப் படிநிலைகளை எளிதாக உருவாக்குவதற்கும் வடிவமைப்பதற்கும் உரிய பண்புக்கூறுகளைக் கொண்டிருக்கும்.

நாம் ஏற்கெனவே சி-மொழியைக் கற்றுள்ளோம். அது, ***செயல்முறை நோக்கு*** மொழியாகும். பெயருக்கு ஏற்றுவாறு, தீர்வுக்கான செயல்முறைகளே முக்கியத்துவம் பெற்றன. சி-மொழி சக்திவாய்ந்த பொதுப்பயன் மொழியாகும். இந்நூல் தொகுதி சி-மொழியின் மேம்பட்ட வடிவமான சி++ மொழி பற்றி விளக்குகிறது. சி++, ***பொருள்நோக்கு நிரலாக்கம்*** (Object Oriented Programming

-OOP) என்னும் முற்றிலும் புதிய கருத்துருவைக் கொண்டுள்ளது. எனவே, இம்மொழியை ஒரு பொருள்நோக்கு நிரலாக்கத் தொழில்நுட்பம் என வகைப் படுத்துகின்றனர். நாம் சி++ மொழியைக் கற்பதற்குக் காரணம், இன்றைக்கு அதுவே தொழிலகத் தரப்பாட்டு ‘ஊப்’ (OOP) மொழியாக ஆகிவிட்டது.

சி++ போன்ற ஊப் மொழிகளில் ***பொருள்கள்*** (Object) என அறியப்படும் நடப்புலகின் உள்பொருள்களுக்கே முக்கியத்துவம் தரப்படுகிறது. இந்தப் பொருள்கள், நிரல்களில் எடுத்தாளப்பட வேண்டிய ஒரு நபர், ஒரு கார், ஒரு தரவு அட்டவணை அல்லது இது போன்ற எந்தப் பொருளாகவும் இருக்கலாம். மனிதர்களாகிய நாம், அன்றாட வாழ்க்கைப் பிரச்சினைகளை தனித்த பொருள் களின் ஒரு தொகுதியாகவே நோக்குகிறோம். அப்பொருள்களுக்கிடையே நிலவும் உறவுமுறைகளைக் கணக்கில் எடுத்துக்கொண்டு அப்பிரச்சினை களுக்குத் தீர்வுகாண முயல்கிறோம். அதுபோலவே சி++ மொழியில், நிரலாக்கச் சிக்கல்கள், பொருள்கள் மற்றும் அவற்றுக்கிடையேயான தகவல் பரிமாற்றம் ஆகியவற்றின் அடிப்படையிலேயே பகுத்தாயப்படுகின்றன.

கற்றுணர்வதற்கு எளிய, ஏராளமான எடுத்துக்காட்டுகளும் விளக்க நிரல் களும் இந்நூல் தொகுதியில் வழங்கப்பட்டுள்ளன. சி++ மற்றும் பொருள் நோக்கு நிரலாக்க உலகில் மிக விரைவில் அனுபவம் பெற இந்நூலின் பாடப் பகுதிகள் மாணவர்களுக்கு உதவும் என்பதில் ஐயமில்லை.

தகவல் தொழில்நுட்பம் சார்ந்த பயன்பாடுகள் பலவற்றையும் இந்நூல் தொகுதி விளக்குகிறது. அவற்றைப் புரிந்துகொள்வதற்கு உதவியாக அசைவூட்ட உள்ளடக்க வடிவில் படங்களும் வழங்கப்பட்டுள்ளன (தனியாக ஒரு குறுவட்டில்). அனைத்துப் பயன்பாடுகளிலும் தகவல் தொழில்நுட்பத்தை நன்னெறியுடன் பயன்படுத்துவது பற்றியும் விளக்கப்பட்டுள்ளது.

நூலாசிரியர்கள், மீள்பார்வையாளர்கள், பள்ளிக் கல்வி இயக்கக அலுவலர்கள் அனைவர்க்கும் அவர்களின் மிகச்சிறந்த பணிக்காகவும் ஒத்துழைப்புக்காகவும் என்னுடைய மெய்யான பாராட்டையும் நன்றியையும் தெரிவிக்கக் கடமைப்பட்டுள்ளேன்.

இ.பாலகுருசாமி
தலைவர்
பாடத்திட்ட மீள்பார்வைக் குழு

பொருளடக்கம்

பாடம் 1	**சி++ மொழியின்வழி பொருள்நோக்கு நிரலாக்கக் கருத்துருக்கள்** (Object Oriented Concepts using C++)	**1**
	1.1 பொருள்நோக்கு நிரலாக்க மொழி (Object Oriented Programming Language)	1
	1.2 பல்லுருவாக்கம் (Polymorphism)	4
	1.3 மரபுரிமம் (Inheritance)	5
	1.4 ஒரு நடைமுறை எடுத்துக்காட்டு: வீட்டு நீர்ப்பயன்பாடு	6
	பயிற்சி வினாக்கள்	9
பாடம் 2	**சி++ மொழியின் முன்னோட்டம்** (Overview of C++)	**10**
	2.1 முன்னுரை	10
	2.2 சி++ குறியுருத் தொகுதி (C++ Character Set)	10
	2.3 தரவு இனங்கள் (Data Types)	30
	2.4 மாறிகள் (Variables)	41
	பயிற்சி வினாக்கள்	51
பாடம் 3	**அடிப்படைக் கூற்றுகள்** (Basic Statements)	**54**
	3.1 உள்ளீட்டு / வெளியீட்டுக் கூற்றுகள்	54
	3.2 முதல் சி++ நிரல் – சி++ நிரலின் கட்டமைப்பு	56
	3.3 அறிவிப்புக் கூற்றுகள் (Declaration Statements)	57
	3.4 மதிப்பிருத்து கூற்றுகள் (Assignment Statements)	58
	3.5 கட்டுப்பாட்டுக் கட்டளை அமைப்புகள் (Control Structures)	59
	3.6 நிரல் உருவாக்கம்	88
	பயிற்சி வினாக்கள்	89
பாடம் 4	**செயற்கூறுகள்** (Functions)	**94**
	4.1 முன்னுரை	94
	4.2 செயல்கூறின் முன்வடிவு (Function Prototype)	96
	4.3 ஒரு செயற்கூறினை அழைத்தல்	98
	4.4 செயற்கூறினுக்கு அளபுருக்களை அனுப்பிவைத்தல்	99
	4.5 மதிப்பினைத் திருப்பியனுப்புதல்	109

4.6 inline செயற்கூறுகள் 112
4.7 மாறிகளின் வரையெல்லை (scope) விதிமுறைகள் 114
பயிற்சி வினாக்கள் 118

பாடம் 5 **கட்டமைப்புத் தரவினம் – அணிகள்** **124**
(Structured Data Type - Arrays)

5.1. முன்னுரை 124
5.2. ஒருபரிமாண அணி (Single Dimensional Array) 126
5.3 சரங்கள் (Strings) 131
5.4. இருபரிமாண அணி (Two-Dimensional Array) 136
5.5. சரங்களின் அணி 143
பயிற்சி வினாக்கள் 145

பாடம் 6 **இனக்குழுக்களும் பொருள்களும்** **151**
(Classes and Objects)

6.1 இனக்குழுக்கள் – ஒரு முன்னுரை 151
6.2 இனக்குழுவை வரையறுத்தல் 151
6.3 தரவு அருவமாக்கம் (Data Abstraction) 154
6.4 தரவு உறுப்புகளும் உறுப்புச் செயற்கூறுகளும் 154
6.5 பொருள்களை உருவாக்குதல் 155
6.6 இனக்குழு உறுப்புகளை அணுகுதல் 156
6.7 இனக்குழுவின் செயற்கூறுகளை வரையறுத்தல் 157
6.8 பொருள்களுக்கு நினைவக ஒதுக்கீடு 159
6.9 static தரவு உறுப்புகள் 160
6.10 பொருள்களின் அணிகள் 164
பயிற்சி வினாக்கள் 164

பாடம் 7 **பல்லுருவாக்கம்** **169**
(Polymorphism)

7.1 முன்னுரை 169
7.2 செயற்கூறு பணிமிகுப்பு (Function Overloading) 169
7.3 செயற்குறிப் பணிமிகுப்பு (Operator Overloading) 174
பயிற்சி வினாக்கள் 182

பாடம் 8 **ஆக்கிகளும் அழிப்பிகளும்** **184**
(Constructors and Destructors)

8.1 முன்னுரை 184
8.2 ஆக்கிகள் (Constructors) 184
8.3 ஆக்கியின் செயல்பாடுகள் 185
8.4 ஆக்கியின் பணிமிகுப்பு (Constructor Overloading) 185
8.5 ஆக்கி வரையறுப்பு மற்றும் பயன்படுத்தலுக்கான விதிமுறைகள் 190
8.6 அழிப்பிகள் (Destructors) 190
8.7 அழிப்பி வரையறுப்பு மற்றும் பயன்படுத்தலுக்கான விதிமுறைகள் 191
பயிற்சி வினாக்கள் 192

பாடம் 9 **மரபுரிமம்** **193**
(Inheritance)

9.1 முன்னுரை 193
9.2 மரபுரிமத்தின் பலன்கள் 194
9.3 தருவிக்கப்பட்ட இனக்குழுவும் அடிப்படை இனக்குழுவும் 194
9.4 காண்புநிலைப் பாங்கு / அணுகியல்பு வரையறுப்பி (Visibility Mode / Accessibility Specifier) 197
9.5 மரபுரிமத்தின் வகைகள் 201
பயிற்சி வினாக்கள் 204

பாடம் 10 **சமுதாயத்தின் மீது கணிப்பொறியின் தாக்கம்** **208**
(Impact of Computers on Society)

10.1 முன்னுரை 208
10.2 சொந்தப் பயன்பாட்டுக்குக் கணிப்பொறிகள் 209
10.3 கணிப்பொறி மயமாக்கப்பட்ட வீடுகள் 209
10.4 வீட்டிலிருந்தே வங்கிப் பணியும் கடைப்பொருள் வாங்கலும் 211
10.5 கல்வியில் கணிப்பொறிகள் 213
10.6 பொழுதுபோக்கில் கணிப்பொறிகள் 215
10.7 நலவாழ்வுக்குக் கணிப்பொறிகள் 216
10.8 வேளாண்மைத் துறையில் கணிப்பொறிகள் 217
10.9 நிகழ்நேரப் பயன்பாடுகளில் இணையம் 218
பயிற்சி வினாக்கள் 218

பாடம் 11 **தகவல் தொழில்நுட்பம் சார்ந்த சேவைகள்**
(IT Enabled Services) **219**

11.1 முன்னுரை 219
11.2 மின்–அரசாண்மை (e-Governance) 220
11.3 அழைப்புதவி மையங்கள் (Call-Centres) 222
11.4 தரவு மேலாண்மை (Data Management) 222
11.5 மருத்துவப் பெயர்ப்பாவணமும் தொலைமருத்துவமும் (Medical Transcription and Tele-medicine) 224
11.6 கணிப்பொறித் தரவாக்கம் (Data Digitization) 225
11.7 வலையகச் சேவைகள் 227
பயிற்சி வினாக்கள் 227

பாடம் 12 **கணிப்பொறி நன்னெறி**
(Computer Ethics) **228**

12.1 தரவுப் பாதுகாப்பு (Data Security) 229
12.2 கணிப்பொறிக் குற்றம் (Computer Crime) 229
12.3 அரண் உடைத்தல் (Cracking) 232
12.4 வேலை, குடும்பம், பொழுதுபோக்கு 232
பயிற்சி வினாக்கள் 232

பாடம் 1

சி++ மொழியின்வழி பொருள்நோக்கு நிரலாக்கக் கருத்துருக்கள்

(Object Oriented Concepts using C++)

1.1 பொருள்நோக்கு நிரலாக்க மொழி

(Object Oriented Programming Language)

பலதரப்பட்ட சிக்கல்களுக்குத் தீர்வுகாணும் ஒரு கருவியாகக் கணிப்பொறி விளங்குகிறது. சிக்கல்களுக்கான தீர்வுகள் கணிப்பொறி நிரல்களாகவும் (Programs) பயன்பாட்டு மென்பொருள்களாகவும் (Application Software) நமக்குக் கிடைக்கின்றன. இத்தகைய நிரல்களும் மென்பொருட்களும் ஏதேனும் ஒரு நிரலாக்க மொழியில் எழுதப்படுகின்றன.

ஒரு கணிப்பொறி நிரலானது, உள்ளீடாகத் தரப்படும் தரவுக் கூறுகளின் மீது செயல்படுகின்றது. இந்த உள்ளீட்டுத் தரவுகள் நமக்குத் தேவையான தகவல் கூறுகளாக மாற்றப்பட்டு, கணிப்பொறி நிரலின் வெளியீடாகப் பெறப்படுகின்றன.

தொடக்க கால நிரலாக்க மொழிகளில் உள்ளீட்டு வெளியீட்டுத் தரவுக் கூறுகள் ***மாறிகளாகக்*** (Variables) கையாளப்பட்டன. உள்ளீட்டுத் தரவுக் கூறுகள் ***தரவு இனங்களாக*** (Data Types) வகைப்படுத்தப்பட்டன. தரவுக் கூறுகளின் மீது நிகழ்த்தப்படவேண்டிய செயல்பாடுகளை கணிப்பொறிக்கு அறிவுறுத்த ***கட்டுப்பாட்டுக் கூற்றுகள்*** (Control Statements) பயன்பட்டன.

நிரலாக்க மொழிகள் வேறொரு வகையிலும் பயன்படுகின்றன. ஒரு சிக்கலின் தீர்வுபற்றிய நமது கருத்துகளை ஒருங்கமைக்க அவை உதவுகின்றன. தீர்க்கப்பட வேண்டிய சிக்கல்களும் உருவாக்கப்பட வேண்டிய பயன்பாட்டு மென்பொருள்களும் மிகவும் சிக்கலாகிப் போகும்போது, நிரலாக்க மொழிகளின் இந்தப் பண்புக்கூறு அதிக முக்கியத்துவம் பெறுகின்றது. இத்தகைய செயல்திறன் கொண்ட நிரலாக்க மொழிகள் பல உருவாக்கப்பட்டுள்ளன. இவை கணிப்பொறிக்கு அறிவுறுத்துவதற்கான எளிய வழிமுறைகளைக் கொண்டுள்ளன.

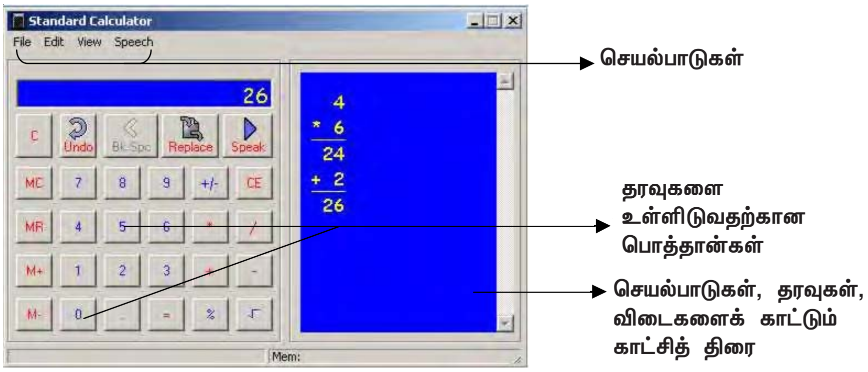
இருப்பினும், ஒரு சிக்கலுக்கான தீர்வினை, தரவு (data), செயல்பாடுகள் (operations) என இருகூறாகப் பிரித்து நோக்குவது, மனிதர்கள் நடைமுறை வாழ்க்கையின் சிக்கல்களுக்கு தீர்வுகாணும் முறையோடு ஒத்திருக்கவில்லை என்பது உணரப்பட்டது.

சி++ போன்ற பொருள்நோக்கு நிரலாக்க மொழிகள், நடைமுறை வாழ்க்கையின் சிக்கலான பிரச்சினைகளைத் தீர்க்க, பொதுவாக மனிதர்கள்

கையாளும் முறைகளை அடிப்படையாகக் கொண்டு வடிவமைக்கப் பட்டுள்ளன. பொதுவாக, மனிதர்கள் நடைமுறைப் பிரச்சினைகளைத் தீர்ப்பதற்கு முன், தீர்வுக்கு தேவைப்படும் பல்வேறு தனித்த பொருள்களை அடையாளம் காண்கின்றனர். அதன்பின் அப்பொருள்களுக்கு இடையேயான உறவுமுறைகளை உணர்ந்து கொள்கின்றனர். உறவுமுறை என்பது 'இது இதன் ஒரு பகுதி' அல்லது 'இது இந்த இனத்தைச் சார்ந்தது' என்பதைக் குறிப்பதாகும். 'ஒரு பொருளானது வேறொரு பெரிய பொருளின் ஒரு பகுதி' என்பதையும் 'ஒரு பொருளானது இன்னொரு பொருளின் இனத்தைச் சார்ந்தது' என்பதையும் புரிந்து கொள்கின்ற சாதாரணத் திறம்பாடுகளே நடைமுறை வாழ்க்கையின் சிக்கல்களுக்குத் தீர்வு காண்பதில் மிக இன்றியமையாத இடம் வகிக்கின்றன. தரவுகளையும் அத்தரவுகளின்மீது நிகழ்த்தப்படும் செயல் பாடுகளையும் ஒருங்கு சேர்த்து, சிக்கல்களுக்குத் தீர்வுகாணும் இந்த வழி முறையைப் ***பொருள்நோக்கு நிரலாக்கம்*** வழங்கியுள்ளது.

வேறு சொற்களில் கூறுவதெனில், தரவுக் கூறுகளின் தொகுதி, வேறு பிற செயற்கூறுகளை அழைக்காமலே, செயல்பாடுகளை நிகழ்த்துவதற்கேற்ற சிறு குழுக்களாகப் பிரிக்கப்படுகிறது. இத்தகைய தரவுக்குழு அதற்கான செயல்பாடுகளுடன் சேர்த்து ***இனப்பொருள்*** - சுருக்கமாக - ***பொருள்*** (object) எனப்படுகிறது. செயல்பாடுகள், பொருளின் பண்பியல்பை உணர்த்துகின்றன. நடப்புலகப் பொருளைக் கணிப்பொறி நிரலில் எடுத்தாள்வதற்கு இந்தப் ***பொருள்*** கருத்துரு உதவுகிறது.

எடுத்துக்காட்டாக, ஒரு கணிப்பியை (Calculator) நோக்குங்கள். அதற்கு ***நிலை*** (State), ***பண்பியல்பு*** (Behaviour) ஆகியவை உள்ளன. ***நிலை*** என்பது, நீள அகலம், பொத்தான்கள், காட்சித்திரை, செயற்குறிகள் போன்ற அதன் புறத்தோற்றத்தைக் குறிக்கின்றது. ***பண்பியல்பு*** என்பது, கூட்டல், கழித்தல், நினைவகத்தில் சேமித்தல், நினைவகத்தை அழித்தல் போன்ற அதன் செயல்பாடுகளை உணர்த்துகின்றது.



படம் 1.1 அடிப்படைக் கணிப்பி

பொருள்நோக்கு நிரலாக்க மொழியில் ஒரு ***கணிப்பி*** என்பது இவ்வாறு நோக்கப்படுகிறது:

பொருள் - கணிப்பி
தரவுகள்:
Number1,result, operator, Number_backup

செயல்கூறுகள்:
Additon()
Subtraction()
Erase_Memory()
Display_Result()

✓ ***பொருள்*** என்பது தொடர்புடைய செயற்கூறுகள், அச்செயற்கூறு களுக்கான தரவுகள் ஆகியவற்றைக் கொண்ட ஒரு குழுவாகும்.
✓ ***பொருள்*** என்பது, குறிப்பிட்ட செயற்பரப்புடன் கூடிய, தற்சார்பு கொண்ட ஒருவகைத் துணை நிரல் ஆகும்.

தரவுகளையும் அவை தொடர்பான செயற்கூறுகளையும் ***பொருள்கள்*** எனப்படும் அலகுகளாகக் குழுவாக்கும் செயலாக்கம் ***உறைபொதியாக்கம்*** (Encapsulation) என்னும் கருத்துருவுக்கு வழி வகுக்கிறது.

தரவுகளையும் செயற்கூறுகளையும் ஒரு ***பொருள் வரையறைக்குள்*** ஒன்றாகப் பிணைத்துவைக்கும் செயல்நுட்பம் ***உறைபொதியாக்கம்*** எனப்படுகிறது.

ஒரு வங்கிக் கணக்கு, மாணவர், பறவை, கார், நாற்காலி போன்றவை நிலை மற்றும் பண்பியல்பு ஆகியவற்றை ஒருங்கே பெற்றுள்ளன என்பதை எளிதாகக் காணமுடியும். நடைமுறை வாழ்வில் காணப்படும் உண்மையான பொருள்களோடு உள்ள ஒற்றுமையே கணிப்பொறி நிரலின் பொருள்களுக்கு அவற்றின் சக்தியையும் பயன்பாட்டையும் வழங்குகின்றது. மென்பொருள் நிரல்களில் உண்மைப் பொருள்களை உருவகிக்கும் பணியை ***பொருள்கள்*** (Objects) எளிமைப்படுத்தியுள்ளன.

எடுத்துக்காட்டுப் பொருள்கள் - வங்கிக் கணக்கும் மாணவரும்

வங்கிக் கணக்கு	மாணவன்
தரவுகள்: Account number – long int Name – char[15] Opening balance –float; Account type – char	**தரவுகள்:** Date_of_birth – char[10] Name – char[15]; Class, sec char[4]; Marks float
செயற்கூறுகள்: Accept_Details() Display_Details() Update_opening_balance() Withdrawls() Deposit()	**செயற்கூறுகள்:** Accept_Details() Display_Details() Total() Average() Grading()

1.2 பல்லுருவாக்கம் (Polymorphism)

செவ்வகம், சதுரம், வட்டம், வளைகோடு போன்ற பல்வேறு வடிவங்களை வரையும் பணியை எடுத்துக் கொள்வோம். வெவ்வேறு வடிவங்களை வரைய வெவ்வேறு செயற்கூறுகளை வரையறுக்க வேண்டியிருக்கும். அத்தகைய வரையறைகள் இவ்வாறு இருக்கலாம்:

Draw_Square() Read side Drawrequired lines	Draw_Rectangle() Read length, breadth Draw required lines	Draw_Circle() Read radius Draw	Draw_Arc() Read Start_angle, End_angle,radius draw

கீழேயுள்ள செயற்கூறுகளை நோக்குக:

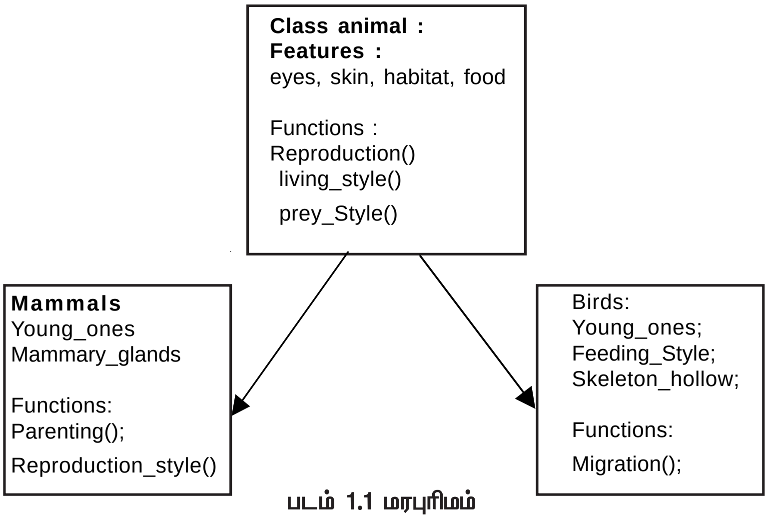
Draw (side) - ஒரு சதுரம் வரைவதற்கு
Draw (length, breadth) - ஒரு செவ்வகம் வரைவதற்கு
Draw (radius) - ஒரு வட்டம் வரைவதற்கு
Draw(radius, start_angle, end_angle) - ஒரு வளைகோடு வரைவதற்கு

Draw( ) என்னும் செயற்கூறு வேறுபட்ட உள்ளீடுகளை ஏற்கிறது. வேறுபட்ட செயல்பாட்டைக் கொண்டுள்ளது. பயனரைப் பொறுத்தவரையில் வேறுபட்ட வடிவங்களை வரைவதற்கு வேறுபட்ட உள்ளீடுகளுடன் Draw() செயற் கூறினைப் பயன்படுத்துவார். Draw( ) செயற்கூறு வேறுபட்ட உள்ளீடுகளின் அடிப்படையில் வேறுபட்ட முறைகளில் செயல்படும் தன்மையே ***பல்லுருவாக்கம்*** (Polymorphism) எனப்படுகிறது.

வேறுபட்ட செய்திகளுக்கு மாறுபட்டுச் செயல்படும் ஒரு பொருளின் திறனே ***பல்லுருவாக்கம்*** என்றழைக்கப்படுகிறது.

1.3 மரபுரிமம் (Inheritance)

இனக்குழு (Class) என்னும் தரவினம் நடப்புலகில் நிலவும் ஒரு பொருளை உருவகிக்கிறது. இனக்குழு என்பது பொதுவான பண்பியல்பு கொண்ட உரு பொருள்களின் ***வார்ப்புரு*** (Template) ஆகும். எடுத்துக்காட்டாக, உயிரினங்களின் குழுவை ***விலங்கினம்*** என்கிறோம். அதாவது, விலங்குகளை ஓர் இனக் குழுவாக வகைப்படுத்துகிறோம். விலங்குகளைப் பாலூட்டிகள், ஊர்வன, நிலம்-நீர் வாழ்வன, பூச்சிகள், பறவைகள் எனப் பலவாறாகப் பிரிக்க முடியும் என்பதை நாமறிவோம். அனைத்து விலங்குகளும் பொதுவான நடத்தையையும் பொதுவான பண்புக்கூறுகளையும் கொண்டுள்ளன. கண்கள், தோல், வாழிடம், உணவு ஆகியவற்றை விலங்குகளின் இயல்புகள் அல்லது பண்புக்கூறுகள் என்கிறோம். இனப்பெருக்கம், வாழும்முறை, இரைகொள்ளும் முறை போன்றவை விலங்குகளின் நடத்தை எனக் கொள்ளலாம். விலங்குகளின் ஒவ்வோர் உட்குழுவும் பொதுவான நடத்தையையும், இயல்பையும் கொண்டிருக்கும். அதே வேளையில் தத்தமக்கே உரிய தனித்த இயல்புகளையும் பாணிகளையும் கொண்டுள்ளன. பாலூட்டிகள், ஊர்வன, நிலம்-நீர் வாழ்வன, பூச்சிகள், பறவைகள் ஆகிய உட்குழுக்கள், விலங்கு என்னும் தாய்க்குழுவின் பண்புகளைப் பகிர்ந்து கொள்கின்றன. இதனைப் படமாக இவ்வாறு உருவகிக்கலாம்:



படம் 1.1 மரபுரிமம்

விலங்கு என்பது ***அடிப்படை இனக்குழு*** (base class) எனப்படுகிறது. பாலூட்டிகள், பறவைகள் என்பவை ***தருவிக்கப்பட்ட இனக்குழுக்கள்*** (derived classes) எனப்படுகின்றன. தருவிக்கப்பட்ட இனக்குழுக்கள் சக்தி மிக்கவை. காரணம் அவை தமக்குரிய தனித்த இயல்புகளோடு அடிப்படை இனக் குழுவின் செயலாற்றல்களையும் தம்மகத்தே கொண்டுள்ளன. அடிப்படை

இனக்குழுவின் பண்புகளை ஈட்டிக் கொள்ளும் தகைமை ***மரபுரிமம்*** (Inheritance) என்றழைக்கப்படுகிறது.

மரபுரிமம் என்பது தருவிக்கப்பட்ட இனக்குழுவின் செயலாற்றலை மிகுவிக்கிறது. நிரல் குறிமுறையின் மறுபயனாக்கத்தையும் (Code Reusability) சாத்தியமாக்கியுள்ளது.

பொருள்நோக்கு நிரலாக்கத்தின் பலன்கள்:

- ✓ நிரல்களில், தரவுகள் மற்றும் செயற்கூறுகளை ஒன்றுசேர்த்துப் பொருள்களாக ஒருங்கமைக்க ***இனக்குழு*** (Class) என்னும் தரவினம் பயன்படுகிறது.
- ✓ தொடர்பில்லாத வெளிச் செயற்கூறுகள் (இனக்குழுவுக்கு வெளியே வரையறுக்கப்பட்டுள்ள செயற்கூறுகள்) இனக்குழுவுக்குள் இருக்கும் தரவுகளை அணுக முடியாது. தரவுகளை அணுகத் தடை செய்வதுடன் ஒரு பொருளின் மிகத்தேவையான இயல்புகளை மட்டுமே வெளிக் காட்டுகிறது. இத்தகைய ***தரவு மறைப்பு*** (Data Hiding) அல்லது ***தரவு அருவமாக்கம்*** (Data Abstraction) தரவுகளுக்குப் பாதுகாப்பை நல்குகிறது.
- ✓ ஒரு செயற்கூறு (Function) அல்லது செயற்குறிக்கு (Operator) பல வரையறைகளைச் சாத்தியமாக்குதல் மூலம் ***பல்லுருவாக்கம்*** (Polymorphism) மென்பொருள் சிக்கற்பாட்டினை (Complexity) குறைக்கிறது.
- ✓ ***மரபுரிமம்*** (Inheritance), ஏற்கெனவே இருக்கும் ஓர் இனக்குழுவின் அடிப்படையில் வேறோர் இனக்குழுவைத் தருவித்துக் கொள்ள வழிவகுக்கிறது. இதன் மூலம் ***குறிமுறை மறுபயனாக்கம்*** (Code Resuability) சாத்தியமாகிறது. ஏற்கெனவே பயன்பாட்டில் உள்ள மென்பொருளில், புதுப்பிக்கப்பட்ட கூறுகளை இணைத்து நாள்தோறும் மாறிக் கொண்டிருக்கும் உலகின் தேவைகளை எதிர்கொள்ள வழி வகுக்கிறது.

1.4 ஒரு நடைமுறை எடுத்துக்காட்டு: வீட்டு நீர்ப்பயன்பாடு

வீட்டுக்கு நீர்வழங்கும் அமைப்பையொத்து ஒரு நிரலை உருவாக்குவதாகக் கொள்வோம். கட்டடத்திலுள்ள ஒவ்வொரு நீர் வெளியேற்றிகளில் வெளியேறும் நீரின் அளவையும், ஒட்டுமொத்த நீரின் நுகர்வையும் கணக்கிடுவதே இந்த நிரலின் நோக்கம். இந்த நிரலை எழுதுவதற்குத் தேவையான விவரங்கள்: அந்தக் கட்டடத்தில் நிறுவப்பட்டுள்ள நீர்த் திறப்பிகளின் (Taps) எண்ணிக்கை, ஒவ்வொரு திறப்பியின் வழியாகவும் வெளியேறும் நீரின் அளவு, இறுதியாக மொத்த நீரின் நுகர்வளவு. ஒவ்வொரு திறப்பியையும் ஒரு ***பொருளாக*** நோக்கலாம். இதனோடு தொடர்புடைய செயற்கூறுகள், நீர்ப்

படம் 1.2 வீட்டு நீர்ப்பயன்பாடு

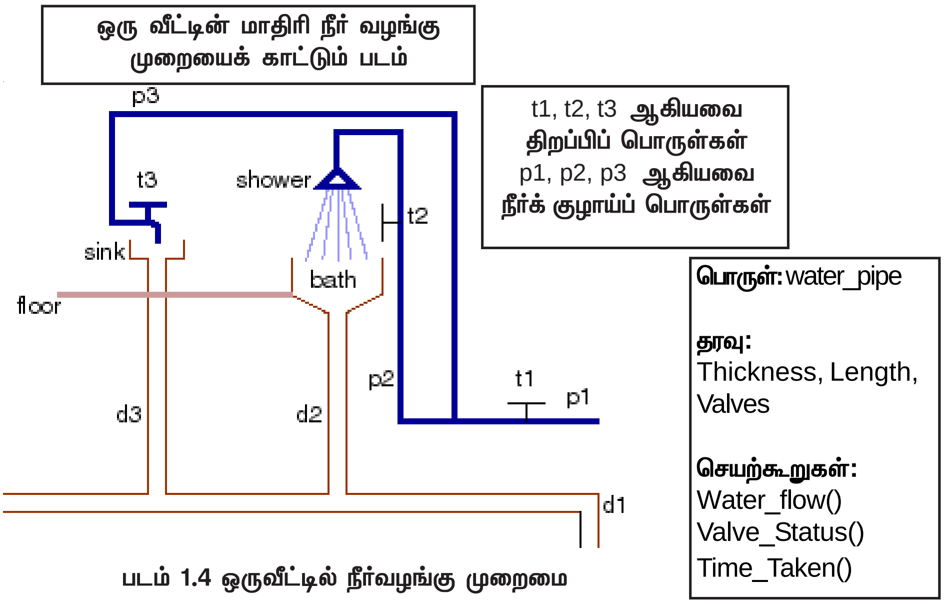
பாய்வை தொடங்குவது/ நிறுத்துவது மற்றும் குறிப்பிட்ட கால அளவில் நுகரப் 0படும் நீரின் அளவைத் தருகின்றதாகவும் இன்னும் இவற்றை ஒத்ததாகவும் இருக்கும். இந்தப் பணிகளைச் செய்வதற்குத் ***திறப்பி*** என்னும் பொருள், திறப்பி திறந்துள்ளதா, மூடியுள்ளதா என்பதைக் கண்காணிக்கவும், பயன் படுத்தப்பட்ட நீரின் அளவைக் குறிக்கவும், நீர் எங்கிருந்து வருகிறது என்பதை அறியவுமான சான்றுரு மாறிகளைக் (Instance Variables) கொண்டிருக்க வேண்டும். திறப்பி என்னும் பொருளை இவ்வாறு உருவகிக்கலாம்:

தரவு:
Tap_open, Qty_water, Water_source

செயற்கூறுகள்:
Start()
Stop()

படம் 1.3 திறப்பி என்னும் பொருள்

நீரின் பயன்பாட்டை உருவகிக்கும் நிரல், திறப்பிகளுக்கு நீரை வினியோக்கிக்கும் நீர்க்குழாய் (Water pipe) என்னும் பொருளையும் கொண்டி ருக்கும். அனைத்து நீர்க்குழாய்களையும் திறப்பிகளையும் ஒருங்கிணைக்கும் கட்டடம் (Building) என்னும் பொருளும் இருக்கலாம். எவ்வளவு நீர் நுகரப் பட்டுள்ளது எனக் கட்டடப் பொருளைக் கேட்டோம் எனில், அது ஒவ்வொரு குழாய் மற்றும் அதில் இணைக்கப்பட்டுள்ள திறப்பிகளின் தற்போதைய நிலை என்ன எனக் கேட்டறிந்து சொல்லும். இந்தத் திட்டப்பணியை படம் 1.4 -ல் உள்ளவாறு உருவகிக்கலாம்.



படம் 1.4 ஒருவீட்டில் நீர்வழங்கு முறைமை

நுகரப்பட்ட மொத்த நீரின் அளவு total_amount என்னும் மதிப்பு,

t1.water_consumed()+t2.water_consumed+t3.water_consumed()

என்று கணக்கிடப்படும். தனித்தனி திறப்பியிலிருந்து நுகரப்பட்ட நீரின் அளவு t1.water_consumed(), t2.water_consumed(),.... என்று அறியப்படும்.

t1.water_consumed() என்னும் செயற்கூறு அழைப்பு P1 என்னும் குழாய்ப் பொருளுடன் தொடர்புகொண்டு அந்தக் குழாய் வழியே பாய்ந்த நீரின் அளவைக் கேட்டறியும். காரணம் t1 என்னும் திறப்பி நுகரும் நீரின் அளவு P1 என்னும் குழாயினால் தீர்மானிக்கப்படுகிறது. இவ்வாறு ஒரு நிரலானது, இறுதி விடையைக் கணக்கிட ஒன்றையொன்று அழைக்கும் பொருள்களைக் கொண்டதாய் உள்ளது. ஒவ்வொரு பொருளும் தத்தம் பங்களிப்பை நல்கு கின்றன. அனைத்து நிரல் கூறுகளின் ஒருங்கிணைந்த செயல்பாடு, நிரலின் இறுதி விடையை வழங்குகின்றது. பொருள்கள், தரவுகளை உள்ளீடுகளாக அனுப்பித் தமக்குள்ளே தகவல் பரிமாற்றம் செய்துகொள்கின்றன.

இன்றைய நிரலாக்கத்தில் மாணவர்கள், மேலாளர்கள், வங்கிக் கணக்கு கள் இன்னும் இதுபோன்று நடப்புலகில் நாம்காணும் பொருள்கள் எடுத்தாளப் படுகின்றன. மிக இயல்பான முறையில் நடப்புலகப் பொருள்கள், நிரல்களில் கையாளப்படும் பொருள்களில் விளக்கம் பெறுகின்றன. இவ்வாறு ஒரு பொருளின் பண்பியல்பைச் செயற்கூறுகளாகப் பாவிக்கும் கருத்தோட்டமே பொருள்நோக்கு நிரலாக்கத்தின் குறிப்பிட்டுச் சொல்ல வேண்டிய சிறப்புக் கூறாகும்.

பயிற்சி வினாக்கள்

I. கோடிட்ட இடத்தை நிரப்புக

அ) நடப்புலகப் பொருள்களைக் கணிப்பொறி நிரலில் எடுத்தாள _________ கருத்துரு உதவுகிறது.

ஆ) தரவுகளையும் அவற்றைக் கையாளும் செயற்கூறுகளையும் ஒன்றாகப் பிணைத்துவைக்கும் செயல்நுட்பம் ______________________ என்றழைக்கப்படுகிறது.

இ) வேறுபட்ட செய்திகளுக்கு மாறுபட்டுச் செயல்படும் ஒரு பொருளின் திறனே______________________________ என்றழைக்கப்படுகிறது

ஈ) ஏற்கெனவே இருக்கும் தரவினத்தை அடிப்படையாகக் கொண்டு புதிய தரவினங்களை உருவாக்கும் செயல்நுட்பம்_____________________ என்றழைக்கப்படுகிறது.

II. கீழ்க்காணும் வினாக்களுக்குச் சுருக்கமாக விடை அளிக்கவும்:

1. ***பொருள்*** என்பதன் உட்கருத்து என்ன?
2. உறைபொதியாக்கம் என்பது என்ன?
3. பல்லுருவாக்கம் என்பது மரபுரிமத்திலிருந்து எவ்வாறு வேறுபடுகிறது?

III. கீழ்க்காணும் திட்டப்பணிக்கான தரவினங்களை வடிவமைக்கவும்

ஒரு நிறுவனம் தன்னுடைய நடவடிக்கைகளுக்காக ஒரு தரவு மாதிரியத்தை (Data Model) தயாரிக்க விரும்புகிறது. அந்நிறுவனம் தனது பணியாளர்கள் அனைவரின் தகவலையும் சேமித்து வைத்துள்ளது. பணியாளர்களின் பொதுவான விவரங்கள்: Name, date of birth, language, nativity.

பணியாளர்களின் பணிப்பொறுப்பு அடிப்படையில் சேமித்து வைக்கப்பட்டுள்ள கூடுதல் விவரங்கள்:

அ) Stores - date of joining, dept, salary

ஆ) Scientist - area of specialisation, current project details, paper presentations

இ) Technician - height, weight, ailments, risk factor, department, wages

பாடம் 2

சி++ மொழியின் முன்னோட்டம்

(Overview of C++)

2.1 முன்னுரை

சி++ மொழி 1980-களின் தொடக்கத்தில் ஏடீ&டீ பெல் ஆய்வுக் கூடத்தில் ***ஜேர்ன் ஸ்ட்ரௌஸ்ட்ரப்*** (Bjarne Stroustrup) அவர்களால் உருவாக்கப்பட்டது. சி++ என்னும் பெயரைச் (சி பிளஸ் பிளஸ் என உச்சரிக்க வேண்டும்) சூட்டியவர் ரிக் மாஸ்சிட்டி என்பவர் ஆவார். ++ என்பது சி மொழியின் மிகுப்புச் செயற்குறி ஆகும்.

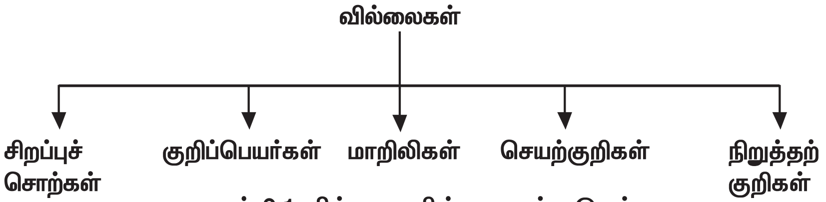
2.2 சி++ குறியுருத் தொகுதி (C++ Character Set)

சி- மொழியைப் போன்றே சி++ மொழியும் ஒரு குறியுருத் தொகுதியைக் கொண்டுள்ளது. அதிலிருந்து ***வில்லைகள்*** (Tokens- நிரலாக்கக் குறிமுறைக்குத் தேவையான உறுப்புகளின் அடிப்படை வகைகள்) உருவாக்கப் படுகின்றன. குறியுருத் தொகுதியில் கீழ்க்காணும் குறியுருக்கள் அடங்கியுள்ளன:

A........Z, a.......Z, 0........9
+, -, /, *, \, (,),[,], {, }
=, !=, <, >, ., ‘, “, ;, :
%, !, &, ?, -, #, <=, >=, @

வெற்று இடவெளி, கிடைமட்டத் தத்தல்,
செலுத்தி திருப்பல் மற்றும் பிற குறியுருக்கள்.

அடிப்படை இனங்கள் பற்றிச் சுருக்கமாகக் காண்போம். அடிப்பபடை இனங்கள் ஒட்டுமொத்தமாக வில்லைகள் (Tokens) எனப்படுகின்றன. வில்லை என்பது ஒரு நிரலில் உள்ள மீச்சிறு தனித்த அலகு ஆகும். வில்லைகள் படம் 2.1-ல் உள்ளவாறு வகைப்படுத்தப்படுகின்றன.



படம் 2.1 வில்லைகளின் வகைப்பாடுகள்

2.2.1 சிறப்புச் சொற்கள்

ஒரு கணிப்பொறி மொழியின் நிரல்பெயர்ப்பிக்குப் (Language Compiler) புரிகின்ற பொருள்கொண்ட சொற்கள் சிறப்புச் சொற்கள் எனப்படுகின்றன.

சிறப்புப் பயன்பாட்டுக்கென ஒதுக்கப்பட்ட சொற்கள் இவை. இச்சொற்களைச் சாதாரண குறிப்பெயர்களாகப் பயன்படுத்த முடியாது. சி++ மொழியில் பயன் படுத்தப்படும் சிறப்புச் சொற்களின் பட்டியலை அட்டவணை 2.1-ல் காண்க.

auto	break	case	const	class	continue
default	delete	do	else	enum	for
friend	goto	if	inline	new	operator
private	protected	public	return	signed	sizeof
static	struct	switch	this	unsigned	virtual
while					

அட்டவணை 2.1 சிறப்புச் சொற்கள்

2.2.2 குறிப்பெயர்கள் (Identifiers)

குறிப்பெயர்கள், ***மாறிகள்*** (Variables) என்றும் அழைக்கப்படுகின்றன. மாறிகள் என்பவை, மதிப்பு அல்லது மாறிலிகளைத் தாங்கியுள்ள நினைவகப் பெட்டிகளைக் குறிக்கின்றன. மாறியின் பெயர் ஓர் எழுத்தில் அல்லது அடிக்கீறில்(Underscore) தொடங்கி, அதனைத் தொடர்ந்து எழுத்து அல்லது எண்களைக் கொண்டிருக்கும். எடுத்துக்காட்டாக _test; test; sum12 ஆகியவை ஏற்கத்தகு குறிப்பெயர்களாகும். தரவு இனங்களைப் பற்றிப் படித்த பிறகு மாறிகளைப் பற்றி மேலும் பார்ப்போம்.

2.2.3 மாறிலிகள் (Constants)

மாறிலி என்பது மாற்ற முடியாத மதிப்பைக் கொண்ட ஒரு தரவு விவரம் ஆகும். மாறிலி என்பது எண்வகையைச் சேர்ந்ததாகவோ, எண் அல்லாத வகையைச் சார்ந்ததாகவோ இருக்கலாம். எண்வகை மாறிலி எண் மதிப்புகளையே கொண்டிருக்கும். முழு எண்ணாக இருக்கலாம். பதின்மப் புள்ளி எண்ணாகவும் (decimal numbers) இருக்கலாம். முழுஎண்கள் (Integers) மற்றும் மிதப்புப் புள்ளி (floating point) எண்கள் எண்வகை மாறிலிகள் ஆகும்.

2.2.4 முழுஎண் மாறிலி (Integer Constant)

முழுஎண் மாறிலி என்பது குறைந்தது ஓர் இலக்கத்தையாவது கொண்டிருக்க வேண்டும். பின்னப்பகுதி எதுவும் இருக்கக்கூடாது.

- + அல்லது - என்கிற முன்னொட்டுக் குறி (prefix) இருக்கலாம்.
- 0 - வில் தொடங்கும் எண்கள் எண்ம (Octal) மாறிலிகளாகக் கருதப்படும். (எ-டு) 010 = 8 $([8]_{10} = [10]_{8})$.
- 0X எனத் தொடங்கும் எண்கள் பதினறும (hexadecimal) எண்களாகக் கருதப்படும். (எ-டு) 0XF=15 $([15]_{10} = [F]_{16})$.

2.2.5 மிதப்புப் புள்ளி மாறிலி (Floating Point Constant)

மிதப்புப் புள்ளி மாறிலி என்பது குறியிட்ட மெய் எண்ணைக் குறிக்கும். அது, முழுஎண் பகுதி, பதின்மப் புள்ளி, பின்னப் பகுதி, அடுக்கெண் பகுதி ஆகிய நான்கு பகுதிகளைக் கொண்டிருக்கும். ஒரு மிதப்புப் புள்ளி எண் மாறிலியைக் குறிப்பிடும்போது முழுஎண் பகுதி, பின்னப் பகுதி ஆகிய இரண்டில் ஒன்றை விட்டுவிடலாம். ஆனால் இரண்டும் இல்லாமல் இருக்க முடியாது. எடுத்துக்காட்டாக 58.64 என்பது ஏற்கத்தகு மிதப்புப் புள்ளி எண் (மெய்யெண்) ஆகும். இந்த எண்ணை அடுக்கெண் முறையில் கீழ்க்காணுமாறு குறிப்பிடலாம்:

- 5.864E1 => $5.864*10^1$ => 58.64
- 5.864E-2 => $5864*10^{-2}$ => 58.64
- 0.5864E2 => $0.5864*10^2$ => 58.64

மிதப்புப் புள்ளி எண்களை அடுக்கெண் முறையில் குறிப்பிடும்போது E அல்லது e என்னும் எழுத்தைப் பயன்படுத்த வேண்டும்.

2.2.6 குறியுரு மாறிலி (Character Constant)

குறியுரு மாறிலி என்பது ஒற்றை மேற்கோள் குறிகளுக்குள் தரப்படும் ஒற்றைக் குறியுருவைக் கொண்டிருக்கும். சி++ மொழியின் குறியுருத் தொகுதியில் வரையறுக்கப்பட்டுள்ளன எந்தக் குறியுருவாகவும் (ASCII தொகுதியின் அங்கமான எண்கள் கணித/ஒப்பீட்டு/பிற சிறப்புக் குறியுருவாகவும்) இருக்கலாம். தத்தல் (tab), பின்னிடவெளி (backspace) வரிச்செலுத்தி (line feed), மதிப்பிலி (null), பின்சாய்வு (backslash) ஆகிய சில குறியுருக்கள் ***விடுபடு வரிசையாக*** (escape sequence) குறிப்பிடப்படுகின்றன. விடுபடு வரிசைகள் பின்சாய்வுக் கோட்டை முன்னொட்டாகக் கொண்டு ஒற்றை எழுத்தால் குறிப்பிடப்படும். அட்டவணை 2.2-ல் விடுபடு வரிசைக் குறியுருக் களைக் காண்க.

விடுபடு வரிசை	வடிவற்ற குறியுரு
\a	மணி ஒலிப்பு
\b	பின்னிடவெளி
\n	புதிய வரி/ வரிச்செலுத்தி
\t	கிடைமட்டத் தத்தல்
\v	செங்குத்துத் தத்தல்
\\	பின்சாய்வுக் கோடு
\' (அ) \"	ஒற்றை/இரட்டை மேற்கோள்
\o	எண்ம எண்
\X	பதினறும எண்
\0	மதிப்பிலி

அட்டவணை 2.2. விடுபடு வரிசைக் குறியுருக்கள்

2.2.6 சர நிலையுரு (String Literal)

சர நிலையுரு என்பது இரட்டை மேற்கோள் குறிகளுக்குள் தரப்படும் குறியுருக்களின் வரிசையைக் கொண்டிருக்கும். சர நிலையுருக்கள் குறியுருக்களின் அணியாகக் (array of characters) கருதப்படுகின்றன. ஒவ்வொரு சர நிலையுருவும் தானமைவாக (by default) '\0' என்னும் சிறப்புக் குறியுருவை ஈற்றில் இணைத்துக் கொள்ளும். இது சரத்தின் இறுதியைக் குறிக்கிறது. (எ-டு) : "testing"

2.2.7 செயற்குறி (Operator)

செயற்குறி என்பது ஒரு மதிப்பை விடையாகப் பெறுவதற்கு நிகழ்த்தப்பட வேண்டிய ஒரு செயல்பாட்டைக் குறிக்கிறது. செயற்குறியானது ***செயலேற்பி களின்*** (Operands) மீது செயல்படுகின்றது.

எடுத்துக்காட்டு :

RESULT = NUM1+ NUM2

இதில், NUM1, NUM2 ஆகியவை செயலேற்பிகள். + என்பது இரண்டு எண் களைக் கூட்டி விடைதரும் கூட்டல் செயற்குறி ஆகும். = என்னும் மதிப்பிருத்தல்(assignment) செயற்குறியின் உதவியுடன் விடை மதிப்பு, RESULT என்னும் மாறியில் சேமிக்கப்படுகிறது. சி++ மொழியில் பயன் படுத்தப்படும் செயற்குறிகளின் பட்டியலை அட்டவணை 2.3-ல் காண்க.

[ ]	*	%	==	=	>=
( )	+	<<	!=	*=	&=
.	-	>>	^	/=	^=
->	~	<	|	+=	|=
++	!	>	&&	-=	, –
	size of	<=	||	%=	#
&	/	>=	?:	<<=	##

அட்டவணை 2.3 சி++ மொழியின் செயற்குறிகள்

கீழ்க்காணும் செயற்குறிகள் சி++ மொழிக்கே உரித்தானவை:

:: .* ->*

மற்றும் ## ஆகிய செயற்குறிகள் முன்செயலியால் (Preprocessor) பயன்படுத்திக் கொள்ளப்படுபவை.

செயற்குறிகளை இவ்வாறு வகைப்படுத்தலாம்:

- கணக்கீடு (Arithmatic)
- மதிப்பிருத்தல் (Assignment)
- பொருட்கூறுத் தெரிவு (Component Selection)
- நிபந்தனை (Conditional)
- தருக்கம் (Logical)
- கையாளுகை (Manipulator)
- உறுப்பினர் குறிப்பு விலக்கம் (Member dereferencing)
- நினைவக மேலாண்மை (Memory Management)
- முன்செயலி (Preprocessor)
- ஒப்பீடு (Relational)
- வரையெல்லைக் கூறுபாடு (Scope Resolution)
- நகர்வு (Shift)
- இனமாற்றம் (Type Cast)

இடம்பெறும் செயவேலற்பிகளின் எண்ணிக்கை அடிப்படையில் செயற்குறிகளை ஒருமம் (Unary) இருமம் (Binary), மும்மம் (Ternary) என மூவகையாகப் பிரிக்கலாம்.

எடுத்துக்காட்டு:

ஒருமச் செயற்குறிக்கு ஒரு செயலேற்பியே தேவை. இருமச் செயற்குறிக்கு இரு செயலேற்பிகள் தேவை. மும்மச் செயற்குறிக்கு மூன்று செயலேற்பிகள் தேவை.

செயற் குறி	**விளக்கம்**
&	முகவரி சுட்டல்
!	தருக்க எதிர்மறை
*	உள்நோக்கல்
++	மிகுப்பு
~	பிட்நிலை நிரப்பு
—	குறைப்பு
-	ஒருும எதிர்மம்
+	ஒருும நேர்மம்

படம் 2.4(அ)
ஒருமச் செயற்குறிகள்

<table>
<tr><th>வகை</th><th>செயற்குறி</th><th>விளக்கம்</th></tr>
<tr><td>கூட்டல்</td><td>+</td><td>இருமக் கூட்டல்</td></tr>
<tr><td></td><td>-</td><td>இருமக் கழித்தல்</td></tr>
<tr><td>பெருக்கல்</td><td>*</td><td>பெருக்கல்</td></tr>
<tr><td></td><td>/</td><td>வகுத்தல்</td></tr>
<tr><td></td><td>%</td><td>வகுமீதி</td></tr>
<tr><td>நகர்வு</td><td><<</td><td>இடது நகர்வு</td></tr>
<tr><td></td><td>>></td><td>வலது நகர்வு</td></tr>
<tr><td>பிட்நிலை</td><td>&</td><td>உம்</td></tr>
<tr><td></td><td>|</td><td>அல்லது</td></tr>
<tr><td></td><td>^</td><td>தனித்த அல்லது</td></tr>
<tr><td>தருக்கம்</td><td>&&</td><td>தருக்க உம்</td></tr>
<tr><td></td><td>||</td><td>தருக்க அல்லது</td></tr>
<tr><td>மதிப்பிருத்தல்</td><td>=</td><td>மதிப்பிருத்தல்</td></tr>
<tr><td></td><td>/=</td><td>வகுத்து மதிப்பிருத்தல்</td></tr>
<tr><td></td><td>+=</td><td>கூட்டி மதிப்பிருத்தல்</td></tr>
<tr><td></td><td>*=</td><td>பெருக்கி மதிப்பிருத்தல்</td></tr>
<tr><td></td><td>%=</td><td>வகுமீதி மதிப்பிருத்தல்</td></tr>
<tr><td></td><td>-=</td><td>கழித்து மதிப்பிருத்தல்</td></tr>
<tr><td></td><td><<=</td><td>இடம்நகர்ந்து மதிப்பிருத்தல்</td></tr>
<tr><td></td><td>>>=</td><td>வலம்நகர்ந்து மதிப்பிருத்தல்</td></tr>
<tr><td></td><td>&=</td><td>பிட்நிலை உம் மதிப்பிருத்தல்</td></tr>
<tr><td></td><td>|=</td><td>பிட்நிலை அல்லது மதிப்பிருத்தல்</td></tr>
<tr><td>ஒப்பீடு</td><td><</td><td>விடச்சிறிது</td></tr>
<tr><td></td><td>></td><td>விடப்பெரிது</td></tr>
<tr><td></td><td><=</td><td>விடச்சிறிது அல்லது நிகர்</td></tr>
<tr><td></td><td>>=</td><td>விடப்பெரிது அல்லது நிகர்</td></tr>
<tr><td>நிகர்ப்பாடு</td><td>==</td><td>நிகர்</td></tr>
<tr><td></td><td>! =</td><td>நிகரில்லை</td></tr>
<tr><td>பொருட்கூறு தெரிவு</td><td>.</td><td>நேரடிப் பொருட்கூறு தெரிவு</td></tr>
<tr><td></td><td>-></td><td>மறைமுகப்பொருட்கூறுதெரிவு</td></tr>
<tr><td>இனக்குழு உறுப்பினர்</td><td>: :</td><td>வரையெல்லை அணுகல்/ கூறுபாட்டுச் செயற்குறி</td></tr>
<tr><td></td><td>. *</td><td>குறிப்பு விலக்குச் செயற்குறி</td></tr>
<tr><td>நிபந்தனை</td><td>-> *</td><td>இனக்குழு உறுப்புக்கான குறிப்பு விலக்கம்</td></tr>
<tr><td></td><td>?:</td><td>மும்மச் செயற்குறி</td></tr>
<tr><td>காற்புள்ளி</td><td>,</td><td>மதிப்பிடல்</td></tr>
</table>

அட்டவணை 2.4(ஆ) இருமச் செயற்குறிகள்

2.2.7.1 கணக்கீட்டுச் செயற்குறிகள் (Arithmatic Operators)

கணக்கீட்டுச் செயற்குறிகள் கணிதச் செயல்பாடுகளை நிறைவேற்றப் பயன்படுத்தப்படுகின்றன. கணக்கீட்டுச் செயற்குறிகளின் பட்டியல் காண்க:

செயற்குறி	பயன்பாடு
+	கூட்டல்
-	கழித்தல்
*	பெருக்கல்
/	வகுத்தல்
%	முழுஎண் வகுமீதி
+=, -= *=, /= %=	கணக்கிட்டு மதிப்பிருத்தல்

கணக்கீட்டுச் செயற்குறிகள், எண்வகை மாறிலிகள்/மாறிகள், கணக்கீட்டுச் செயற்குறிகளால் இணைக்கப்பட்ட செயற்கூறு அழைப்பு ஆகியவற்றை உள்ளடக்கி, கணக்கீட்டுக் கோவைகள் (arithmatic expressions) அமைக்கப்படுகின்றன.

எடுத்துக்காட்டுகள்:

- a = -5;
- a = +b;
- a/=5; (a=a/5)
- a++; (பின் மிகுப்புச் செயற்குறி. a = a+1 என்பதற்கு நிகர்)
- a- -; (பின் குறைப்புச் செயற்குறி. a = a-1என்பதற்கு நிகர்)
- ++a; (முன் மிகுப்புச் செயற்குறி. a = a+1 என்பதற்கு நிகர்)
- - -a; (முன் குறைப்புச் செயற்குறி. a = a-1 என்பதற்கு நிகர்)
- a *= 2; (a = a*2)
- a %= 5; (a = a%5)
- a = b + pow(x,y); (a = b + x^y)

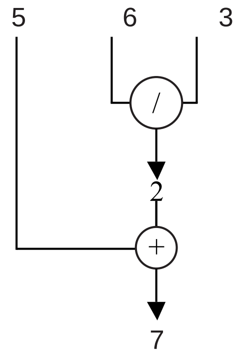
செயற்குறிகள் முன்னுரிமையின் அடிப்படையில் நிறைவேற்றப்படுகின்றன. செயலேற்பிகளும், செயற்குறிகளும் குறிப்பிட்ட தருக்க முறைப்படி குழுவாக்கப்பட்டு மதிப்பிடப்படுகின்றன. இத்தகைய தருக்கக் குழுவாக்கம் ***தொடர்புறுத்தம்*** (association) எனப்படுகிறது. ***திசைமுகம்*** என்றும் கூறலாம். கணிதச் செயற்குறிகளின் வகை மற்றும் தொடர்புறுத்தத்தை அட்டவணை 2.6 -ல் காண்க.

செயற்குறி முன்னுரிமை	வகை	திசைமுகம்
( ), [ ]	கணிதம்	இடமிருந்து வலம்
பின்னொட்டு ++,—	கணிதம்-ஒருமம்	இடமிருந்து வலம்
முன்னொட்டு ++, — ஒருமம +, ஒருமம -	கணிதம்-ஒருமம்	வலமிருந்து இடம்
*. /. %	கணிதம்-இருமம்	இடமிருந்து வலம்
+, -	கணிதம்-இருமம்	இடமிருந்து வலம்

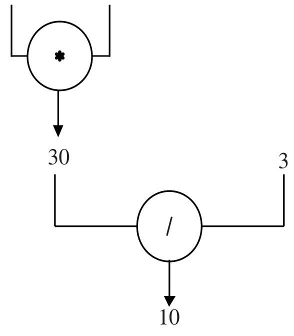
அட்டவணை 2.6 கணிதச் செயற்குறிகளின் முன்னுரிமை

கீழ்க்காணும் எடுத்துக்காட்டுகள், கணக்கீட்டுக் கோவைகள் மதிப்பிடப் படும் வரிசைமுறையை விளக்குகின்றன.

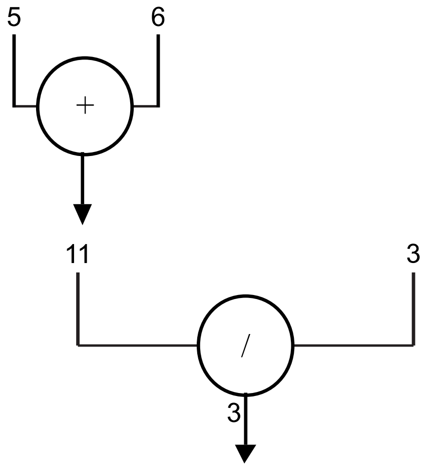
5 + 6/3 = 7



5 * 6/3 = 10

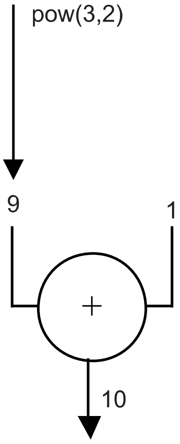


(5 + 6) /3 =3

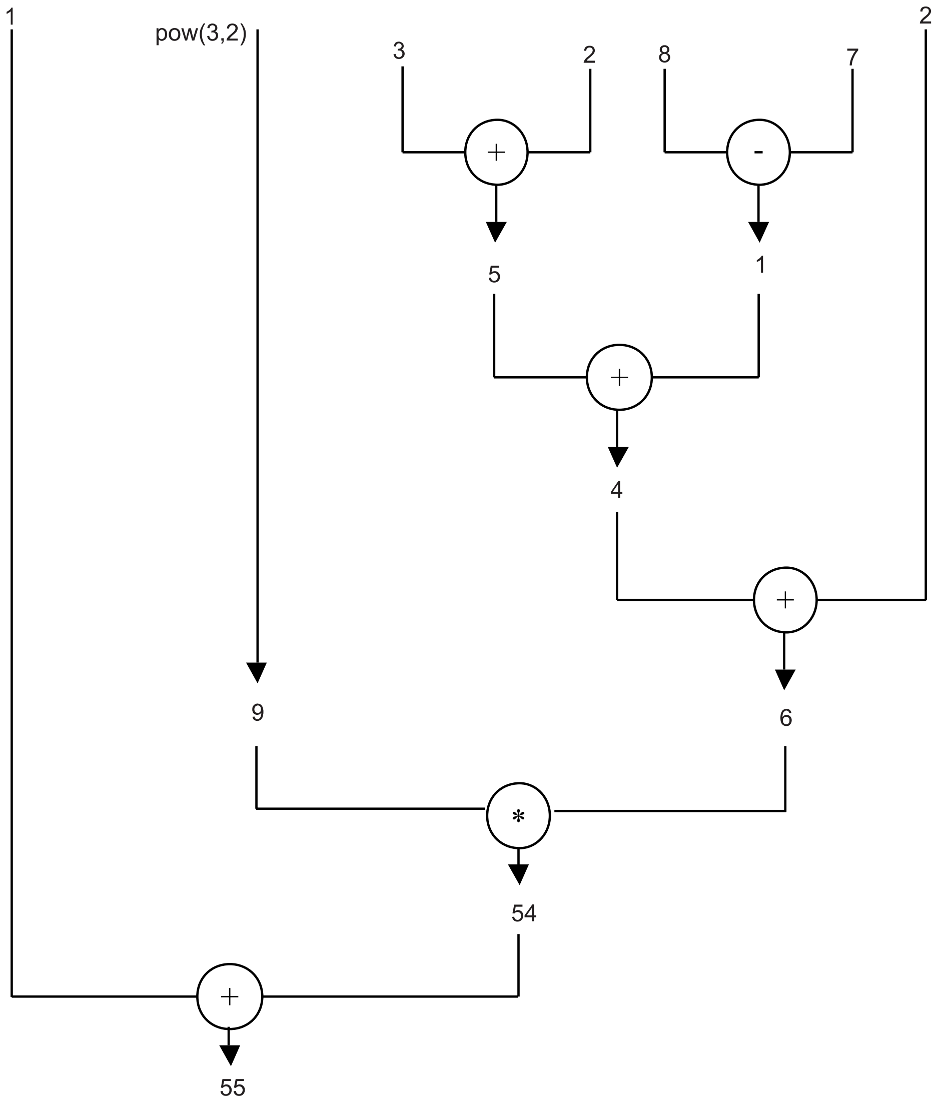


அனைத்து உள்ளீடுகளும் int இனம் என்பதால் விடை 3 ஆகும். ஏதேனும் ஓர் உள்ளீடு float இனம் எனில், விடை 3.66 எனக் கிடைக்கும்.

1 + pow (3, 2) = 10



1 + pow (3, 2) * [(3 + 2) (8 - 7) + 2] = 55



மிகுப்பு, குறைப்புச் செயற்குறிகள் சி மொழியில் உள்ளவாறே சி++ மொழியிலும் செயல்படுத்தப்படுகின்றன. இச்செயற்குறிகளைக் கொண்ட கோவைகள், மதிப்பிடப்படும் முறைகளை அட்டவணை 2.7-ல் காண்க.

கோவை	செயல்பாடு	எடுத்துக்காட்டு

a++	a-ன் மதிப்பைப் பெற்று அந்த மதிப்பில் 1 மிகுக்கவும்	a = 5;
		c = a++;
		செயலாக்கம்:
		c = a;
		a = a+1;
		எனவே, c என்னும் மாறியில் இருத்தப்படும் மதிப்பு 5 ஆகும்.

++a	a-ன் மதிப்பில் 1 மிகுத்து அந்த மதிப்பைப் பெறுக	a = 5;
		c = ++a;
		செயலாக்கம்:
		a = a+1;
		c = a;
		எனவே, c- என்னும் மாறியில் இருத்தப்படும். மதிப்பு 6 ஆகும்.

a --	a-ன் மதிப்பைப் பெற்று அந்த மதிப்பில் 1 குறைக்கவும்	a=5;
		c = a--;
		செயலாக்கம்:
		c = a;
		a = a-1;
		இப்போது c- ன் மதிப்பு என்னவாக இருக்கும்?

--a	a-ன் மதிப்பில் ஒன்றைக் குறைத்து அந்த மதிப்பைப் பெறுக	a = 5;
		c = --a;
		செயலாக்கம்:
		a = a-1;
		c = a;
		இப்போது c-ன் மதிப்பு என்னவாக இருக்கும்?

அட்டவணை 2.6 மிகுப்பு, குறைப்புச் செயற்குறிகள்

அட்டவணை 2.7- ல் தரப்பட்டுள்ள கட்டளைத் தொகுதிகளில் மாறிகளில் இருத்தப்படும் மதிப்புகள் என்னவாக இருக்கும்?

1. a = 5;
b = 5;
a = a+b++;
மாறியில் இருத்தப்படும் மதிப்பு______
	2. x = 10;
	f = 20;
	c = x++ + ++f;
	மாறியில் இருத்தப்படும் மதிப்புகள்
	c-ல்________
	x-ல்________
	f-ல்________
		3. fun=1;
		sim=2;
		final= - -fun + ++sim-fun- -
		மாறியில் இருத்தப்படும் மதிப்புகள்
		fun-ல்________
		sim-ல்_______
		final-ல்_______

அட்டவணை 2.7 எளிய வினாக்கள்

2.2.7.2 **ஒப்பீட்டுச் செயற்குறிகள்** (Relational Operators)

மதிப்புகளை ஒப்பிட ஒப்பீட்டுச் செயற்குறிகள் பயன்படுத்தப்படுகின்றன. ஒப்பீட்டுச் செயற்குறிகளின் பட்டியல்:

- == நிகரானது
- > விடப்பெரிது
- < விடச்சிறிது
- >= விடப்பெரிது அல்லது நிகர்
- <= விடச்சிறிது அல்லது நிகர்
- != நிகரில்லை

ஒப்பீட்டுச் செயற்குறிகள் மதிப்புகளை ஒப்பிடப் பயன்படுத்தப்படுகின்றன. இரண்டு செயலேற்பிகள் ஓர் ஒப்பீட்டுச் செயற்குறியால் இணைக்கப்பட்டு ஒப்பீட்டுக் கோவைகள் அமைக்கப்படுகின்றன. எடுத்துக்காட்டாக, கீழ்க்காணும் நிபந்தனைகளை அமைக்க ஒப்பீட்டுச் செயற்குறிகள் பயன் பட்டுள்ளன.

- 10 > 20
- 500.45 <= 1005
- 99 != 99.5
- 9 == 9

ஒப்பீட்டுச் செயல்பாடு, ***சரி*** (true) அல்லது ***தவறு*** (false) என்கிற விடையைத் தரும். எண் மாறிலி 0 (சுழியம்) தவறு (false) என்னும் மதிப்பைக் குறிக்கிறது. சுழியம் அல்லாத எந்த எண் மதிப்பும் சரி (true) என்பதைக் குறிக்கும். மேற்கண்ட கோவைகளின் வெளியீடு இவ்வாறு இருக்கும்:

- 0 (10>20 என்பது தவறு)
- 1 (500.45<1005 என்பது சரி. எனவே ஒரு சுழியமில்லா மதிப்பு)
- 1 (99!=99.5 என்பது சரி. எனவே ஒரு சுழியமில்லா மதிப்பு)
- 1 (9==9 என்பது சரி. எனவே ஒரு சுழியமில்லா மதிப்பு)

கீழ்க்காணும் கோவையின் மதிப்பு என்னவாக இருக்கும்?

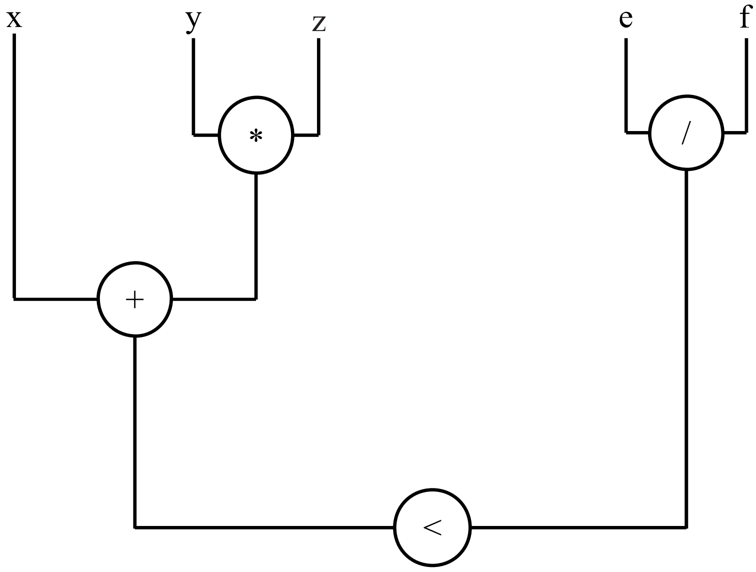
num1=99, num2=20, num3=10 எனில் (num1+num2-num3)/5*2<(num1%10) எவ்வளவு?

அட்டவணை 2.8 -ல் உள்ள ஒப்பீட்டுக் கோவைகளை மதிப்பிடுக:

செயற்குறி	**கோவை**	**விடை**
==	5==6	0
!=	‘a’!=’a’	
>	5>6	
	‘a’>’A’	
<	5<6	
	‘a’<‘A’	
>=	‘a’>=’z’	
	5>=5	
<=	‘a’<=’z’	
	5<=6	

அட்டவணை 2.8 ஒப்பீட்டுக் கோவைகளை மதிப்பிடுக

ஒப்பீட்டுச் செயற்குறிகள், கணக்கீட்டுச் செயற்குறிகளைவிடக் குறைந்த முன்னுரிமை உடையவை. எடுத்துக்காட்டாக, x+y*z< e/f என்னும் கோவை இவ்வாறு மதிப்பிடப்படும்:



2.2.7.3 தருக்கச் செயற்குறிகள் (பூலியன் செயற்குறிகள்)

ஒன்று அல்லது ஒன்றுக்கு மேற்பட்ட நிபந்தனைகளின் விடைகளை தருக்கச் செயற்குறிகள் ஒன்றிணைக்கின்றன. && (AND), || (OR) ! (NOT) – ஆகியவை தருக்கச் செயற்குறிகள் ஆகும்.

எடுத்துக்காட்டு:

c=5. d=6. choice='y', term='2' (***சரி*** என்பதை 1 எனவும், ***தவறு*** என்பதை 0 எனவும் கருதிக் கொள்க).

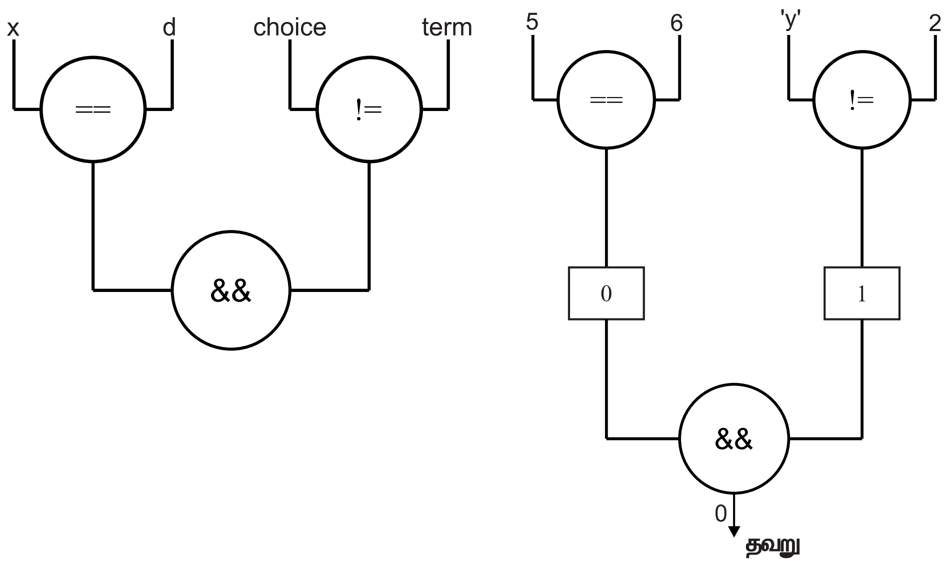
Result_1 = (c == d) && (choice != term)
Result_2 = (‘y’ == ‘Y’) || (term != ‘0’)

Result_3 = (c==d) &&(‘y’ == ‘Y’) || (choice != term)
Result_4 = (c==d) || (‘y’ == ‘Y’) && (choice != term)

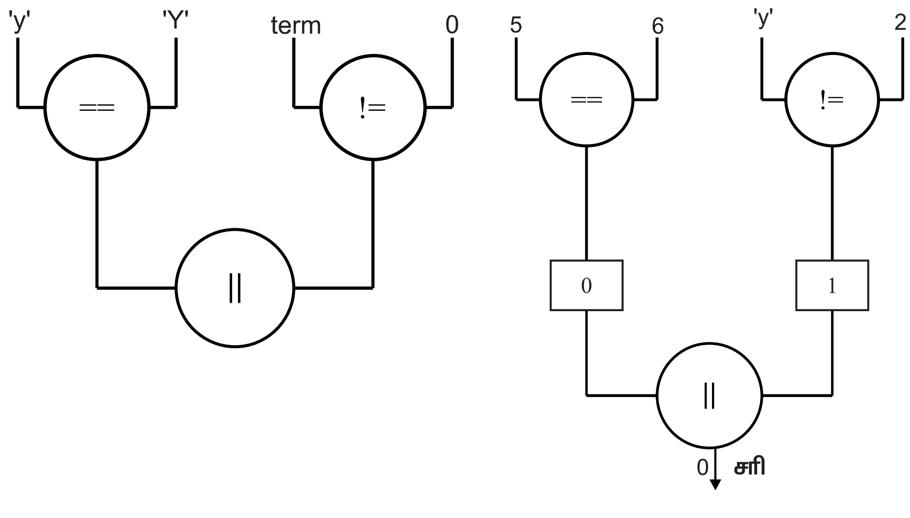
Result_1, Result_2 ஆகியவற்றில் இருத்தப்பட்ட மதிப்பு யாது?

Result_1-ல் உள்ள மதிப்பு 0 (தவறு); Result_2-ல் உள்ள மதிப்பு 1 (சரி)
Result_3-ல் உள்ள மதிப்பு 1 (சரி); Result_4-ல் உள்ள மதிப்பு 0 (தவறு)

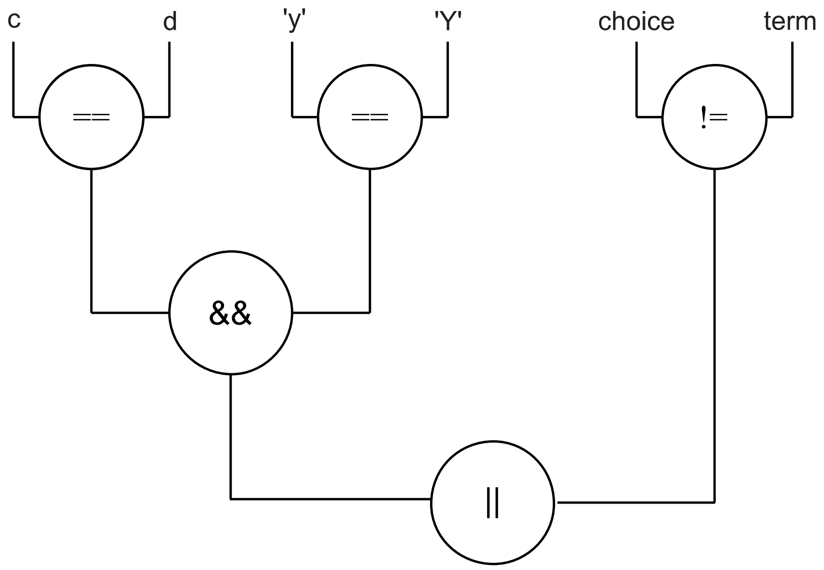
Result_1 = (c == d) && (choice != term) மதிப்புகளைப் பதிலீடு செய்யின்



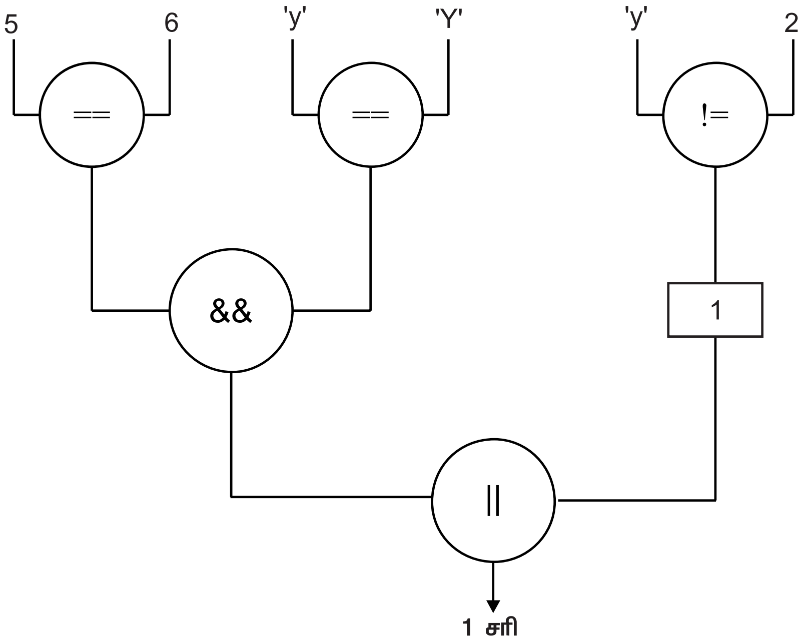
Result_2 = (‘y’ == ‘Y’)&& (term != ‘0’) மதிப்புகளைப் பதிலீடு செய்யின்



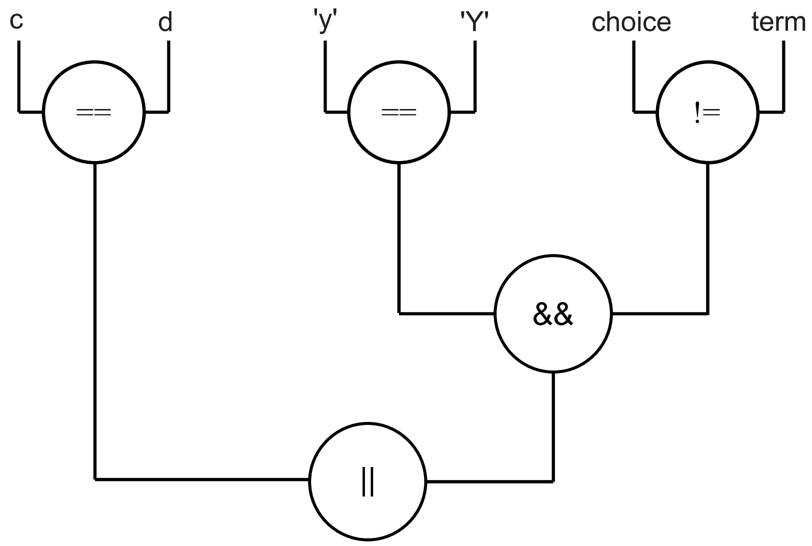
Result_3 = (c == d) && (‘y’ = = ‘Y’) || (choice != term)



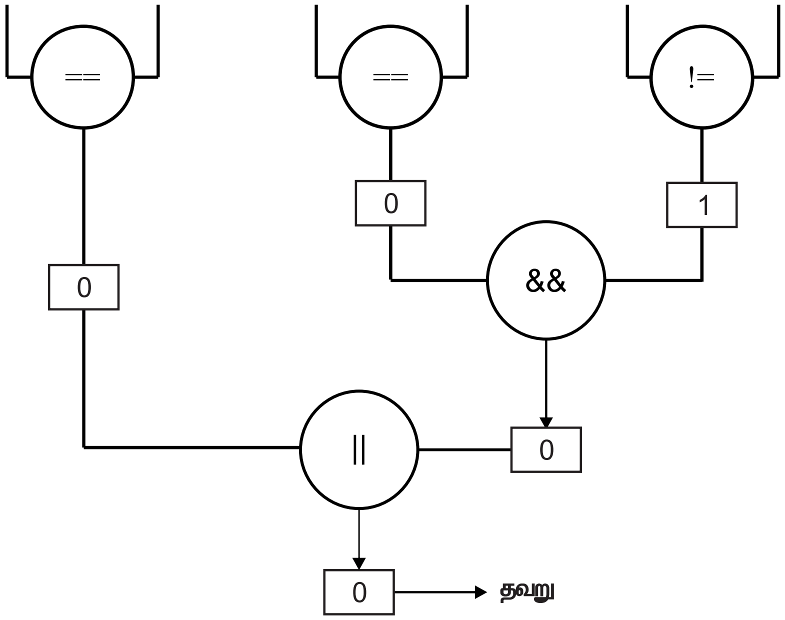
மதிப்புகளைப் பதிலீடு செய்யின்,



Result_4 = (c == d) || (‘y’ = = ‘Y’) && (choice != term)

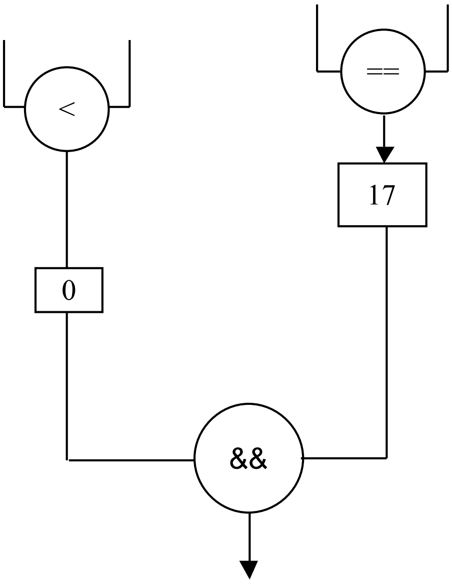


மதிப்புகளைப் பதிலீடு செய்யின்,



தருக்கச் செயற்குறிகள், ஒப்பீட்டுச் செயற்குறிகள் மற்றும் கணக்கீட்டுச் செயற்குறிகளைவிடக் குறைந்த முன்னுரிமை கொண்டவை. பின்வரும் கோவையின் மதிப்பைக் கண்டறிய முடியுமா?

5<4 && 8+9



0 – தவறு

2.2.7.4 நிபந்தனைச் செயற்குறி (?:) (Conditional Operator)

நிபந்தனைச் செயற்குறி,

(num1>num2) ? ''சரி'': ''தவறு'';

என அமையும். ? : என்பது மும்மச் செயற்குறி. (num1>num2), ''சரி'', ''தவறு'' - ஆகியவை செயலேற்பிகள். மும்மச் செயற்குறியை ***நிபந்தனைச் செயற்குறி*** என்றும் அழைக்கிறோம். இச்செயற்குறியின் பொதுவான கட்டளை அமைப்பு,

E1? E2: E3;

என அமையும். E1, E2, E3, ஆகியவை செயலேற்பிகள் ஆகும். E1 என்பது ஒரு நிபந்தனையாக இருக்கும். E2, E3 ஆகியவை மதிப்புகளாகவோ, கூற்றுகளாகவோ இருக்கலாம். எடுத்துக்காட்டாக, இரண்டு எண்களில் பெரிய எண்ணை ஒரு மாறியில் இருத்துவதற்கு,

max=(num1>num2) ? num1: num2;

எனக் கட்டளை அமைக்கலாம். num1 என்னும் மதிப்பு num2-ஐக் காட்டிலும் பெரிது எனில் max-ல் num1 இருத்தப்படும். இல்லையேல், max-ல் num2 இருத்தப்படும்.

கீழேயுள்ள கட்டளைத் தொகுதியில், x என்னும் மாறியில் இருத்தப்படும் மதிப்பு என்னவென்று சொல்ல முடியுமா?

```
a = 10;
b = 10;
x = (a<b)? a*a : b%a;
```

2.2.7.5 மதிப்பிருத்து செயற்குறிகள் (Assignment Operators)

= என்பது சாதாரண மதிப்பிருத்து செயற்குறி ஆகும். ஒரு கோவையின் (செயற்குறியின் வலப்பக்கம் இருக்கும்), விடையை ஒரு மாறியில் (செயற்குறி யின் இடப்பக்கம் இருக்கும்) இருத்துவதற்கு இது பயன்படுத்தப்படுகிறது. இந்தச் சாதாரண மதிப்பிருத்து செயற்குறி தவிர வேறு பத்து குறுக்குவழி மதிப்பிருத்து செயற்குறிகளும் உள்ளன. அனைத்து மதிப்பிருத்து செயற்குறி களையும் கீழேயுள்ள அட்டவணை 2.9-ல் காண்க.

கோவை	**செயலாக்கம்**	**விடை**
A = 5	A என்னும் மாறியில் 5என்னும் மதிப்பு இருத்தப்படுகிறது.	மாறி 5 என்னும் மதிப்பைப் பெறுகிறது
A += 2	A += 2 என்பதன் பொருள் A = A +2 ஆகும்.	A- யில் இருத்தப்படும் மதிப்பு 7.
A *= 4	A = A * 4	A- யில் இருத்தப்படும் மதிப்பு 20.
A /= 2	A = A / 2	A- யில் இருத்தப்படும் மதிப்பு 2.
A - = 2	A = A – 2	A- யில் இருத்தப்படும் மதிப்பு 3.
A %= 2	A = A % 2	A- யில் இருத்தப்படும் மதிப்பு 1.
a=5, b=6, c=7 எனில் கீழ்க்காணும் கோவைகளின் மதிப்பைக் காண்க:		
a += b*c c *= a + a / b b += a % 2 * c		

அட்டவணை 2.9. மதிப்பிருத்து செயற்குறிகள்

சி++ மொழியில் பயன்படுத்தப்படும் அனைத்துச் செயற்குறிகளின் முன்னுரிமையையும் நிறைவேற்றப்படும் திசைமுகத்தையும் அட்டவணை 2.10-ல் காண்க.

செயற்குறி முன்னுரிமை	**வகை**	**திசைமுகம்**
( ) [ ]		இடமிருந்து வலம் இடமிருந்து வலம்
பின்னொட்டு ++, — , முன்னொட்டு ++, — ! – தருக்க இல்லை ஒரும +, ஒரும -	கணித ஒருமம் கணித ஒருமம் தருக்க ஒருமம் கணித ஒருமம்	வலமிருந்து இடம் வலமிருந்து இடம் வலமிருந்து இடம் இடமிருந்து வலம்
* / %	கணித இருமம்	இடமிருந்து வலம்
+ -	கணித இருமம்	இடமிருந்து வலம்
< <= > >=	ஒப்பீட்டு இருமம்	இடமிருந்து வலம்
= = !=	ஒப்பீட்டு இருமம்	இடமிருந்து வலம்
&& (AND)	தருக்க இருமம்	இடமிருந்து வலம்
|| (OR)	தருக்க இருமம்	இடமிருந்து வலம்
?:	தருக்க மும்மம்	இடமிருந்து வலம்
= *= /= %= += -= <<= >>= &= ^= |=	மதிப்பிருத்தல்	வலமிருந்து இடம்

அட்டவணை 2.10 செயற்குறி முன்னுரிமை

(குறிப்பு: C++ மொழிக்கே உரிய சிறப்புச் செயற்குறிகள் அந்தந்தப் பாடங்களில் விளக்கப் படும்)

2.2.8 **நிறுத்தற் குறிகள்** (Punctuators)

நிறுத்தற் குறிகள் என்பவை குறிப்பிட்ட பணியைச் செய்யும் குறியுருக்களாகும். நிறுத்தற் குறிகளையும் அவற்றின் பயன்பாட்டையும் அட்டவணை 2.11-ல் காண்க.

நிறுத்தற் குறி	**பயன்பாடு**
;	ஒரு சி++ கட்டளையை முடித்து வைக்கிறது
//	குறிப்புரைகளுக்குப் (Comments) பயன்படுத்தப்படுகிறது.
/* */	இவற்றில் உள்ளடங்கிய உரைத்தொகுதி குறிப்புரையாகக் கருதப்படும்.
{ }	சி++ கட்டளைகளை ஒரு தொகுதியாகக் குறிக்கப் பயன்படுகிறது. ஒரு செயற்கூறின் கட்டளைத் தொகுதி இந்த அடைப்புக் குறிகளுக்குள் இருக்கும்.
[ ]	ஓர் அணியின் குறிப்பிட்ட உறுப்பைச் சுட்டுகின்ற சுட்டெண் (Index) இந்த அடைப்புக் குறிகளுக்குள் இடம்பெறும்.
‘ ’	ஒற்றைக் குறியுரு இந்த மேற்கோள் குறிகளுக்குள் தரப்படும்.
“ ”	சரம் இந்த மேற்கோள் குறிகளுக்குள் இடம்பெறும்.

அட்டவணை 2.11 நிறுத்தற் குறிகளும் அவற்றின் பயன்பாடும்

2.3 **தரவு இனங்கள்** (Data Types)

ஒரு நிரலாக்க மொழியில் கையாளப்படும் மாறிகள் ஒவ்வொன்றும் குறிப்பிட்ட வகை மதிப்புகளையே ஏற்கும். இத்தகைய தரவின் வகைகளையே ***தரவு இனங்கள்*** (Data Types) என்கிறோம். சி++ மொழியில் தரவுகளை வேறுபட்ட இனங்களாகப் பிரித்துக் காணும் திறன், சிக்கல் மிகுந்த பொருள்களை நிரல்களில் எடுத்தாளும் பணியை எளிமையாக்குகிறது. தரவுகளைப் பல்வேறு வகையினங்களாகப் பிரித்தமைப்பதற்கு இரண்டு காரணங்கள் உள்ளன:

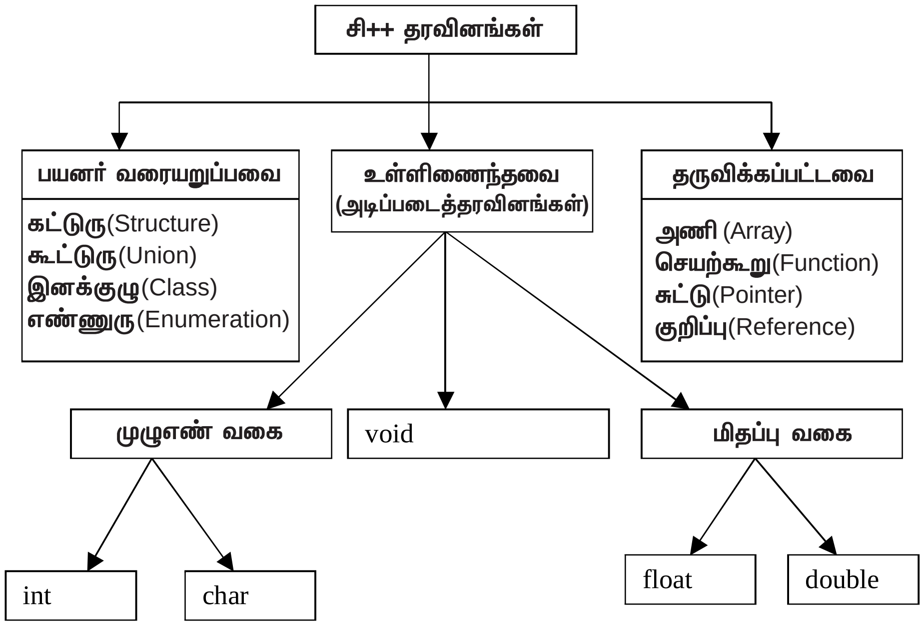
- ஒவ்வொரு தரவினத்துக்கும் நிரல்பெயர்ப்பி (Compiler) உரிய அகநிலை உருவகிப்பைப் பயன்படுத்திக் கொள்ளும்.

● நிரல்களை உருவாக்கும் நிரலர் ஒவ்வொரு தரவினத்துக்கும் ஏற்ற செயற்குறிகளைப் பயன்படுத்திக்கொள்ள முடியும்.

தரவினங்களை முப்பெரும் வகையினங்களாகப் பிரிக்கலாம்:

- பயனர் வரையறுக்கும் தரவினம் (User defined type)
- உள்ளிணைந்த தரவினம் (Built-in type)
- தருவிக்கப்பட்ட தரவினம் (Derived type)

மேற்கண்ட பரந்த வகைப்பாட்டினை உட்பிரிவுகளோடு படம் 2.2-ல் காண்க:

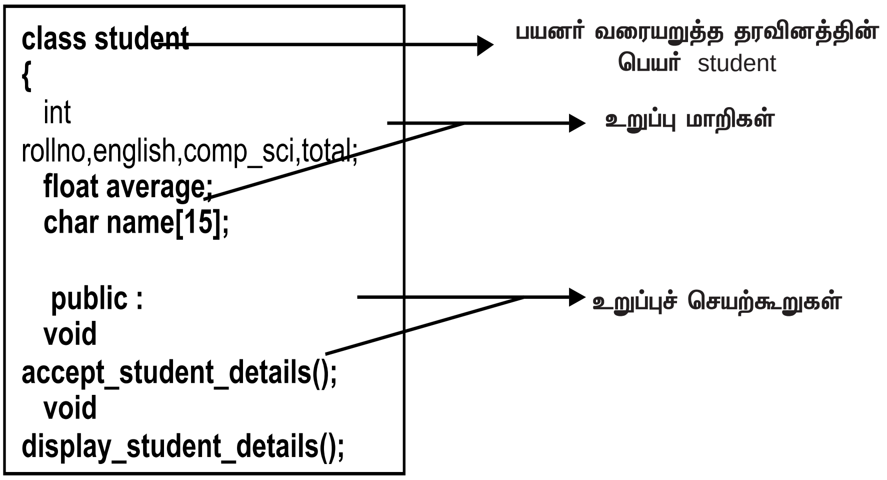


படம் 2.2 சி++ தரவினங்கள்

2.3.1 **பயனர் வரையறுக்கும் தரவினம்** (User defined data type)

நிரலர் தன்னுடைய சொந்தத் தரவினங்களையும், அவை ஏற்கின்ற மதிப்புகளையும் வரையறுத்துக்கொள்ள வகை செய்யப்பட்டுள்ளது. ஒரு நிரலின் படிப்பெளிமையை (readability) மேம்படுத்த இது உதவுகிறது.

எடுத்துக்காட்டாக, கீழேயுள்ள பயனர் வரையறுத்த தரவினத்தைக் காண்க:



படம் 2.3 பயனர் வரையறுத்த தரவினம்

பயனர் வரையறுத்த student என்னும் தரவினம் ***இனக்குழு*** (class) என்னும் தரவு வகையைச் சார்ந்தது. ***உறுப்பு மாறிகள், அவற்றோடு தொடர்புடைய செயற்கூறுகளின் வடிவில்*** ஒரு மாணவரின் பரிமாணங்களை இது வரையறுத்துள்ளது. ஒரு மாணவரின் விவரங்களைப் பெறவும், அவ்விவரங் களைத் திரையில் காட்டவும் செயற்கூறுகள் வரையறுக்கப்பட்டுள்ளன. இவ்வாறு, class student என்னும் தரவினம் தேவையான தரவுகளையும் அவற்றோடு தெடர்புடைய செயற்கூறுகளையும் மாணவர்க்கான ஒரு புதிய தரவின வடிவில் ஒன்றாகப் பிணைத்து வைத்துள்ளதன் மூலம் நிரலின் ***தரத்தையும் படிப்பெளிமையையும் மேம்படுத்தியுள்ளது.***

ஏற்கெனவே நிலவும் ஒரு தரவினத்துக்குப் பயனர் ஒரு புதிய பெயரைச் சூட்டிக்கொள்ள முடியும். ***இன வரையறை*** (Type definition) என்னும் வசதி பயனர், தரவினக் குறிப்பெயர்களை உருவாக்கிக் கொள்ள அனுமதிக்கிறது. அதற்கான கட்டளை அமைப்பு:

typedef <தரவினம்> <பயனர் வரையறுக்கும் தரவினக் குறிப்பெயர்>;

எடுத்துக்காட்டு:
typedef int ***marks;***
typedef char ***grade;***

marks, grade ஆகியவை பயனர் வரையறுத்த தரவினக் குறிப்பெயர் களாகும். பயனர்கள் int, char மாறிகளை இவ்வாறு வரையறுக்கலாம்:

marks eng_marks, math_marks;
grade eng_grade, math_grade;

பொருள் பொதிந்த தரவினக் குறிப்பெயர்களை உருவாக்கிக் கொள்ள typedef உதவுகிறது. அவை நிரலின் படிப்பெளிமையை மேம்படுத்துகின்றன.

பயனர் வரையறுக்கும் இன்னொரு தரவினம் எண்ணுருத் தரவினமாகும். பயனர்கள் குறிப்பெயர்களின் பட்டியலை ஒரு தரவினமாக வரையறுத்துக் கொள்ள இத்தரவினம் உதவுகிறது. ***int இனக் குறியீட்டு எண்வகை மாறிலி*** என்றும் இதனை அழைப்பர்.

கட்டளை அமைப்பு:

enum <தரவினக் குறிப்பெயர்> (மதிப்பு1, மதிப்பு2...... மதிப்பு n);

எடுத்துக்காட்டுகள்:

enum ***working_days*** (Monday, Tuesday, Wednesday, Thursday, Friday);
enum ***holidays*** (Sunday, Saturday);

working_days, holidays ஆகிய குறிப்பெயர்கள் பயனர் வரையறுத்த தரவினங்கள் ஆகும். (Monday, Tuesday,...) என்பது ***எண்ணுரு மாறிலிகள்*** அல்லது ***எண்வகை மாறிலிகள்*** என்று அழைக்கப்படும் மதிப்புகளைக் கொண்ட பட்டியலாகும்.

இந்த எண்ணுருத் தரவினத்தில் பயனர்கள் கீழ்க்காணுமாறு மாறிகளை அறிவிக்கலாம்:

enum <குறிப்பெயர்> மாறி1, மாறி2,.......மாறிn;

எடுத்துக்காட்டாக, working_days என்னும் தரவினத்தில் first_workingday, last_workingday என்னும் மாறிகளைக் கீழ்க்காணுமாறு அறிவிக்க முடியும்:

working_days first_workingday, last_workingday;

இந்த மாறிகள் working_days இனத்தில் வரையறுக்கப்பட்டுள்ள மதிப்புகளில் ஒன்றை மட்டுமே ஏற்கும்.

first_workingday = Monday;
last_workingday = Friday;

குறிப்பெயர் மாறிலிகள் Monday, Tuesday, Wednesday....... ஆகியவை அக நிலையில் நிரல்பெயர்ப்பியால் முன்னியல்பாக 0-வில் தொடங்கி 1, 2, 3,........... என வரிசையான int மாறிலிகளாகவே கையாளப்படுகின்றன. எனவே, மேற்கண்ட மதிப்பிருத்தல் கட்டளைகளை இவ்வாறும் எழுதலாம்.

```
first_workingday = 0;
last_workingday = 4;
```

பயனர்கள், குறிப்பெயர் மாறிலிகளுக்கு வெளிப்படையாக வேறு எண் மதிப்புகளைக் குறிப்பிட்டு int மாறிலிகளை இவ்வாறு மறுவரையறை செய்துகொள்ள முடியும்.

```
enum working_days (Monday=1, Tuesday, Wednesday, Thursday, Friday);
```

இதில் Monday என்பது 1 என்ற மதிப்பைப் பெறும். அடுத்துள்ள மாறிலிகள் 2, 3, 4.... என அடுத்தடுத்த int மாறிலி மதிப்புகளைப் பெறும்.

2.3.2 **சேமிப்பு இனம்** (Storage Class)

ஒரு மாறியின் அறிவிப்புக்கு முன்னால் ***சேமிப்பு இனம்*** என்னும் தகுதியாக்கியை (qualifier) இணைத்துக்கொள்ள முடியும். auto, static, extern, register என்னும் நான்கு வகையான சேமிப்பு இனங்கள் உள்ளன. static, register ஆகிய இன மாறிகளில் அவை அறிவிக்கப்படும்போதே 0 என்ற தொடக்க மதிப்பு தானாகவே இருத்தப்பட்டு விடுகிறது. auto இன மாறிகளில் அவ்வாறு தரவினத்தின் அடிப்படையில் ஏற்ற மதிப்புகள் தொடக்க மதிப்பாக இருத்தப்படுவதில்லை. auto மாறிகள், குப்பை (Garbage) எனப்படும் வரையறுக்கப்படாத மதிப்புகளைப் பெறுகின்றன. சேமிப்பு இனங்களுக்கான பொருளும், பொருத்தமான எடுத்துக்காட்டுகளும் அட்டவணை 2.12-ல் வழங்கப் பட்டுள்ளன:

சேமிப்பு இனம்	பொருள்	எடுத்துக்காட்டு

auto	உள்ளக மாறிகள் (local variables) . இவை அறிவிக்கப்படும் தொகுதிக்குள் மட்டுமே அறியப் படும். முன்னியல்பாகவே, உள்ளக மாறிகள் அனைத்தும் auto இனத்தைச் சார்ந்தவையே. எனவே auto என்னும் சேமிப்பு இனப்பெயர் பெரும்பாலும் குறிப்பிடப்படுவதில்லை.	`void main()`
		`{`
		`auto float ratio;`
		`auto int count;`
		`}`
		ratio, count ஆகிய மாறி கள் main()செயற்கூறுக்குள் வரையறுக்கப்பட்டுள்ளன. இவைautoஎன்னும்சேமிப்பு இனத்தைச் சேர்ந்தவை

static	ஒரு செயற்கூறினுள் அல்லது தொகுதிக்குள் வரையறுக்கப்படும் மாறிகள் அந்த செயற்கூறு அல்லது தொகுதி செயல்பட்டு முடிந்தவுடன் அழிந்து போகின்றன. அவ்வாறின்றி மாறிகள் அவை அறிவிக்கப்படும் செயற்கூறு அல்லது தொகுதி செயல் பட்டு முடிந்தபின்னும் கணிப்பொறி நினைவகத்தில் தங்கியிருக்க static என்னும் பண்புணர்த்தி(Modifier)வகை செய்கிறது.அந்த மாறிகள் கடைசியாக இருத்தப்பட்ட மதிப்புகளைத் தக்க வைத்துக் கொள்கின்றன.	`void fun()`
		`{`
		`static int x;`
		`x++;`
		`}`

extern	இந்த வகையைச் சார்ந்த மாறிகள், நடப்பு நிரலில் அனைத்து செயற்கூறு களிலும் பயன்படுத்தக்கூடிய முழுதளாவியமாறிகள்(global variables) ஆகும். இந்த மாறிகள் வேறொரு நிரலில் வரையறுக்கப்பட்டிருக்கும்.	`extern int filemode;` `extern void factorial();`

register	register என்னும் பண்புணர்த்தி, அவ்வாறு அறிவிக்கப்படும் மாறிகளை சிபீயுவின் பதிவகங்களில் (registers) இருத்திவைக்குமாறு நிரல்பெயர்ப்பிக்கு உணர்த்துகிறது. இவற்றை மிக வேகமாய் அணுக முடியும்.	`void fun()`
		`{`
		`register int i;`
		`}`

அட்டவணை 2.12 சேமிப்பு இனக்குழுக்கள்

2.3.4 உள்ளிணைந்த தரவினங்கள் (Built-in Data Types)

உள்ளிணைந்த தரவினங்கள் ***மூலத் தரவினங்கள்*** அல்லது ***அடிப்படைத் தரவினங்கள்*** என்றும் அழைக்கப்படுகின்றன. நிரல்பெயர்ப்பியில் அவை முன்வரையறுக்கப்பட்டுள்ளன. அடிப்படைத் தரவினங்களை முழுஎண் (integral), மிதப்பு(float), மதிப்பிலி(void) என மூன்று வகையாகப் பிரிக்கலாம்.

முழுஎண் வகை, int, char ஆகிய தரவினங்களை உள்ளடக்கியது. int தரவினம். 1, 2, 3...... என முழுஎண் மதிப்புகளையே ஏற்கும். பின்ன மதிப்பு களை ஏற்காது. char என்பது குறியுருத் தரவினம். எனினும் இது குறியுரு மதிப்பு, முழுஎண் மதிப்பு ஆகிய இரண்டையுமே ஏற்கும். எடுத்துக்காட்டாக, ch என்னும் மாறியின் அறிவிப்பையும், அதில் தொடக்க மதிப்பிருத்தலையும் காண்க:

char ch = ‘A’;

இதே கட்டளையை,

char ch = 65;

என்றும் அறிவிக்கலாம். (65 என்பது A என்னும் எழுத்தின் ஆஸ்க்கி மதிப் பாகும்). மேற்கண்ட இரண்டு கூற்றுகளும் ஒரே பணியையே நிறைவேற்று கின்றன. ch என்னும் மாறியில் ‘A’ என்னும் எழுத்தை இருத்துகின்றன.

மிதப்புவகைத் தரவினம் float, double ஆகிய தரவினங்களை உள்ள டக்கியது. மிதப்புவகைத் தரவினம் பின்னப் பகுதியுடன் கூடிய எண் மதிப்பு களை இருத்திவைக்கும் திறன் படைத்தவை. (மிதப்புப் புள்ளி மாறிலிகள் பற்றி ஏற்கெனவே படித்துள்ளதை நினைவு கூர்க).

மதிப்பிலித் தரவினம் (void type) இரண்டு வகையில் பயன்படுகின்றன:

- ஒரு செயற்கூறு எந்த மதிப்பையும் திருப்பி அனுப்பாது என்பதைக் குறிக்க
- பொது இனச் சுட்டினை (generic pointer) அறிவிக்க

எடுத்துக்காட்டாக, கீழேயுள்ள void.cpp, fun.cpp ஆகிய சி++ நிரல்களில் வரையறுக்கப்பட்டுள்ள செயற்கூறுகளை நோக்குக:

```
Program  void.cpp
#include<iostream.h>
#include<conio.h>
void  fun(void)
{
  int a,b;
  cin >> a >> b;
  cout << a+b;
}

void main()
{
  fun();
}
```

```
Program  fun.cpp
#include<iostream.h>
#include<conio.h>
int fun(int a, int b)
{
  return  a+b;
}
void main()
{
   int a = 5 , b = 6, sum = 0;
   sum =  fun(a,b);
   cout << sum;
}
```

void.cpp நிரலில் void cpp(void) என அறிவிக்கப்பட்டுள்ள முன்வடிவு (prototype) இச்செயற்கூறு எவ்வித மதிப்பையும் திருப்பியனுப்பாது என்பதையும், அளபுருக்களின் (parameters) வடிவில் எவ்வித மதிப்புகளையும் ஏற்றுக் கொள்ளாது என்பதையும் உணர்த்துகிறது. எனவேதான் main( ) செயற்கூறில், அழைப்புக் கூற்று fun() என்று இடம்பெற்றுள்ளது. fun.cpp நிரலில் அறிவிக்கப் பட்டுள்ள int fun(int a, int b) என்னும் முன்வடிவு fun( ) செயற்கூறு ஓர் int மதிப்பைத் திருப்பியனுப்பும் என்பதை நிரல்பெயர்ப்பிக்கு உணர்த்துகிறது. எனவேதான் main() செயற்கூறில் அழைப்புக் கூற்று sum=fun(a,b) என அமைக்கப்பட்டுள்ளது. sum என்னும் மாறி return a+b என்னும் கட்டளை மூலமாக int மதிப்பினைப் பெற்றுக் கொள்கிறது.

✓ void தரவினம், அச்செயற்கூறு எந்த மதிப்பையும் திருப்பியனுப்பாது என்பதை நிரல்பெயர்ப்பிக்கு உணர்த்துகிறது. பரந்த சூழலில் நோக்கினால், void தரவினம் எந்த மதிப்பையும் ஏற்காது என்பதை உணர்த்துகிறது.

அடிப்படைத் தரவினங்கள் பல்வேறு பண்புணர்த்திகளை (modifiers) ஏற்கின்றன. இந்தப் பண்புணர்த்திகள் அகநிலையில் தரவுகளை உருவகிப்பதில் ஆழமான விளைவுகளை ஏற்படுத்துகின்றன. signed. unsigned. long. short ஆகியவை அவற்றுள் சில பண்புணர்த்திகள் ஆகும். தரவினங்கள், நினைவகத்தில் அவை எடுத்துக் கொள்ளும் பைட் அளவு, அவற்றின் வரம்பு மதிப்புகள் ஆகியவற்றை அட்டவணை 2.13-ல் காண்க.

Type Byte		Range
char	1	-128 to 127
unsigned char	1	0 to 255
signed char	1	-128 to 127
int	2	-32768 to 32767
unsigned int, unsigned short int	2	0 to 65535
signed int,short int, signed short int	2	-32768 to 32767
long int,signed long int	4	-2147483648 to 2147483647
unsigned long int	4	0 to 4294967295
float	4	3.4e-38to3.4e+38
double	8	1.7e-308 to 1.7e+308
long double	10	3.4e-4932 to 1.1e+4932

அட்டவணை 2.13 தரவினங்களும் அவற்றின் அளவும் வரம்பு மதிப்புகளும்

2.3.4 தருவிக்கப்பட்ட தரவினங்கள் (Derived Data Types)

தருவிக்கப்பட்ட தரவினங்கள் int, float போன்ற உள்ளிணைந்த அடிப்படைத் தரவினங்கள் அல்லது பயனர் வரையறுக்கும் தரவினங்களிலிருந்து உருவாக்கப்படுபவை ஆகும். எடுத்துக்காட்டாக, அணி (Array) வகைத் தரவினத்தின் அறிவிப்பு/மதிப்பிருத்தலைக் காண்க:

```
int num_array [5] = {1,2,3,4,5};
char dayname [7] [4] = { “Sun”, “Mon”, “Tue”, “Wed”, “Thu”, “Fri”, “Sat”};
```

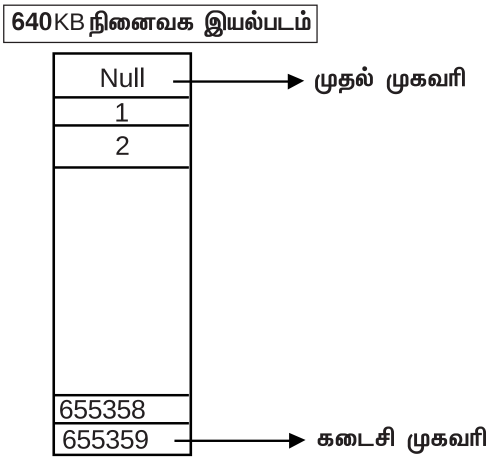
num_array 5 மதிப்புகளைக் கொண்டுள்ளது. அணியிலுள்ள ஒவ்வோர் உறுப்பும் அதன் இருப்பிடச் சுட்டெண் வழியாக அணுகப்படுகிறது. இருப்பிட எண் 0 விலிருந்து தொடங்குகிறது. num_array[0] என்னும் உறுப்பில் 1 இருக்கும். num_array [4] என்னும் உறுப்பில் 5 இருக்கும்.

dayname[0], dayname[5], dayname[3][2] ஆகிய உறுப்புகளில் உள்ள மதிப்புகள் என்னென்ன என்பதைக் கூற முடியுமா?

சுட்டுகள் (Pointers)

சுட்டு என்பது ஒரு நினைவக முகவரியை ஏற்கும் மாறி ஆகும். ஒரு மாறியின் நினைவக இருப்பிடத்தை நேரடியாக அணுகுவதற்கு சுட்டுகள் உதவுகின்றன. கணிப்பொறி நினைவகத்தில் உள்ள ஒவ்வொரு பைட்டும் ஒரு முகவரியைக் கொண்டுள்ளன. முகவரி என்பது ஓர் எண்ணாகும். நமது வீட்டுக் கதவெண்களைப் போல. முகவரி எண் என்பது NULL என்பதில் தொடங்கி 1, 2, 3 .... என வரிசையாக அமையும்.

எடுத்துக்காட்டாக, 640Kb அளவுள்ள நினைவகம், NULL இல் தொடங்கி 6,55,356 வரை முகவரிகளைக் கொண்டிருக்கும். படம் 2.4 காண்க.



படம் 2.4 640KB நினைவக இயல்படம்

ஒரு நிரலை எந்திர மொழிக்கு மாற்றும்போது, நிரல்பெயர்ப்பி, மாறிகளுக்குரிய நினைவக இருப்பிடங்களை ஒதுக்கீடு செய்கிறது. மாறி சார்ந்துள்ள தரவினத்துக்கு ஏற்ப அதற்கு ஒதுக்கப்படும் நினைவகத்தின் இட அளவு அமைகிறது.

எடுத்துக்காட்டாக, கீழேயுள்ள அறிவிப்புகளைக் காண்க:

char c; int i; float f;

char c - 1 பைட்

int i - 2 பைட்டுகள்

float f – 4 பைட்டுகள்

ஒவ்வொரு மாறியும் அதன் முகவரியைக் கொண்டே குறிப்பிடப்படுகிறது. நமது எடுத்துக் காட்டில், c, i, f ஆகிய மாறிகளின் நினைவக இருப்பிடங்கள் முறையே 1, 2, 4 பைட்டுகளைக் கொண்டிருக்கும். இவற்றின் முகவரிகள் பதினறும எண் முறையில் குறிக்கப்படுகின்றன.

சுட்டுத் தரவினத்தைக் கையாள்பவர், ***முகவரி சுட்டல் (&), மதிப்பு சுட்டல் (*)*** ஆகிய செயற்குறிகளைப் பற்றி அறிந்துகொள்ள வேண்டும்.

‘&’ செயற்குறி:

int num1=10;

என்ற கட்டளையைத் தந்தால், சி++ நிரல்பெயர்ப்பி (C++ Compiler) கீழ்க்காணும் செயல்பாடுகளை செய்து முடிக்கின்றது:

1. நினைவகத்தில் ஓர் int மதிப்பை இருத்திவைக்க இடம் ஒதுக்கீடு செய்கிறது
2. அந்த நினைவக இடத்துக்கு num1 என்ற பெயரைச் சூட்டுகிறது.
3. நினைவகத்தில் அந்த இடத்தில் 10 என்னும் மதிப்பை இருத்துகிறது.

num1 ⟶ மாறியின் பெயர் (இருப்பிடத்தின் பெயர்)

10	தரவு (இருப்பிடத்தில் இருத்தப்பட்டுள்ள மதிப்பு)

0X8f90f2 ⟶ num1 என்ற மாறியின் நினைவக முகவரி

```
// Program - 2.1
// to demonstrate use of & operator
#include<iostream.h>
#include<conio.h>
void main()
{     clrscr();
      int i = 10;

cout << “\n Address of the variable... “ <<&i;
cout << “\nValue stored in the variable ..” << i;

      getch();
}
```

கீழேயுள்ள கட்டளைத் தொகுதியைக் காண்க:

```
int *x , num1;
num1=10;
x=&num1;
cout<<*x;
```

குறிப்பு: நட்சத்திரக் குறி (*)

1. ஒரு சுட்டு இன மாறியை அறிவிக்கப் பயன்படுத்தப்பட்டுள்ளது.
2. நினைவகத்தில் இருத்தப்பட்டுள்ள மதிப்பைத் திரையில் காட்டப் பயன்பட்டுள்ளது (மதிப்பு சுட்டல் செயற்குறி).
3. இது ஓர் ஒருமச் செயற்குறி (Unary Operator) ஆகும்.

2.4 மாறிகள் (Variables)

குறிப்பிட்ட வரம்பெல்லைக்குள் ஒரு மதிப்பினை ஏற்கின்ற ஒரு தரவுப் புலத்தின் பெயரே மாறி எனப்படுகிறது. எடுத்துக்காட்டாக, கீழேயுள்ள கூற்றுகளைக் காண்க:

```
int num;
num = 5;
```

int num ; என்னும் கூற்று "n என்பது int இனத்தைச் சேர்ந்த மாறி" என்ற பொருளை உணர்த்துகிறது.

```
int num;
num  =  5;
```

என்னும் மதிப்பிருத்து கூற்று, "5 என்னும் மதிப்பு num என்னும் மாறியில் இருத்தப்படுகிறது'' என்ற பொருளை உணர்த்துகிறது.

> தரவுகள் இருத்தப்படும் நினைவக இருப்பிடங்களுக்குப் பயனர் சூட்டும் பெயர்களே ***மாறிகள்*** ஆகும்.

மாறியின் பெயர்கள் எழுத்துகள், எண்கள், அடிக்கீறு (underscore) ஆகியவற்றைக் கொண்டிருக்கலாம். பெயர்கள் எழுத்தில் அல்லது அடிக்கீறில் தொடங்க வேண்டும். எனினும், அடிக்கீறில் தொடங்கும் பெயர்கள் அகநிலை முறைமை மாறிகளுக்கான (internal system variables) ஒதுக்கப்பட்டுள்ளன என்பதை அறிக. பெயர்கள் எழுத்து வடிவ உணர்வு (Case Sensitive) உள்ளவை. அதாவது. சிறிய எழுத்து, பெரிய எழுத்தில் அமையும் பெயர்கள் வேறுபடுத்தி அறியப்படுகின்றன. அட்டவணை 2.14 -ஐ நிறைவு செய்க.

மாறி	**சரியானதா/ பிழையானதா?**	**பிழையானது என்பதற்கான காரணம்**
A_b	சரியானது	
1a_b	பிழையானது	மாறியின் பெயர் எழுத்து அல்லது அடிக்கீறில்தான் தொடங்க வேண்டும்.
_Test		
Balance$		
#one		
Include		

அட்டவணை 2.14 மாறிப் பெயர்களின் ஏற்புத் தகைமை

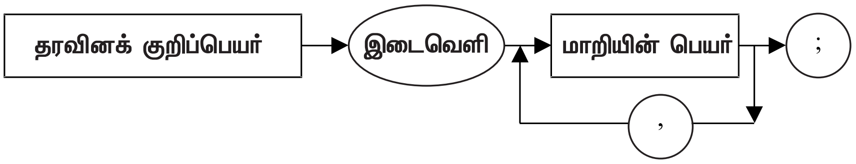
2.4.1 மாறிகளின் அறிவிப்பு

மதிப்புகளை இருத்திக்கொள்ள மாறிகளுக்கு நினைவகம் ஒதுக்கப் படுகிறது. மாறி சார்ந்துள்ள தரவினத்தின் அடிப்படையில் நிரல்பெயர்ப்பி நினைவகத்தை ஒதுக்குகிறது. எனவே, மாறிகளைப் பயன்படுத்துவதற்கு முன்பாக அவற்றைத் தரவினம் குறிப்பிட்டு அறிவிக்க வேண்டும்.

எடுத்துக்காட்டு

```
int q;
float f1, f2;
char name [10], choice;
```

அறிவிப்புக் கூற்றின் அமைப்பு:



int side, float hypotenuse, area;

என்னும் அறிவிப்பை எடுத்துக் கொள்வோம். இது தவறான அறிவிப்பாகும். காரணம், நிரல்பெயர்ப்பி இக்கூற்றினை இவ்வாறு புரிந்துகொள்ளும்:

- side, float, hypotenuse, area ஆகியவை int இன மாறிகளாகக் கருதப்படும். எனவே, “”float என்னும் சொல்லுக்குப் பிறகு காற்புள்ளி இடம்பெற வேண்டும்'' என்னும் பிழைசுட்டும் செய்தியைத் தரும்.
- side என்னும் int இன மாறியையும், hypotenuse, area என்னும் float இன மாறிகளையும் அறிவிப்பதே நமது நோக்கம். எனவே மேற்கண்ட அறிவிப்புக் கூற்றினை இவ்வாறு அமைக்க வேண்டும்:

```
int side ;
float hypotenuse , area ;
```

```
int side ; float hypotenuse,area ;
```

✓ ஒரே இனத்தைச் சார்ந்த ஒன்றுக்கு மேற்பட்ட மாறிகளை ஒரே அறிவிப்புக் கூற்றில் குறிப்பிட முடியும். ஆனால் காற்புள்ளியால் பிரிக்கப்பட்டிருக்க வேண்டும்.

தரவினங்களை அறிவிப்பதற்கென char, int, double, float, void, short, signed, long, unsigned ஆகிய ஒன்பது சொற்கள் பயன்படுத்தப்படுகின்றன.

long, short, signed, unsigned ஆகியவை, உள்ளிணைக்கப்பட்ட தரவு இனங்களின் பண்புகளை மாற்றியமைக்கப் பயன்படும் பண்புணர்த்திகள் (modifiers) அல்லது தகுதியாக்கிகள் (qualifiers) எனப்படுகின்றன. ஆனால் void இனத்துக்கு இவற்றைப் பயன்படுத்த முடியாது.

அகநிலையில் 15 என்னும் int மதிப்பு 00000000 00001111 என்று கையாளப்படுகிறது. int மதிப்புகள் இரும எண்முறையில் 16 பிட் வடிவமைப்பில் இருத்தி வைக்கப்படுகின்றன. வலப்புறம் தொடங்கி ஒவ்வொரு பிட்டாக இடப்புறம் அமைந்து மொத்தம் 15 பிட்டுகள் மதிப்பை இருத்தப் பயன்படுத்தப்படுகின்றன. எனவே ஓர் int மாறியில் இருத்தப்படும் பெரும மதிப்பு +32767. சிறும மதிப்பு -32768. (2^{15} = 32768. நேர்ம மதிப்பு 0 முதல் + 32767 வரை. எதிர்ம மதிப்பு -1 முதல் -32768 வரை) 16-வது பிட், உச்ச மதிப்பு பிட் (Most Sifgnificant Bit) அல்லது குறி பிட் (Sign Bit) எனப்படுகிறது. இது, அவ்வெண் நேர்மம் (Positive) அல்லது எதிர்மம் (Negative) என்பதைக் குறிக்கிறது. 16-வது பிட் 1 எனில் அவ்வெண் எதிர்ம எண், 0 எனில் நேர்ம எண் ஆகும்.

ஓர் int இன மாறியின் அறிவிப்பில் unsigned என்னும் பண்புணர்த்தியைப் பயன்படுத்தினால் அம்மாறியில் நேர்ம மதிப்புகளை மட்டுமே இருத்த முடியும். குறி பிட்டும் தரவு மதிப்புக்காகவே பயன்படுத்திக்கொள்ளப்படும். எனவே, மதிப்பின் வரம்பெல்லை 2^{16} வரை இருக்க முடியும். அதாவது 0 முதல் 65535 வரை இருத்தி வைக்கலாம்.

> ✓ பண்புணர்த்தி (modifier) ஓர் அடிப்படைத் தரவினத்தின் பண்பினை மாற்றியமைத்துப் புதிய தரவினத்தை உருவாக்குகிறது.

பண்புணர்த்திகளின் விளைவுகள்:

- **unsigned** என்னும் பண்புணர்த்தி, நேர்மம்/எதிர்மம் என்பதைக் குறிக்கும் குறி பிட்டையும் மதிப்புணர்த்தப் பயன்படுத்திக் கொள்வதால் மாறி ஏற்கும் மதிப்பின் வரம்பெல்லை மாற்றியமைக்கப்படுகிறது.
- **long** என்றும் பண்புணர்த்தி குறிப்பிட்ட தரவினத்தின் பைட்டுகளை அதிகரித்து, மதிப்பின் வரம்பெல்லையை நீட்டிக்கிறது.

அடிப்படைத் தரவினத்தில் மாறிகள் அறிவிக்கப்படும்போது, முன் னொட்டாகப் பண்புணர்த்திகள் இடம்பெற வேண்டும். எடுத்துக்காட்டாக,

unsigned int registration_number;
unsigned long int index;
signed char c;

> ✓ மாறிகளை அறிவிக்கும்போது, பண்புணர்திகளைத் தரவினப் பெயருக்கு முன்னொட்டாகத் தர வேண்டும்.

const என்னும் தகுதியாக்கி (qualifier) நிரலின் இயக்க நேரத்தில் ஒரு மாறியின் மதிப்பு மாற்றப்பட முடியாது என்பதை உணர்த்துகிறது. const மாறியின் மதிப்பை மாற்ற முயன்றால் நிரல்பெயர்ப்பி பிழை சுட்டும் செய்தியைத் தரும். const தகுதியாக்கி, பிற பண்புணர்த்திகளைப் போன்றே மாறிகள் அறிவிக்கப்படும்போது தரவினப் பெயருக்கு முன்னொட்டாக இடம் பெறுகிறது.

எடுத்துக்காட்டு:

const float pi = 3.14;

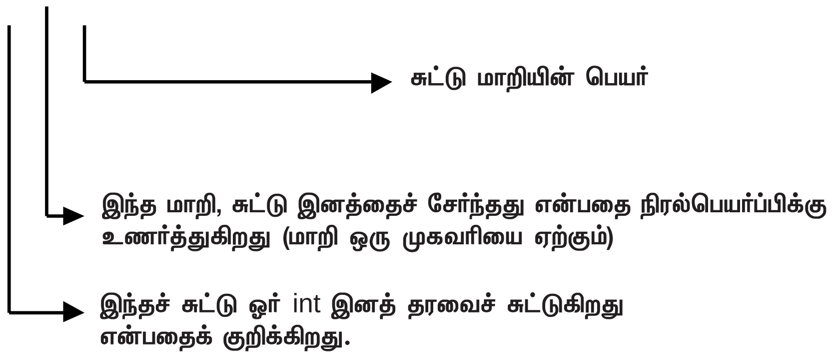
தகுதியாக்கிகளும் பண்புணர்த்திகளும் பயன்படுத்தும்போது பிட் எண்ணிக்கையும் மதிப்பின் வரம்பெல்லையும் மாற்றியமைக்கப்படும் தரவினங்களை அட்டவணை 2.15-ல் காண்க:

Data Types		
Type	Length	Range
unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
enum	16 bits	-32,768 to 32,767
unsigned int	16 bits	0 to 65,535
short int	16 bits	-32,768 to 32,767
int	16 bits	-32,768 to 32,767
unsigned long	32 bits	0 to 4,294,967,295
long	32 bits	-2,147,483,648 to 2,147,483,647
float	32 bits	3.4×10^{-38} to 3.4×10^{38}
double	64 bits	1.7×10^{-308} to 1.7×10^{308}
long double	80 bits	3.4×10^{-4932} to 1.1×10^{4932}

அட்டவணை 2.15 பண்புணர்த்திகளுடன் தரவினங்கள்

சுட்டு மாறிகளை அறிவித்தல்

int * iptr;



int * iptr என்னும் அறிவிப்புக் கூற்றுக்கு, iptr என்பது int இனத்தைச் சுட்டும் ஒரு சுட்டு மாறி (pointer variable) என்று பொருள்கொள்ள வேண்டும். int இன மதிப்புகள் இருத்திவைக்கப்பட்டுள்ள ஒரு முகவரியை மட்டும் iptr மாறி ஏற்கும்.

சுட்டு மாறியின் அறிவிப்புகளுக்கான எடுத்துக்காட்டுகள்:

char * cptr float * fptr	char இனத்தைச் சுட்டும் ஒரு சுட்டின் அறிவிப்பு float இனத்துக்கான ஒரு சுட்டு
void * vptr	எந்தத் தரவினத்தையும் சுட்டுகின்ற ஒரு சுட்டு. ஒரு பொதுநிலைச் சுட்டு இவ்வாறு அறிவிக்கப்படுகிறது.
const int *ptr	ஒரு மாறிலி int மதிப்புக்கான சுட்டு ptr. (ptr சுட்டும் நினைவக முகவரியிலுள்ள int மதிப்பை மாற்றியமைக்க முடியாது)
char * const cp	cp என்பது ஒரு மாறிலிச் சுட்டு. cp-ல் இருத்தப் பட்டுள்ள முகவரியை மாற்றியமைக்க முடியாது.

அட்டவணை 2.16 சுட்டு மாறிகளுக்கு எடுத்துக்காட்டுகள்

2.4.2 மாறிகளைத் தொடங்கிவைத்தல் (Initialization of Variables)

மாறிகளை அறிவிக்கும்போதே அதில் குறிப்பிட்ட மதிப்பு இருத்தப் படுவதைத் ***தொடக்க மதிப்பிருத்தல்*** என்கிறோம். மாறிகளில் தொடக்க மதிப்பிருத்தலையே ***தொடங்கிவைத்தல்*** (Initialization) என்கிறோம்.

```
int num = 10;
int fun(5);
```

முதலாவது கூற்றில் num என்னும் மாறியில் 10 என்னும் தொடக்க மதிப்பிருத்தப்பட்டுள்ளது. இரண்டாவது கூற்றில் ஓர் ஆக்கி (constructor) மூலம் 5 என்னும் மதிப்பு fun என்னும் மாறியில் இருத்தப்பட்டுள்ளது.

உள்ளுறை இனமாற்றங்கள் (Implicit Conversions)

ஒரு கோவையில் இடம்பெறும் தரவுகளின் இனம் நிரல்பெயர்ப்பியால் மாற்றப்படுவது உள்ளுறை இனமாற்றம் எனப்படுகிறது. எடுத்துக்காட்டாக, கீழேயுள்ள கட்டளைத் தொகுதியைக் காண்க:

```
float f = 7.6;
int x = f;
```

இப்போது, x என்னும் மாறியில் 7 என்னும் மதிப்பு இருக்கும். float இனம் int இனமாக மாற்றப்பட்டுவிட்டது. நிரல்பெயர்ப்பி இந்த இனமாற்றத்தைத் தானாகவே செய்கிறது.

உள்ளுறை இனமாற்றத்துக்கான விதிமுறைகள்:

ஒரு செயற்குறியால் இணைக்கப்பட்டுள்ள இரண்டு செயலேற்பிகளையும் ஒரு செயற்குறியையும் கொண்ட ஒரு கோவையை எடுத்துக்கொள்ளுங்கள். அதில் கீழேயுள்ளவாறு இனமாற்றங்கள் நிகழும்:

1. ஒரு செயலேற்பி long double இனம் எனில் இன்னொரு செயலேற்பியின் மதிப்பும் long double இனமாக மாற்றப்படும்.
2. ஒரு செயலேற்பி double இனம் எனில் இன்னொரு மதிப்பும் double இனமாக மாற்றப்படும்.
3. ஒரு செயலேற்பி float இனம் எனில் இன்னொரு மதிப்பும் float இனமாக மாற்றப்படும்
4. ஒரு செயலேற்பி unsigned long int இனம் எனில் இன்னொரு மதிப்பும் unsigned long int இனமாக மாற்றப்படும்.
5. ஒரு செயலேற்பி long int இனம் எனில் இன்னொரு மதிப்பும் long int இனமாக மாற்றப்படும்.
6. ஒரு செயலேற்பி unsigned int இனம் எனில் இன்னொரு மதிப்பும் unsigned int இனமாக மாற்றப்படும்.

```
// demonstrating implicit type conversions
// Program - 2.2
# include <iostream.h>
# include <conio.h>
# include <iomanip.h>

void main()
{
    clrscr();

    int i;
    float f;
    double d;
    long double ld;
    unsigned int ui;
```

```
  unsigned long int uli;
  i = -5;
  f = 2;
  d = 3;
  ld = 3;
  ui = 6;
  uli = 4;
  cout <<"\nSizeof long double..”<<sizeof(ld*d)<<'\t'<<ld*d;
  cout << “\nSizeof double...” << sizeof(d*f)<<'\t'<<d*f;
  cout << “\nSizeof float...” << sizeof(f * i)<<'\t' << f*i;
  cout << “\nSizeof unsigned long int ...”
           << sizeof(uli* f)<<'\t'<< uli * f;
  cout << “\nSizeof unsigned int...” << sizeof(ui * i)
         <<'\t'<< ui * i;
  getch();
}
```

குறிப்பு: sizeof என்பது ஒரு செயற்குறி. தரப்படும் ஒரு கோவை அல்லது தரவினத்தின் அளவினை (அத்தரவின் மதிப்பை நினைவகத்தில் இருத்தத் தேவைப்படும் இடத்தை) பைட்டுகளில் விடையாகத் தரும்.

```
Output displayed by the above program:
Sizeof long double       ...10  9
Sizeof double            ...8   6
Sizeof float             ...4   -10
Sizeof unsigned long int ...4   8
Sizeof unsigned int      ...2   65506
```

Program 2.2 -ன் அடிப்படையில் அட்டவணை 2.17 -ஐ நிறைவு செய்க. காரணம் என்னும் நெடுக்கையில் முதல் மதிப்புக்குத் தரப்பட்டுள்ள விளக்கத்தைப் போல விடைகளை எழுதுக.

வரிசை எண்	விடையின் பைட் அளவு	கோவை	காரணம்
1.	10	ld*d	ld என்னும் மாறி long double என்பதால் கோவையிலிருந்து பெறப்படும் மதிப்பும் long double இனமாக இருக்கும்.
2.	8	d*f	———————————————— ——————————------ கோவையிலிருந்து பெறப்படும் மதிப்பும் double இனமாக இருக்கும்.
3.	4	f*l	
4.	4	uli*f	
5.	2	ui*i	

அட்டவணை 2.17 Program 2.2**–ன் அடிப்படையிலான பயிற்சி**

சுட்டு மாறிகளைத் தொடங்கிவைத்தல்

சுட்டுமாறிகள் பிற மாறிகளின் முகவரியை இருத்திவைக்கின்றன. ஆனால், ஒரு சுட்டு மாறி எந்த இனத் தரவினைச் சுட்டுவதற்காக உருவாக்கப்பட்டதோ, அதே இன மாறியின் முகவரியையே ஏற்றுக் கொள்ளும். எடுத்துக்காட்டாக,

```
int *iptr, num1;
num1 = 10;
iptr = &num1; // சுட்டு மாறியில் தொடக்க
                        மதிப்பிருத்தல்
```

கீழே காணும் தொடக்க மதிப்பிருத்தல் பிழையானதாகும்:

```
int * iptr;
float num1 = 10.5;
iptr = &num1;  // int இனத்தைச் சுட்டும் iptr என்னும் சுட்டு மாறியில்
                  float இன மாறியின் முகவரியை இருத்த முயல்வதால்
                  பிழைசுட்டும் செய்தி கிடைக்கும்.
```

சுட்டு மாறிகள் அவை சுட்டும் மாறியின் தரவினத்தோடு ஒத்திருப்பது கட்டாயமாகும்.

இனமாற்றம் (Type Cast):

ஒரு மாறியில் இருத்திவைக்கப்பட்டுள்ள மதிப்பின் தரவினத்தை மாற்றிப் பயன்படுத்தும் செயலாக்கம் ***இனமாற்றம்*** என்று அறியப்படுகிறுது (float)7 என்னும் கூற்று 7 என்னும் முழுஎண் மதிப்பை float இனமாக மாற்றுகிறது. ஒரு மாறி அல்லது மதிப்பின் முன்னொட்டாகத் தேவையான தரவினத்தைக் குறிப்பிடுவதன் மூலம் இனமாற்றம் எய்தப்படுகிறது. கட்டளை அமைப்பு:

(<தரவு இனம்>) <மாறி/மதிப்பு>
அல்லது,
<தரவு இனம்> (மாறி/மதிப்பு)

இனமாற்றம் என்பது, உள்ளிணைந்த அடிப்படைத் தரவினங்களுக்கு மட்டுமே உரியது.

x = 8% 7.7;

என்னும் கட்டளை அமைப்பு நிரல்பெயர்ப்பின் (compilation) போது ஒரு பிழைசுட்டும் செய்தியைத் தரும். காரணம், % என்னும் வகுமீதி செயற்குறி int தரவின மதிப்புகளின் மேல்தான் செயல்படும். மேற்கண்ட பிழையான கட்டளையை,

x = 8% (int) 7.7;

எனத் திருத்தி அமைக்கலாம். float மாறிலி 7.7 இனமாற்றத்தின் மூலம் int மாறிலியாக மாற்றப்பட்டுள்ளது. அட்டவணை 2.18 -ஐ நிறைவு செய்க.

int x; x =7 / 3;	x-ல் இருத்தி வைக்கப்பட்டுள்ள மதிப்பு யாது?
float x; x = 7 / 3;	x-ல் இருத்தி வைக்கப்பட்டுள்ள மதிப்பு யாது?
float x; x = 7.0 / 3.0;	x-ல் இருத்தி வைக்கப்பட்டுள்ள மதிப்பு யாது?
float x; x = (float) 7 / 3;	x-ல் இருத்தி வைக்கப்பட்டுள்ள மதிப்பு யாது?

float x;
int a = 7 , b = 3;
x = a/b;
	x-ல் இருத்தி வைக்கப்பட்டுள்ள மதிப்பு யாது?

float x;
int a = 7 , b = 3;
x = a/ (float) b;
	x-ல் இருத்தி வைக்கப்பட்டுள்ள மதிப்பு யாது?

அட்டவணை 2.18 x–ன் மதிப்பைக் கண்டறிக

பயிற்சி வினாக்கள்

1. கீழ்க்காணும் கோவைகளில் மதிப்பிடல் வரிசையைத் தீர்மானிக்கவும்:

    i. a + pow(b, c) * 2
    ii. a || b && c
    iii. a<b && c || d > a
    iv. (c>=50) || (!flag) && (b+5 == 70)
    v. (a+b) / (a-b)
    vi. (b*b) – 4 * a * c

2. கீழ்க்காணும் நிரல்களில் பிழைகளைச் சுட்டிக் காட்டுக:

a.
```
# include <iostream.h>
void main()
{
      float f = 10.0;
      x = 50;
      cout << x << f;
}
```

b.
```
# include <iostream.h>
void main()
{
      float f = 10.0;
      x = 50;
      cout << x << f;
}
```

c.
```
# include <iostream.h>
void main()
{
      int x,y,k,l;
      x =  y + k——l;
      cout << x;
}
```

3. இந்த நிரல்களின் வெளியீடு என்னவாக இருக்கும்?

```
a.
# include <iostream.h>
# include <conio.h>

void main()
{
  int i=20;
  cout << i << i++ << ++i;
  getch();
}
```

```
b.
# include <iostream.h>
# include <conio.h>
void main()
{
  clrscr();
  int i = 1, a= 3;
  i = a++;
  cout << i;
  getch();
}
```

```
c.
# include <iostream.h>
# include <conio.h>

void main()
{
  clrscr();
  int i = 3,x;
  x = i ? i++ : ++i;
  cout << x;
  getch();
}
```

```
d.
# include <iostream.h>
# include <conio.h>
void main()
{
  int z,x = 3, y = 2;
  z = --x + y++;
  cout << z;
  getch();
}
```

```
e.
# include <iostream.h>
# include <conio.h>

void main()
{
  clrscr();
  char ch = 'a';
  ch = (ch == 'b') ? ch :' b';
  cout << ch;
  getch();
}
```

4. கீழ்க்காணும் சி++ மொழிக் கோவைகளை மதிப்பிடுக:

a=5, b=3, d=1.5, int c, float f எனக் கொள்க.

a. f = a+b/a;

b. c = d * a+b;

c. x = a++ * d + a;

d. y = a – b++ * —b;

e. (x >= y) || (!(z==y) && (z < x)) எனில்,

- ✓ x = 10, y = 5, z = 11 (அனைத்தும் முழு எண்கள்)
- ✓ x = 10, y = 10, z = 10
- ✓ x = 9, y = 10, z = 2

5. கீழ்க்காணும் நிபந்தனைகளை, நிபந்தனைச் செயற்குறியைப் பயன்படுத்தி சி++ கோவையாக எழுதிக் காட்டுக:

- ✓ x = 30 எனில் f = 0.5 இல்லையேல் f = 5
- ✓ x> = 60 எனில் fn = 0.9 இல்லையேல் .7

6. சுட்டு மாறிகள் என்றால் என்ன?

7. ஒரு குறியுருத் தரவின மாறியின் முகவரியை இருத்தி வைக்கும் name என்னும் சுட்டு மாறியை அறிவிக்கும் கூற்றினை எழுதிக் காட்டுக.

பாடம் 3

அடிப்படைக் கூற்றுகள்
(Basic Statements)

சி++ மொழியிலுள்ள அடிப்படைக் கூற்றுகள் (Basic Statements) வில்லை களைப் (tokens) பயன்படுத்தி உருவாக்கப்படுகின்றன. சி++ மொழிக் கூற்று களைப் பல்வேறு வகையாகப் பிரிக்கலாம்:

- உள்ளீடு/ வெளியீடு (Input / Output)
- அறிவிப்பு (Declaration)
- மதிப்பிருத்தல் (Assignment)
- கட்டுப்பாட்டுக் கட்டமைப்புகள் (Control Structures)
- செயற்கூறு அழைப்பு (Function Call)
- பொருளின் செய்தி (Object Message)
- திருப்பியனுப்பல் (Return)

3.1 உள்ளீட்டு / வெளியீட்டுக் கூற்றுகள்

தரவுகளைப் பெற்றுக்கொண்டு அவற்றைச் செயற்படுத்தித் தகவலாக வெளியிடுவதே எந்தவொரு கணிப்பொறி நிரலின் இன்றியமையாத செயல்பாடாகும். மாறிகளில் தரவுகளை இருத்தி வைக்க இரண்டு வழி முறைகள் உள்ளன. அவற்றுள் ஒன்று, மதிப்பிருத்து கூற்று மூலமாகச் செய்யப் படுவது. இதனை முந்தைய பாடத்தில் ஏற்கெனவே படித்துள்ளோம். இன்னொரு வழிமுறை, நிரலின் இயக்க நேரத்தில் உள்ளீடாகப் பெற்றுக் கொள்வது. நிரல் இயங்கிக் கொண்டிருக்கும்போது, விசைப்பலகை வழியாகத் தரவுகளை உள்ளீடாகப் பெற **cin** என்னும் ***பொருள்*** (சி-இன் என உச்சரிக்க வேண்டும்) பயன்படுகிறது. cin என்பது சி++ மொழியில் முன்வரையறுக்கப் பட்டுள்ள ஒரு ***பொருள்*** (Object). இது ***அடிப்படை உள்ளீட்டுத் தாரையை*** (Standard Input Stream) உருவாகிக்கிறது. உள்ளீட்டுத் தாரை என்பது, அடிப்படை உள்ளீட்டுச் சாதனமான விசைப் பலகையிலிருந்து பெறப்படும் தரவுகளைக் குறிக்கிறது. cin வேறுவகை உள்ளீட்டு மூலங்களிலிருந்தும் தரவுகளைப் படிக்கும். இதைப் பற்றிப் பிறகு பார்ப்போம். cin என்னும் பொருளைப் பற்றிய அறிவிப்புகள் istream.h என்னும் தலைப்புக் கோப்பில் (Header file) தரப் பட்டுள்ளன. அடிப்படையான உள்ளீட்டு/வெளியீட்டுச் செயல்பாடுகள் istream.h, ostream.h ஆகிய தலைப்புக் கோப்புகளில் வரையறுக்கப்பட்டுள்ள சில அறிவிப்புகளின் மூலம் நிறைவேற்றப்படுகின்றன. istream, ostream ஆகிய இரண்டின் பண்புக் கூறுகளையும் iostream.h என்னும் கோப்பு தன்னகத்தே கொண்டுள்ளது.

- தலைப்புக் கோப்பு, முன்வரையறுக்கப்பட்ட செயல்கூறுகளின் அனைத்து அடிப்படை அறிவிப்புகளையும் வரையறைகளையும் கொண்டுள்ளது.
- முன்-செயலி நெறியுறுத்தத்தைப் பயன்படுத்தி, ஒரு தலைப்புக் கோப்பினை நிரலில் இணைத்துக்கொள்ள முடியும்.
- முன்-செயலி நெறியுறுத்தம் # என்ற குறியுடன் தொடங்கும். இது, தேவையான பணியைச் செய்யுமாறு நிரல்பெயர்ப்பிக்கு உணர்த்துகிறது.
- வழக்கமாய் ஒரு முன்செயலி நெறியுறுத்தம் #include<iostream.h> என்று அமையும். இக்கூற்று, iostream.h என்னும் தலைப்புக் கோப்பினை நிரலில் சேர்த்துக் கொள்ளுமாறு நிரல்பெயர்ப்பிக்கு உணர்த்துகிறது. cin/cout ஆகிய பொருள்களைப் பயன்படுத்திக் கொள்ள வேண்டுமெனில், iostream.h என்னும் கோப்பினை நிரலில் இணைத்துக்கொள்ள வேண்டும்.
- வேறுசில தலைப்புக் கோப்புகள்: iomanip.h, stdio.h, ctype.h, math.h, fstream.h மற்றும் பிற.

>> என்னும் செயற்குறி ***தரவு ஈர்ப்பு*** (extraction) அல்லது ***தரவு பெறும்*** (get from) செயற்குறி ஆகும். இச்செயற்குறி, தனக்கு இடப்புறமுள்ள தாரைப் பொருளிலிருந்து (stream object) தரவு மதிப்புகளை எடுத்து, வலப்புறமுள்ள மாறியில் இருத்திவைக்கும். எடுத்துக்காட்டாக, கீழேயுள்ள கட்டளைகளைக் காண்க:

```
float temperature;
cin >> temperature;
```

தரவு ஈர்ப்புச் செயற்குறி (>>), உள்ளீட்டுத் தாரைப் பொருளான cin-லிருந்து தரவினை ஈர்த்தெடுத்து, வலப்புறமுள்ள temperature என்னும் மாறியில் இருத்தி வைக்கிறது. தரவு ஈர்ப்புச் செயற்குறியை அடுத்தடுத்து அமைத்து, ஒன்றுக்கு மேற்பட்ட தரவு மதிப்புகளை உள்ளீட்டுத் தாரையிலிருந்து பெற்று, உரிய மாறிகளில் இருத்திவைக்க முடியும். எடுத்துக் காட்டாக, temperature, humidity ஆகிய இரு மாறிகளுக்குத் தரவுகளை உள்ளீடாகப் பெறுவதற்கு,

```
cin >> temperature >> humidity;
```

எனக் கட்டளை அமைக்க வேண்டும்.

cout என்பது (சி-அவுட் என உச்சரிக்க வேண்டும்) அடிப்படை வெளி யீட்டுத் தாரைக்கென முன்வரையறுக்கப்பட்டுள்ள பொருள் (Object) ஆகும். அடிப்படை வெளியீட்டுத் தாரை திரைக் காட்சியாகப் பாய்கிறது. வேறு பல

வெளியீட்டுச் சாதனங்களுக்கும் இத்தாரையைத் திசைதிருப்ப (redirect) முடியும். << என்னும் செயற்குறி ***தரவு விடுப்பு*** (insertion) அல்லது ***தரவு தரும்*** (put to) செயற்குறி ஆகும். இச்செயற்குறி தனக்கு வலப்பக்கம் உள்ள ஒரு மாறியின் மதிப்பை இடப்பக்கமுள்ள பொருளில் இருத்தி வைக்கும். எடுத்துக்காட்டாக, கீழேயுள்ள கட்டளைகளை நோக்குங்கள்:

```
int marks = 85;
cout << marks;
cout << "\n Marks   obtained is : " << marks;
```

marks என்னும் மாறியில் சேமிக்கப்பட்டுள்ள மதிப்பு cout பொருளுக்கு அனுப்பி வைக்கப்படும். marks-ன் மதிப்பு திரையில் காட்டப்படும்.

```
cout << "\n Marks obtained is : " <<marks;
```

என்னும் இரண்டாவது கட்டளை, செய்தியையும் மாறியின் மதிப்பையும் திரைக்கு அனுப்பிவைக்கும். இவ்வாறு, வெளியீட்டுச் செயற்குறியை அடுத்தடுத்து அமைத்து, ஒன்றுக்கு மேற்பட்ட விவரங்களை ஒரே கட்டளையில் வெளியிட முடியும்.

இன்னும் சில எடுத்துக்காட்டுகள்:

```
cout << "\n The sum of the variables a,b .." << a+b;
cout << a+b << '\t' << a-b << '\t' << a/b;
cout << "\n The difference of numbers …." << a-b
      << "\n The sum of two numbers  …. " << a+b;
```

3.2 முதல் சி++ நிரல் – சி++ நிரலின் கட்டமைப்பு

```
// My first program - Program 3.1
# include <iostream.h>//preprocessor directive
# include <conio.h>
float fact = 1; // declaration of variables
int term;
int main()  // function header
{
   clrscr(); // predefined function
   cout << "\n This program computes factorial of a
                                              number";
   cout << '\n' << "Enter a number ...";
   cin >> term;
   // looping statement
   for(int x = 2; x <= term;fact *= x,x++);
   cout << "\nThe factorail of the given number .."
         << term << "  is .." << fact;
   return 0;
}
```

மேலேயுள்ள நிரலை நோக்குக. ஒரு சி++ நிரல் மூன்று முக்கிய பிரிவு களைக் கொண்டுள்ளது:

- சேர்த்துக்கொள்ளும் கோப்புகள்
- மாறிகள், பயனர் வரையறுக்கும் செயற்கூறுகளின் அறிவிப்பு
- main() செயற்கூறு.

பிழையேதுமின்றி நிரல்பெயர்ப்புச் செய்தபின், நிரலை இயக்கும்போது main() செயற்கூறு தானாகவே செயல்படுத்தப்படும். இந்தச் செயற்கூறில்தான், நிரலின் பிற கூறுகளையும், பிற செயலாக்கு கட்டளைகளையும் அழைக்கின்ற கூற்றுகள் இடம்பெற வேண்டும்.

3.3 **அழைப்புக் கூற்றுகள்** (Declaration Statements)

ஒரு நிரலிலுள்ள மாறிகள் பயன்படுத்தப்படுவதற்கு முன்பாக அறி விக்கப்பட்டு வரையறுக்கப்பட வேண்டும்.

```
int *iptr;        // int இனத்தைச் சுட்டும் ஒரு சுட்டினை அறிவிக்கிறது.
iptr = new int; // int இனத் தரவை இருத்திவைக்க நினைவக இடம்
                   ஒதுக்கப்படுகிறது-எனவே சுட்டு மாறி வரையறுக்கப்
                   பட்டு விட்டது.
*iptr = 5;  // 5  என்னும் மதிப்பு இருத்தப்படுகிறது. நினைவகம்
                   ஒதுக்கப்பட்ட பிறகே இது நிகழ்கிறது.
```

ஒரு மாறியின் அறிவிப்பு, மாறியின் பெயரையும் அது சார்ந்த தரவு இனத்தையும் அறிமுகப்படுத்துகிறது. எடுத்துக்காட்டாக, int *iptr; என்னும் அறிவிப்பை எடுத்துக் கொள்ளுங்கள். ‘iptr என்பது int இனத்தைச் சுட்டும் ஒரு சுட்டு மாறி’ என இக்கூற்றினைப் புரிந்துகொள்ள வேண்டும். அனைத்துச் சுட்டு மாறிகளும், தரவு மதிப்பை இருத்திவைப்பதற்குரிய நினைவகம் ஒதுக்கப் படும்போதுதான் வரையறுக்கப்படுகின்றன.

பயனர் வரையறுக்கும் தரவினங்கள், குறிப்பெயர்கள், செயற்கூறு தலைப்புகள், சுட்டு மாறிகள் மற்றும் இவைபோன்றவற்றை அறிவிப்பதற்கு அறிவிப்புக் கூற்றுகள் பயன்படுகின்றன. பாடப்பிரிவு 2.3-ல் பயனர் வரை யறுக்கும் தரவினங்களைப் பற்றிப் படித்ததை நினைவு கூர்க.

ஒரு மாறியை அறிவிக்கும்போதே, நினைவகம் ஒதுக்கப்படுகிறது எனில், அந்த அறிவிப்பை வரையறை (definition) என்றே சொல்லலாம். எடுத்துக் காட்டாக,

```
int num;
```

என்னும் அறிவிப்புக் கூற்றினை நோக்குக. இந்தக் கூற்று, வரையறைக் கூற்று

என்றே கூறப்படுகிறது. காரணம், தரவினை இருத்திவைக்க நினைவகம் ஒதுக்கப்பட்டுவிடுகிறது. கீழேயுள்ள கட்டளைத் தொகுதியை நோக்குங்கள்:

int num; // ஓர் int மாறியின் அறிவிப்பும் வரையறையும்.
num = 5; // மதிப்பு 5 இருத்தப்படுகிறது.

நினைவக ஒதுக்கீட்டுக்கென வெளிப்படையான கோரிக்கை எதுவும் இல்லை என்பதைக் கவனித்தீர்களா? காரணம் char, int, float போன்ற அடிப்படைத் தரவினங்களில் ஒரு மாறியை அறிவிக்கும்போதே நினைவகம் ஒதுக்கப்பட்டு விடுகிறது.

- ✓ அறிவிப்புக் கூற்று ஒரு மாறியின் பெயரை அறிமுகப்படுத்துகிறது. அதனை ஒரு குறிப்பிட்ட தரவினத்தோடு தொடர்புபடுத்துகிறது.
- ✓ ஒரு மாறிக்கு நினைவகத்தில் உரிய இடம் ஒதுக்கப்படும்போது அது வரையறுக்கப்படுகிறது.
- ✓ சில இன மாறிகளை அறிவிக்கும்போதே நினைவகம் ஒதுக்கப் பட்டு விடுகிறது.
- ✓ சுட்டு இன மாறிகள் அறிவிக்கப்படும்போதே வரையறுக்கப் படுவதில்லை. நினைவக ஒதுக்கீடு செய்யப்படும்போதே அவை வரையறுக்கப்படுகின்றன. new செயற்குறி மூலம் நினைவக ஒதுக்கீடு செய்ய முடியும்.

3.4 மதிப்பிருத்து கூற்றுகள் (Assignment Statements)

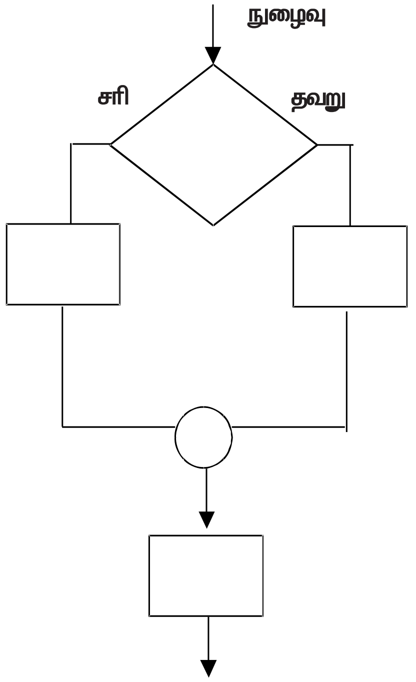
மதிப்பிருத்து கூற்று, மதிப்பிருத்து செயற்குறிக்கு வலப்புறமுள்ள ஒரு கோவையின் மதிப்பை இடப்புறமுள்ள மாறியில் இருத்துகிறது. = என்பது மதிப்பிருத்து செயற்குறி ஆகும். எடுத்துக்காட்டாக, மாறிகளில் மதிப்பிருத்தும் பல்வேறு வகையான கூற்றுகளைக் காண்க:

```
num = 5;
total = english+maths;
sum += class_marks;
```

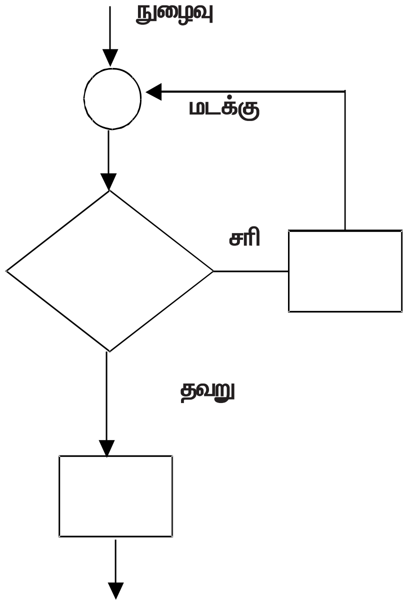
மதிப்பிருத்து செயல்பாட்டின்போது, சி++ நிரல்பெயர்ப்பி, வலப்புறமுள்ள கோவையின் தரவினத்தை, இடப்புறமுள்ள மாறியின் தரவினத்துக்கு மாற்றிக் கொள்கிறது. 2.4.2. - பாடப்பிரிவில் விளக்கப்பட்ட உள்ளுறை / வெளிப்படை இனமாற்றங்களை நினைவு கூர்க.

3.5 கட்டுப்பாட்டுக் கட்டளை அமைப்புகள் (Control Structures)

ஒரு நிரலில் உள்ள கூற்றுகள் கட்டாயமாக ஒன்றன்பின் ஒன்றாய் வரிசையாகத்தான் நிறை வேற்றப்பட வேண்டும் என்கிற தேவை இல்லை. நிரலின் சில பகுதிகள் ஒரு நிபந்தனையின் பேரில் நிறைவேற்றப்படுவதும் உண்டு. அப்படிப்பட்ட சூழ்நிலையில், பாய்வுக் கட்டுப்பாடு நிரலின் ஒரு பகுதியிலிருந்து இன்னொரு பகுதிக்குத் தாவுகிறது. அத்தகைய தாவலுக்குக் காரணமான நிரல் கூற்றுகள் கட்டுப்பாட்டுக் கூற்றுகள் அல்லது கட்டுப் பாட்டுக் கட்டமைப்புகள் எனப்படுகின்றன. கீழேயுள்ள பாய்வுப் படங்களை நோக்குக:



பாய்வுப்படம்–1 : தேர்ந்தெடுப்பு



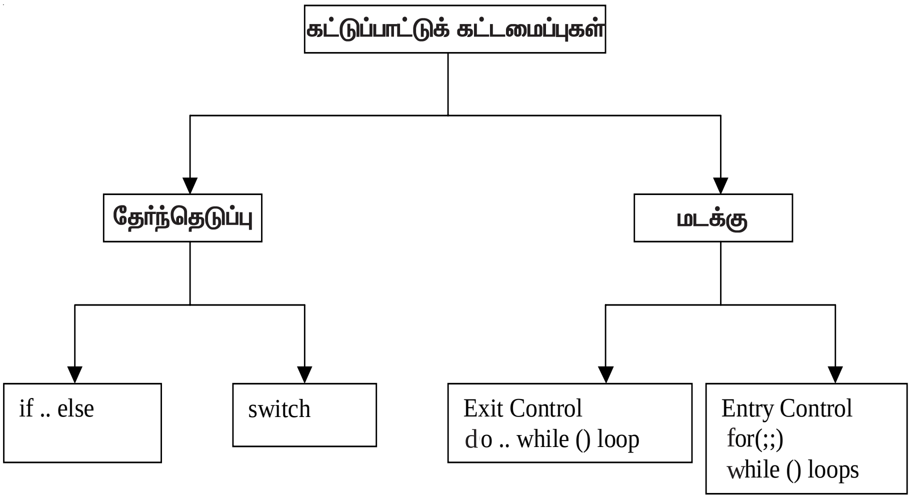
பாய்வுப்படம்–2: மடக்கு

1. ஒரு முழுஎண்ணை உள்ளீடாகப் பெற்று, அது ஒற்றைப்படை எனில் அதனோடு 1 கூட்டியும், இரட்டைப்படை எனில் அதே எண்ணையும் வெளியிடும் படிநிலைகளைப் பாய்வுப் படம் 1-ல் காண்க. செயல்பாடு கள் ஒன்றன்பின் ஒன்றாய் வரிசையாக நிறைவேற்றப்படுகின்றன.
2. ஒரு முழுஎண்ணை உள்ளீடாகப் பெற்று, அது இரண்டால் வகுபடுகிறதா என்பதைச் சோதித்து, அதனடிப்படையில் இரட்டைப்படை அல்லது ஒற்றைப்படை என்னும் செய்தியைக் காட்டும் படிநிலைகளை நோக்குக. இதில் வகுக்கும்போது மீதி வருகிறது எனில் M=ODD என்ற

கூற்றுக்கும், மீதி வரவில்லை எனில் M=EVEN என்ற கூற்றுக்கும் பாய்வின் கட்டுப்பாடு கிளை பிரிகிறது.

✓ கட்டுப்பாடு, நிரலின் ஒரு பகுதியிலிருந்து இன்னொரு பகுதிக்குத் தாவுவதற்குக் காரணமான நிரல் கூற்றுகள் கட்டுப்பாட்டுக் கட்டமைப்புகள் எனப்படுகின்றன.

கட்டுப்பாட்டுக் கட்டளை அமைப்புகளுள் இரண்டு முக்கிய வகையினங்கள், தீர்மானிப்புக் கூற்றுகளும் (Decision Making Statements), மடக்குக் கூற்றுகளும் (Looping Statements) ஆகும். தீர்மானிப்புக் கூற்றுகளைத் தேர்ந்தெடுப்புக் கூற்றுகள் (Selection Statements) என்றும் கூறலாம். சி++ மொழியில் கட்டுப்பாட்டுக் கட்டமைப்புகள் கீழேயுள்ள படத்தில் கண்டுள்ள வாறு கட்டுப்பாட்டுக் கூற்றுகளால் நடைமுறைப்படுத்தப்படுகின்றன.



3.5.1 தேர்ந்தெடுப்புக் கூற்றுகள் (Selection Statements)

ஒரு நிரலில் நிபந்தனை அடிப்படையிலான தீர்மானிப்பு (Decision), நிரலின் வேறு பகுதிக்கு ***ஒருமுறைத் தாவல்*** நடைபெறக் காரணமாகிறது. சி++ மொழியில் தீர்மானிப்புகள் பல்வேறு முறைகளில் மேற்கொள்ளப் படுகின்றன. அவற்றுள் மிகவும் முக்கியமானது, இரண்டு மாற்று வழிகளுள் ஒன்றைத் தேர்ந்தெடுக்கின்ற if... else... கூற்றாகும். இன்னொரு தீர்மானிப்புக் கூற்றான switch, ஒரு மாறியின் மதிப்பு அடிப்படையில், பல்வேறு கிளைப்

பிரிவுகளுள் ஒரு கிளைப் பிரிவிலுள்ள கட்டளைத் தொகுதியை நிறைவேற்ற வழிவகுக்கிறது.

if **கூற்று:** அனைத்துத் தீர்மானிப்புக் கூற்றுகளிலும் மிகவும் எளிமையானது. இது இரு வடிவங்களில் நடைமுறைப்படுத்தப்படுகிறது:

- if கூற்று (ஒருவழி)
- if..else கூற்று (இருவழி)

```
if (நிபந்தனை/ கோவை)
{
    // செயல்பாட்டுத் தொகுதி
}
```

```
if (நிபந்தனை/ கோவை)
{
    //செயல்பாட்டுத் தொகுதி-1
}
else
{
    //செயல்பாட்டுத் தொகுதி-2
}
```

கீழேயுள்ள நிரல் (Program - 3.2) if கூற்றை விளக்குகிறது:

```
// Program - 3.2
# include <iostream.h>
# include <conio.h>
// Demonstrates the use and syntax of if statement
void main()
{
  int a;
  clrscr();
  cout << "\nEnter a number ";
  cin >> a;
  if ( a%2 == 0)
      cout << "\nThe given number " << a << "is even";
  getch();
}
```

மேற்கண்ட நிரலில், தரப்பட்ட நிபந்தனை சரி என்றால் மட்டுமே " The given...." என்னும் செய்தி, காட்டப்படும். இல்லையேல் நிரலின் கட்டுப்பாடு, cout<<"\n The given... என்னும் கட்டளையை நிறைவேற்றாமல் அதனைக் கடந்து getch() கூற்றுக்குத் தாவிவிடும்.

கீழேயுள்ள நிரல் (Program 3.3) if.. else... கூற்றை விளக்குகிறது:

```
// Program - 3.3
// Demonstrates the use and syntax of if else statement

# include <iostream.h>
# include <conio.h>
void main()
{
     int a;
     clrscr();
     cout << "\nEnter a number ";
     cin >> a;
     if ( a%2 == 0)
         cout << "\nThe given number " << a << "is even";
     else
         cout << "\nThe given number " << a << "is odd";
     getch();
}
```

மேற்கண்ட நிரலில், கொடுக்கப்பட்ட கோவை சரி என்றால் மட்டுமே "The given number 10 is even" என்னும் செய்தி காட்டப்படும். இல்லையேல் else-க்கு அடுத்துள்ள கூற்று செயல்படுத்தப்படும்.

if கட்டளை அமைப்பில் நிபந்தனைகள் / கோவைகள் வெவ்வேறு பாணியில் தரப்பட்டுள்ள எடுத்துக்காட்டுகளை நோக்குக:

branch என்னும் மாறி நிபந்தனையாகப் பயன்படுத்தப்பட்டுள்ளது.

```
int branch = 10 > 20;
if (branch)
{
      // செயல்பாட்டுத் தொகுதி-1;
}
else
{
      // செயல்பாட்டுத் தொகுதி-2;
}
```

சுழியம் (Zero) அல்லாத எந்தவோர் எண்ணும் சரி என எடுத்துக் கொள்ளப்படும் என்பதால், if(1) என்று நிபந்தனையை அமைக்க முடியும்.

```
if (1)
{
    // செயல்பாட்டுத் தொகுதி-1;
}
else
{
    // செயல்பாட்டுத் தொகுதி-2;
}
```

ஒரு கோவையை நிபந்தனையாகத் தரமுடியும். கோவையின் மதிப்ப >0 - ஆக மதிப்பிடப்படுமாயின், செயல்பாட்டுத் தொகுதி-1 நிறைவேற்றப்படும். இல்லையேல் செயல்பாட்டுத் தொகுதி-2 நிறைவேற்றப்படும்.

```
if ( a % 2 )
{
    // செயல்பாட்டுத் தொகுதி-1;
}
else
{
    // செயல்பாட்டுத் தொகுதி-2;
}
```

கீழேயுள்ள நிரலை இயக்கினால் என்ன செய்தி காட்டப்படும் என்பதை முன்னறிந்து சொல்ல இயலுமா?

```
// Program - 3.4
# include <iostream.h>
# include <conio.h>
void main()
{
  int count = 1;
  if (count > 0)
  {
    cout << "\nNegating count ....";
    count *= -1;
  }
  else
  {
    cout << "\nResetting count ...";
    count *= 1;
  }
  getch();
}
```

இந்த நிரலை இயக்கினால் Negating count... என்னும் செய்தி காட்டப் படும். count என்னும் மதிப்பு -1- ஆல் பெருக்கப்பட்ட பின்னும் else-க்குப் பிறகுள்ள கட்டளைத் தொகுதி ஏன் செயல்படுத்தப்படவில்லை என்பதைச் சொல்ல முடியுமா?

இதற்கான பதில்: ஒரு if...else கட்டளை அமைப்பில் if-ன் கீழுள்ள கட்டளைத் தொகுதி நிறைவேற்றப்படுமாயின், else தொகுதி நிறைவேற்றப் படாது. if கட்டளைத் தொகுதி நிறைவேற்றப்படாதபோதுதான் else கட்டளைத் தொகுதி நிறைவேற்றப்படும்.

✓ இரு மாற்று வழிகளுள் ஒன்றைத் தேர்ந்தெடுக்கும் if..else... கட்டளை அமைப்பில் நிபந்தனை அடிப்படையில் ஒரு குறிப்பிட்ட கட்டளைத் தொகுதி மட்டுமே நிறைவேற்றப்படும்.

வ. எண்	பிழையான கட்டளை அமைப்பு	என்ன காரணம்?

1.	if a> b
	  cout << “True”;
		நிபந்தனையைப் பிறை அடைப்புக்குறிகளுக்குள் தர வேண்டும். சரியான கட்டளை:
		if (a>b)
		  cout << “True”;

2.	if (a > b)
	a- -;
	cout<<“\nVariable is decremented”;
	else
	  a++;
	cout << “Variable is incremented ..”
		நிரல் பெயர்ப்பி, “Misplaced else” என்னும் பிழைசுட்டும் செய்தியைத் தரும். செயல் பாட்டுத் தொகுதி ஒன்றுக்கு மேற்பட்ட கட்டளைகளைக் கொண்டிருப்பின் நெளிவு அடைப்புக்குறிகளுக்குள் தர வேண்டும்.
		சரியான கட்டளை அமைப்பு:
		If (a> b)
		{
		  a--;
		  cout<<“\nVariable is
		    decremented”;
		}
		else
		{
		  a++;
		  cout << “Variable is
		    incremented ..” }

3.	if (a > b);
	  cout << “Greater..”;
	else
	  cout << “Lesser ..”;
		நிபந்தனைக்குப் பிறகு தரப் பட்டுள்ள அரைப்புள்ளி, if கூற்றின் நோக்கத்தையே தோற் கடித்து விடும். நிரல்பெயர்ப்பி, “Misplaced else” எனப் பிழை சுட்டும். சரியான வடிவம்:
		if (a > b)
		  cout << “Greater..”;
		else
		  cout <<“Lesser ..”;

அட்டவணை 3.1. if **கட்டளை அமைப்பு**

அட்டவணை 3.2.2-ல் தரப்பட்டுள்ள பணிகளை நிறைவேற்றுவதற்குப் பொருத்தமான if கட்டளை அமைப்பை எழுதிக் காட்டுக:

வ.எண்	**பணி**	if **கட்டளை**
1	மதிப்பெண் 90-க்கு மேல் இருப்பின் ‘A’ என்னும் தரநிலை தருக	
2.	மதிப்பெண் 90-க்கு மேல் இருப்பின் ‘A’ என்னும் தரநிலை தருக. இல்லையேல் ‘B’ தருக.	
3.	வேகம் மணிக்கு 30 கி.மீ-க்கு குறைவு எனில் “Accelerate - traffic to flow” வேகம் மணிக்கு 31-லிருந்து 40 கி.மீ. வரை எனில் “Moderate - accelerate by 10 kmph” இல்லையேல், “Good- be careful..” என்ற செய்திகளைத் திரையிட வேண்டும்.	

அட்டவணை 3.2. if **கட்டளையைப் பயன்படுத்துதல்**

பின்னலான if **கூற்று :** ஒரு if..else கட்டளை அமைப்பிற்குள் இன்னோர் if கட்டளை அமைய முடியும். கீழேயுள்ளவாறு if..else கட்டளை ஒன்றுக்குள் ஒன்றாய்ப் பின்னலாக அமையலாம்.

```
if (கோவை–1)
  if (கோவை–2)
  {
      செயல்பாடு–1;
  }
  else
  {
      செயல்பாடு–2;
  }
else
  {
      செயல்பாடு–3;
  };
```

இதுபோன்று பின்னலாய் அமையும் if..else கட்டளை அமைப்பில், **“ஒவ்வோர் else-க்கும் மேலாக, வேறோர் else- உடன் இணைசேராத ஓர்** if **இருக்க வேண்டும்”**

எடுத்துக்காட்டாக,

```
if (grade = = 'A')
   if (basic > 5500)
        incentive = basic * 10/100;
   else
         incentive = basic * 5/100;
else
    cout << "Try to attain Grade A";
```

மேலேயுள்ள கட்டளைத் தொகுதி செயல்படும் முறை:

- grade== 'A', basic ==5501 எனில் incentive-ன் மதிப்பு 550 ஆகும்.
- grade == 'A', basic ==5000 எனில் incentive-ன் மதிப்பு 250 ஆகும்.
- grade != 'A' எனில், உள் if நிறைவேற்றப்படாது. வெளி else நிறைவேற்றப்படும். எனவே "Try to attain Grade A" என்னும் செய்தி கிடைக்கும்.

மேலேயுள்ள if கட்டளை அமைப்பு, கீழேயுள்ள if கட்டளை அமைப்புக்கு நிகரானதா? உங்கள் பதிலை 'காரணம் கூறுக' பெட்டியில் எழுதுக.

```
if (grade = = 'A'  && basic > 5500)
        incentive = basic * 10/100;
else if (grade = = 'A' && basic <5501)
         incentive = basic * 5/100;
else
    cout << "Try to attain Grade A";
```

காரணம் கூறுக:…………

switch கூற்று: இது ஒரு கிளைபிரிப்புக் கூற்றாகும். ஒரு நிபந்தனையின் அடிப்படையில், தரப்பட்டுள்ள பல்வேறு தேர்வுகளில் ஒன்றுக்குக் கட்டுப்பாட்டை எடுத்துச் செல்லும். இக்கட்டளை இவ்வாறு நடைமுறைப் படுத்தப்படுகிறது:

```
switch (கோவை)
{
   case 1 :செயல்பாட்டுத் தொகுதி-1;
            break;
   case 2 :செயல்பாட்டுத் தொகுதி-2;
            break;
   case 3 :செயல்பாட்டுத் தொகுதி-3;
            break;
   default :செயல்பாட்டுத்தொகுதி-4;
}
```

```
switch (remainder)
{
   case  1 : cout << "remanider 1";
            break;
   case  2 : cout << "remanider 2";
            break;

   default :cout << "Divisible by 3";
}
```

கீழேயுள்ள நிரல், switch கூற்றின் பயன்பாட்டை விளக்குகிறது:

```
// Program - 3.5
// to demonstrate the use of switch statement

# include <iostream.h>
# include <conio.h>

void main()
{
   int a, remainder;
   cout << "\nEnter a number ...";
   cin >> a;
   remainder = a % 3;
   switch (remainder)
   {
      case 1 : cout << "\nRemainder is one";
               break;
      case 2 : cout << "\nRemainder is two";
               break;
      default: cout << "\nThe given number is divisible by 3";
               break;
   }
   getch();
}
```

மேற்கண்ட நிரல்,

- a = 4 அல்லது அதுபோன்று மீதி 1 வரும் எண் எனில், "Remainder is one" எனக் காட்டும்
- a = 5 அல்லது அதுபோன்று மீதி 2 வரும் எண் எனில், "Remainder is two" எனக் காட்டும்.
- a = 3 அல்லது அதுபோன்று 3-ஆல் வகுபடும் எண் எனில், "The given number is divisable by 3" எனக் காட்டும்.

சுருங்கச் சொல்வதெனில், கொடுக்கப்பட்ட எண் மூன்றால் வகுபடும் எண்ணா எனப் பார்த்து அதற்கேற்ப செய்தியைத் திரையில் காட்டும்.

கீழேயுள்ள நிரலின் வெளியீடு என்னவாக இருக்கும் என்று ஊகிக்க முடிகிறதா?

```
// Program - 3.6
// to demonstrate the use of switch statement

# include <iostream.h>
# include <conio.h>

void main()
{
   int rank = 1;
   char name[] = "Shiv";
   switch (rank)
   {
      case 1: cout<<'\n'<<name<<"secured 1st rank";
      case 2: cout<<'\n'<<name<<"secured 2nd rank";
   }
   getch();
}
```

இந்த நிரலின் வெளியீடு இவ்வாறு இருக்கும்:

Shiv secured 1st rank
Shiv secured 2nd rank

case 1, case 2 ஆகிய இரண்டு செயல்பாட்டுத் தொகுதிகளுமே ஏன் செயல் படுத்தப்பட்டுள்ளது என்பது புரிகிறதா? Program-3.5 மற்றும் Program-3.6 ஆகிய இரு நிரல்களிலும் உள்ள செயல்பாட்டுத் தொகுதிகளை ஒப்பிட்டுப் பாருங்கள். Program- 3.6-ல் விடுபட்டுப் போன கட்டளை எதுவெனத் தெரிகிறதா? ஆம். break கட்டளை விடுபட்டுள்ளது. இதிலிருந்து அறியவருவது என்ன?

ஒவ்வொரு செயல்பாட்டுத் தொகுதியிலும் இறுதியில் break கட்டளை இடம்பெற வேண்டும். இல்லையேல், நிபந்தனையின் அடிப்படையில் கட்டுப்பாடு தாவிச் சென்று நிற்கும் case-ல் தொடங்கி அடுத்தடுத்துள்ள case-களின் செயல்பாட்டுத் தொகுதிகள் வரிசையாக நிறைவேற்றப்படும்.

மேற்கண்ட எடுத்துக்காட்டில் (Program - 3.6) கட்டுப்பாடு case 1-க்கு எடுத்துச் செல்லப்படுகிறது. அதன் செயல்பாட்டுத் தொகுதி நிறைவேற்றப் படுகிறது. அதில் break கட்டளை இல்லாத காரணத்தால் case 2-ன் செயல் பாட்டுத் தொகுதியும் வரிசையாக நிறைவேற்றப்படுகிறது.

> ✓ குறிப்பிட்ட case-ன் செயல்பாட்டுத் தொகுதி நிறைவேற்றப்பட்டவுடன் switch கூற்றைவிட்டு வெளியேற, அத்தொகுதியின் இறுதியில் break கட்டளையைச் சேர்க்க வேண்டும்.

கீழே காணும் swich கட்டளை அமைப்புகள் பிழையானவை. காரணம் காண்க:

1.
```
char name[ ] = "Shiv";
switch (name)
{
    case "Shiv" : cout << '\n'
    << name << "secured
                1st rank";
    case "Rama" : cout << '\n'
    << name << " secured
                2nd rank";
}
```

"switch selection expression must be of integral type" என நிரல் பெயர்ப்பி பிழைசுட்டும். இதன்பொருள்,switch-ல் இடம் பெறும் கோவையின் மதிப்பு, முழுஎண்(char, enum, int) இனம்சார்ந்த மாறிலியாக இருக்க வேண்டும்.

2.
```
float value;
switch (value)
{
    case 1.5 : cout << '\n'
            << value – 0.5;
    case 2.9 : cout << '\n'
            << value + 0.1;
}
```

switch() கூற்றில் இடம்பெறும் கோவை float இனமாக இருப்பதால் பிழையாகிறது.

3. switch (rank)

```
switch (rank)
{
    case 1 to 2  : cout << '\n'
           << "Best rank";
                break;
    case 3 to 4 : cout << '\n'
           << "Good rank";
}
```

case 1 to 2 என்பது ஏற்கத் தகாத கூற்றாகும். case-ல் ஒரேயொரு முழுஎண் மதிப்பு மட்டுமே இடம்பெற வேண்டும். ஒரு குறிப்பிட்ட செயல்பாட்டுத் தொகுதியை ஒன்றுக்கு மேற்பட்ட மதிப்புகளுக்குப் பயன்படுத்த வேண்டுமெனில், இந்த நிரல் பகுதியை இவ்வாறு எழுதவேண்டும்.

```
switch (rank)
{case 1 :
 case 2 :     cout << "Best  rank";
              break;
case 3 :
case 4 :      cout << "Good   rank";
              break;
}
```

3.5.2. மடக்குகள் (Loops)

மடக்குகள், ஒரு கட்டளைத் தொகுதியை குறிப்பிட்ட தடவைகள் திரும்பத் திரும்ப நிறைவேற்றுகின்றன. எடுத்துக்காட்டாக, கீழேயுள்ள Program - 3.7-ஐ நோக்குக:

```
// Program - 3.7
# include <iostream.h>
# include <conio.h>

void main()
{
  int i = 1;
  loop_start:
    if (i < 6)
    {
        cout << i <<'\t';
        i = i + 1;
        goto loop_start;
    }
}
```

if (i < 6) → **கட்டளைத் தொகுதியை நிறைவேற்றுவதற்குப் பரிசோதிக்க வேண்டிய நிபந்தனை**

திரும்பத் திரும்ப நிறைவேற்றப்பட வேண்டிய கட்டளைத் தொகுதி

goto loop_start; → **நிரலின் கட்டுப்பாடு, திரும்பத் திரும்ப நிறைவேற்றப்பட வேண்டிய கட்டளைத் தொகுதியின் தொடக்கத்துக்கு எடுத்துச் செல்லப்படும்.**

மேலே காணும் நிரலை இயக்கினால், if கூற்றின் கீழ் தரப்பட்டுள்ள செயல்பாட்டுத் தொகுதி 5 முறை நிறைவேற்றப்பட்டு, 1 முதல் 5 வரையிலான எண்களைத் திரையில் காட்டும்.

Program– 3.7 இவ்வாறு செயல்படுகிறது:

1. i என்னும் மாறி அறிவிக்கப்பட்டு தொடக்க மதிப்பிருத்தப்படுகிறது.
2. i < 6 என்னும் ஒப்பீட்டுக் கோவை பரிசோதிக்கப்படுகின்றது.
3. நிபந்தனை சரி எனில் செயல்பாட்டுத் தொகுதி (cout<<i;i=1;) நிறைவேற்றப்படுகிறது. நிரலின் கட்டுப்பாடு மீண்டும் loop_start என்னும் இடத்துக்கு எடுத்துச் செல்லப்படுகிறது (goto loop_start). ஒப்பீட்டுக் கோவையின் நிபந்தனை அடிப்படையில் ஒரு கட்டளைத் தொகுதி திரும்பத் திரும்பச் செயல்படுத்தப்படுகிறது. i என்ற மாறி, கட்டுப்பாட்டு மாறி எனக் குறிப்பிடப்படுகிறது. காரணம், கட்டளைத் தொகுதி திரும்பத் திரும்ப நிறைவேற்றப்பட i-ன் மதிப்பே காரண மாக அமைகின்றது.

ஆக, மடக்குத் தொகுதி, உடற்பகுதி, கட்டுப்பாட்டுக் கூற்று என்னும் இரு பகுதிகளைக் கொண்டுள்ளது. கட்டுப்பாட்டுக் கூற்று நிபந்தனையைச் சரிபார்க்கிறது. அதனடிப்படையில், நிரலின் கட்டுப்பாடு, மடக்கின் உடற் பகுதிக்கு மீண்டும் மீண்டும் திருப்பிவிடப்பட்டு, திரும்பத் திரும்ப நிறை வேற்றப்படுகிறது. இப்போது கீழேயுள்ள நிரல்களை நோக்குங்கள்:

```
//Program - 3.7 A
void main()
{
  int i = 6;
  loop_start:
    if (i < 6)
    {
        cout << i << '\t';
        i = i + 1;
        goto loop_start;
    }
   cout << i;
}
```

```
//Program - 3.7 B
void main()
{
  int i = 6;
  loop_start:
     {
        cout << i << '\t';
        i = i + 1;
       if (i < 6)
          goto loop_start;
    }
 cout << i;
}
```

மேற்கண்ட நிரல்களின் வெளியீடு என்னவாக இருக்குமென்று நினைக் கிறீர்கள்?

Program-3.7A, 6 எனத் திரையில் காட்டும். Program-3.7B, 7 எனக்காட்டும். இந்த நிரலில், (cout<<i; i=i+1;) ஆகிய கட்டளைகளுக்குப் பிறகு நிபந்தனை இடம்பெற்றுள்ளது. எனவே இக்கட்டளைகள் ஒருமுறை நிறைவேற்றப்பட்டு விடும். அதன்பிறகு நிபந்தனை சரிபார்க்கப்படுகிறது. இப்போது i-ன் மதிப்பு 7 என்பதால் நிரலின் கட்டுப்பாடு loop_start இருக்குமிடத்துக்கு எடுத்துச் செல்லப்படவில்லை. இதிலிருந்து நாம் அறிவது என்ன?

- ✓ மடக்கில் உடற்பகுதியின் இறுதியில் நிபந்தனை தரப்படுமாயின், கட்டளைத் தொகுதி ஒரு முறையேனும் நிறைவேற்றப்பட்டுவிடும்.
- ✓ நிபந்தனை இடம்பெறும் இடத்தைப் பொறுத்து, மடக்குகளை, நுழைவு-சோதிப்பு மடக்கு (Program-3.7A), வெளிறேல்-சோதிப்பு மடக்கு (Program - 3.7B) என இருவகையாகப் பிரிக்கலாம்.

பொதுவாக ஒரு மடக்கின் செயலாக்கம், இவ்வாறு நடைபெறுகிறது:

1. நிபந்தனை மாறியில் தொடக்க மதிப்பு இருத்தப்படுகிறது.
2. மடக்கின் உடற்பகுதி ஒருமுறை நிறைவேற்றப்படுகிறது.
3. நிபந்தனை மாறியின் மதிப்பு மிகுக்கப்படுகிறது.
4. நிபந்தனை மாறி, ஓர் ஒப்பீட்டுக் கோவையில் பரிசோதிக்கப் படுகிறது. ஒப்பீட்டுக் கோவையின் மதிப்பு அடிப்படையில் கட்டுப் பாடு, உடற்பகுதியின் தொடக்கத்துக்கு எடுத்துச் செல்லப்படும், அல்லது மடக்கை விட்டு வெளியேறும்.

சி++ மொழியில், மூன்று வகையான மடக்குகள் உள்ளன:

1) for() மடக்கு
2) while() மடக்கு
3) do..while() மடக்கு

do...while() மடக்கு:

இதன் கட்டளை அமைப்பு இவ்வாறு இருக்கும்:

```
do
{
    செயல்பாட்டுத் தொகுதி;
} while <(நிபந்தனை)>
```

கீழேயுள்ள நிரலை நோக்குங்கள்:

```
// Program - 3.8
# include <iostream.h>
# include <conio.h>

// to print the square of numbers
// between 2 to 5

void main()
{
   clrscr();
   int num = 2;
   do
   {
       cout << num * num << '\t';
       num += 1;
   }while (num < 6);
   getch();
}
```

Program -3.8 நிரலின் அடிப்படையில் கீழேயுள்ள வினாக்களுக்கு விடை தருக:

அ) அடையாளம் காண்க:

1. பயன்படுத்தப்பட்டுள்ள கட்டுப்பாட்டு மாறி
2. மடக்கின் உடல்பகுதியிலுள்ள கட்டளைகள்
3. பரிசோதிப்புக் கோவை

ஆ) மடக்குக் கட்டளைகள் எத்தனை முறை நிறைவேற்றப்படும்?

இ) நிரலின் வெளியீடு என்ன?

ஈ) இது என்ன வகையான மடக்கு?

விடைகள்:

அ) 1. கட்டுப்பாட்டு மாறி num

2. மடக்கின் உடல்பகுதிக் கட்டளைகள்:

```
cout<<num*num<<'\t';
        num += 1;
```

3. (num<6) என்பது பரிசோதிப்புக் கோவை

ஆ) நான்கு முறை

இ) 4 9 16 25

ஈ) வெளியேறல்-சோதிப்பு மடக்கு

இந்த மடக்கின் செயல்பாட்டுப் படிநிலைகளைக் காண்க:

1. மடக்கில் நுழைகிறது.
2. num-ன் இரண்டாம் அடுக்கு மதிப்பைத் திரையில் காட்டுகிறது.
3. கட்டுப்பாட்டு மாறியின் மதிப்பு 1 மிகுக்கப்படுகிறது.
4. நிபந்தனை சோதிக்கப்படுகிறது. சோதனை முடிவின் அடிப்படையில் கட்டுப்பாடு படிநிலை-2க்கு மாற்றப்படுகிறது.
5. மடக்கு முடிவு பெறுகிறது.

do..while(நிபந்தனை) என்னும் மடக்கு வெளியேறல் - சோதிப்பு மடக்கு எனப்படும். காரணம், நிபந்தனைக் கோவை, மடக்கின் உடற்பகுதிக் கட்டளை களுள் இறுதியாக இடம்பெற்றுள்ளது. நிபந்தனைக் கோவையைப் பல்வேறு வகையில் அமைக்க முடியும். எடுத்துக்காட்டுகள் காண்க:

```
int ctr = 1, sum = 0, check = 1;
do
{
   cout << ctr;
    sum = sum + ctr;
    ctr = ctr + 2;
    check = (ctr < 11);
}while(check);
```

```
int ctr = 5, sum = 0;
do
{

   cout << ctr;
    sum = sum + ctr;
    ctr = ctr -  2;

}while(ctr);
```

```
int ctr = 5,sum = 0,c=1;
do
{

   cout << ctr;
    sum = sum + ctr;
    ctr = ctr -  2;

}while(ctr >= 1);
```

கீழேயுள்ள நிரல்களின் வெளியீடு என்னவாக இருக்கும்?

```
// snippet A
# include <iostream.h>
# include <conio.h>

void main()
{
   int i = 10;
   do
   {
        cout << i;
        i--;
   } while (i <= 10);
   getch();
}
```

```
//snippet B
# include <iostream.h>
# include <conio.h>

void main()
{
  int i = 10; choice = 1;
  do
  {
    cout << i;
    i++;
  }while (choice);
  getch();
}
```

Snippet A-யில் i-ன் மதிப்பு -32768 ஆகும்வரை மடக்கு நிறைவேற்றப்படும். Snippet A, Snippet B ஆகிய இரண்டு நிரல்களிலும் உள்ள மடக்குகள் வரம்பிலா மடக்குகளாகும். இரண்டு மடக்குகளிலும் தரப்பட்டுள்ள நிபந்தனைகள் எப்போதுமே சரியாகவே இருக்கும். Snippet A-யில் i-ன் மதிப்பு எப்போதுமே 10-ஐ விடக் குறைவாகவே இருக்கும். Snippet B-யில் choice-ன் மதிப்பு எப் போதுமே 1 என்பதால் நிபந்தனை எப்போதுமே சரி என்றாகி விடுகிறது. ஒரு கட்டத்தில் நிபந்தனை ‘தவறு’ என்று ஆகுமாறு சரியான நிபந்தனையை அமைப்பது மிகவும் முக்கியமாகும். மடக்கில் தரப்படும் சோதிப்புக் கோவை யோடு தொடர்புடைய கட்டுப்பாட்டு மாறியின் மதிப்பை மிகுத்து/புதுப்பித்து இந்த நிலையை எட்ட வேண்டும்.

while() மடக்கு:

இது நுழைவு-சோதிப்பு மடக்கு எனப்படும். இதன் கட்டளை அமைப்பு:

```
while <(condition)>
{
   செயல்பாட்டுத் தொகுதி;
}
```

while() கூற்றில் உள்ள சோதிப்புக் கோவை ‘சரி’ என்னும் விடையைத் தந்தால் மட்டுமே மடக்கின் உடற்பகுதியிலுள்ள கட்டளைகள் நிறைவேற்றப் படும். சோதிப்புக் கோவையின் மதிப்பு ‘தவறு’ என ஆகும்போது மடக்கு முடிவுக்கு வரும். do..while() மடக்கில் விளக்கப்பட்ட அதே நிரலை while() மடக்கு பயன்படுத்தி மீண்டும் எழுதுவோம். (Program-3.9).

```
// Program - 3.9
# include <iostream.h>
# include <conio.h>

// to print the square of numbers
// between 2 to 5

void main()
{
   clrscr();
   int num = 2;
   while (num < 6)
   {
         cout << num * num << '\t';
         num += 1;
   }
   getch();
}
```

மடக்கின் உடற்பகுதியின் தொடக்கத்தில் சோதிப்புக் கோவை அதாவது நிபந்தனை தரப்பட்டுள்ளது.

மேற்கண்ட மடக்கு இவ்வாறு செயல்படும்:

1. num என்னும் கட்டுப்பாட்டு மாறியில் 2 என்னும் தொடக்க மதிப்பு இருத்தப்படுகிறது.
2. num<2 என்னும் சோதிப்புக் கோவை மதிப்பிடப்பட்டு, விடை ‘சரி’ என்றால் மட்டுமே கட்டுப்பாடு படிநிலை-3க்கு மாற்றப்படுகிறது.
3. num என்னும் மாறியில் இருத்தப்பட்ட மதிப்பின் இரண்டாம் அடுக்கு, திரையில் காட்டப்படுகிறது.
4. num மாறியின் மதிப்பு ஒன்று மிகுக்கப்படுகிறது.
5. நிரலின் கட்டுப்பாடு படிநிலை-2க்கு மாற்றப்படுகிறது.
6. மடக்கு முடிவு பெறுகிறது.

மேற்கண்ட நிரல் Program- 3.10- ன் அடிப்படையில் கீழ்க்காணும் வினாக் களுக்கு விடை அளிக்கவும்.

```
//Program-3.10
# include <iostream.h>
# include <conio.h>

void main()
{
  int x = 3, y = 4, ctr = 2,res = x;
  while(ctr <= y)
  {
      res *= x;
      ctr += 1;
  }
  cout << "x to the power of y is : " << res;
  getch();

}
```

அ) 1. இந்த நிரலில் பயன்படுத்தப்பட்டுள்ள கட்டுப்பாட்டு மாறி எது?
2. மடக்கின் உடற்பகுதிக் கட்டளைகள் எவை?
3. பரிசோதிப்புக் கோவை எது?

ஆ) மடக்கு எத்தனைமுறை நிறைவேற்றப்படுகிறது?

இ) இந்த நிரலின் வெளிப்பாடு யாது?

ஈ) இதில் பயன்படுத்தப்பட்டுள்ளது எந்த வகையான மடக்கு?

விடைகள்:

அ) 1. கட்டுப்பாட்டு மாறி ctr
2. res *= x; ct +=1;
3. ctr <=y

ஆ) மூன்று தடவைகள்

இ) 81

ஈ) ‘நுழைவு - சோதிப்பு’ அல்லது ‘நுழைவுக் கட்டுப்பாட்டு மடக்கு’ வகை

கீழேயுள்ள நிரல் Program-3.11-ன் வெளியீடு என்னவாக இருக்கும்? இந்த நிரலுக்கு உள்ளீடாக y, y, y, y, n எனத் தரப்பட்டுள்ளதாகக் கொள்க:

```
// Program - 3.11
# include <iostream.h>
# include <conio.h>
void main()
{
  clrscr();
  int counter = 0;
  char choice = 'y';
  while (choice == 'y')
  {
       cout << "Continue <y/n> ...";
       cin >> choice;
       counter = counter + 1;
   }
   cout << "\The loop is executed .."
         << counter << "  times";
   getch();
}
```

கீழேயுள்ள குறுநிரல்கள் பிழையானவை. இவற்றிலுள்ள பிழைகளைக் கண்டறிக. நிரல்கள் சரியாகச் செயல்படுமாறு பிழைகளைத் திருத்துக:

```
//Program - 12 A
# include <iostream.h>
# include <conio.h>
//to print numbers between
5&10
void main()
{
    int start = 5, end = 10;
    while (start >= end)
        cout << start++;
    getch();
}
```

```
//Program - 12 A
# include <iostream.h>
# include <conio.h>
//to print numbers between 5&10
void main()
{
    int start = 5, end = 10;
    while (start <= end)
        cout << start++;
    getch();
}
```

```
//Program - 13 A
# include <iostream.h>
# include <conio.h>
// to print numbers between
10&5
void main()
{
    clrscr();
    int start = 5,end = 10;
    while (start <= end)
        cout << start--;
    getch();
}
```

```
//Program – 13 B
# include <iostream.h>
# include <conio.h>
// to print numbers between 10&5
void main()
{
    clrscr();
    int start = 5,end = 10;
    while (start <= end)
        cout << end--;
    getch();
}
```

```
//Program – 14 A
# include <iostream.h>
# include <conio.h>
// to print numbers
       between 1 & 5
void main()
{
    clrscr();
    int start = 1;
    while (Start <=5)
        cout << start++;
    getch();
}
```

```
//Program – 14 B
# include <iostream.h>
# include <conio.h>
// to print numbers
       between 1 & 5
void main()
{
    clrscr();
    int start = 1;
    while (1)
        cout << start++;
    getch();
}
```

for() மடக்கு:

இது ஒரு நுழைவுக் கட்டுப்பாட்டு மடக்கு. ஒரு செயல்பாட்டைக் குறிப்பிட்ட தடவைகள் நிறை வேற்றுவதற்கு இது பயன்படுகிறது. இதன் கட்டளை அமைப்பு:

for (தொடக்க மதிப்பு; சோதிப்பு நிபந்தனை; மிகுப்பு)
{
செயல்பாட்டுத் தொகுதி;
}

for(;;) மடக்குப் பொதுவாக இவ்வாறு செயல்படும்:

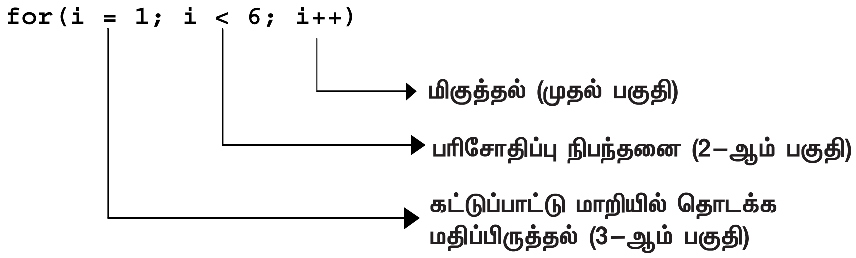
1. நிரலின் கட்டுப்பாடு முதன்முதலில் மடக்கினில் நுழையும்போது, கட்டுப்பாட்டு மாறியில் தொடக்க மதிப்பு இருத்தப்படும்.
2. நிபந்தனை பரிசோதிக்கப்படும். நிபந்தனை ‘சரி’ எனில் மடக்கின் உடற்பகுதி செயல்படுத்தப்படும். இதன் காரணமாகவே இந்த மடக்கு ‘நுழைவுக் கட்டுப்பாட்டு மடக்கு’ எனப்படுகிறது.

3. மடக்கின் உடற்பகுதி மீண்டும் நிறைவேற்றப்படுவதற்கு முன்பாகக் கட்டுப்பாட்டு மாறியின் மதிப்பு மிகுக்கப்பட்டு, மீண்டும் நிபந்தனை பரிசோதிக்கப்படும்.

4. நிபந்தனை 'தவறு' என ஆகும்போது மடக்கு முடிவுக்கு வரும்.

கீழேயுள்ள நிரல் for(;;) மடக்கின் செயல்பாட்டை விளக்கும்:

```
//Program - 3.15
# include <iostream.h>
# include <conio.h>

void main()
{
   int i, fact = 1;
   for(i = 1; i < 6; i++)
        fact *= i;
    cout << "\nThe factorial of the number is .." <<fact;

}
```



- ✓ கட்டுப்பாட்டு மாறியில் தொடக்க மதிப்பிருத்தல், மடக்கின் உடற்பகுதி முதன்முதலில் செயல்படுத்தப்படுவதற்கு முன்பாக, ஒரே ஒருமுறை தான் செயல்படுத்தப்படும்.
- ✓ மடக்கின் உடற்பகுதி ஒவ்வொருமுறை செயல்படுத்தப்படுவதற்கு முன்பாகவும் நிபந்தனை பரிசோதிக்கப்படும்.
- ✓ நிபந்தனை பரிசோதிக்கப்படுவதற்கு முன்பாகக் கட்டுப்பாட்டு மாறியின் மதிப்பு மிகுக்கப்படும்.

கீழேயுள்ள நிரல்களை நோக்குக. இவற்றின் வெளியீடு என்னவாக இருக்கும்? எழுதிக் காட்டுக.

```
//  Program - 3.16
# include <iostream.h>
# include <conio.h>

void main()
{
    int ctr = 10;
    for(; ctr >= 6; ctr—)
       cout << ctr << '\n';
}
```

வெளியீடு:
10
9
8
7
6

```
// Program – 3.17
# include <iostream.h>
# include <conio.h>

void main()
{
  clrscr();
  for(int i=2,fact =1;i<6;fact*=i,i++);
  cout <<"\nThe factorial  .." << fact;
  getch();
}
```

வெளியீடு:

The factorial.. 120

for() கூற்றில் தொடக்க மதிப்பிருத்தல் மற்றும் மிகுப்புப் பகுதியில் ஒன்றுக்கு மேற்பட்ட கூற்றுகள் இடம்பெற்றுள்ளன, கவனித்தீர்களா? கட்டளை அமைப்பின்படியும் தருக்க முறையிலும் மேற்கண்ட கூற்று சரியானதே. for( ) மடக்கிலுள்ள ஒவ்வொரு பகுதியும் பல கூற்றுகளைப் பெற்றிருக்க முடியும். அவை காற்புள்ளியால் பிரிக்கப்பட்டிருக்க வேண்டும். கீழேயுள்ள புரோகிராம் பகுதியின் வெளியீடு என்னவாக இருக்கும்?

```
void main()
{

  for (int i = 1, j = 0 ; i < 8,j<3;i++,j++)
      cout << I << '\t';
  for (int i = 1, j = 0 ;j < 3,i < 8;i++,j++)
      cout << i << '\t';

}
```

வெளியீடு இவ்வாறு இருக்கும்:
1 2 3 // j < 3 வரை மடக்கு செயல்படும்.
1 2 3 4 5 6 7 // j< 8 வரை மடக்குக் செயல்படும். காற்புள்ளிச் செயற்குறி செயல்படும் திசைமுகம் நினைவு கூர்க. ஒரேயொரு நிபந்தனை (வலப் புறம் உள்ளது) மட்டுமே சரிபார்க்கப் படும். ஒன்றுக்கு மேற்பட்ட நிபந்தனைகள் && அல்லது || செயற்குறியுடன் தரப்பட வேண்டும்.

```
// Program – 3.18
# include <iostream.h>
# include <conio.h>

void main()
{
  clrscr();
  int sum =0, ctr = 1;
  for(;ctr <= 5;)
  {
    sum += ctr;
    ctr = ctr + 1;
  }
  cout << "\nSum :" << sum;
  getch();
}
```

வெளியீடு இவ்வாறு இருக்கும்:
sum : 15
தொடக்க மதிப்பிருத்தல் மற்றும் மிகுப்புக் கூற்றுகள் for() அமைப்புக்குள் தரப்படவில்லை என்பதைக் கவனித்தீர்களா?

```
// Program - 3.19
# include <iostream.h>
# include <conio.h>
void main()
{
   clrscr();
   int sum =0, ctr = 1;
   char ch ='y';
   for(;ch == 'y';)
   {
     sum += ctr;
     ctr++;
     cout <<"\nContinue <y/n> ? ";
     cin >> ch;
   }
   cout << "\nSum :" << sum;
   cout << "\nChoice : " << ch;
   getch();
}
```

```
Continue <y/n> ? y
Continue <y/n> ? y
Continue <y/n> ? y
Continue <y/n> ? n
sum:10
Choice : n
```

இதில் for() மடக்கு எத்தனை தடவைகள் நிறைவேற்றப்பட வேண்டும் என்பது இயக்கநேரத்தில் தீர்மானிக்கப்படுவதைக் கவனித்தீர்களா?

கீழேயுள்ள நிரல் பகுதியில் காணப்படும் பிழை யாது?

```
int ctr, sum = 0;
for (ctr =1; ctr <5; ctr++);
   sum += ctr;
cout << sum;
```

இதன் வெளியீடு 5 எனக் காட்டும். காரணம் என்னவென்று புரிகிறதா? for() கூற்றை அடுத்து அரைப்புள்ளி இடப்பட்டுள்ளது. எனவே, sum + = ctr என்னும் கட்டளை, மடக்கின் உடற்பகுதியாகக் கருதப்படவில்லை.

கீழேயுள்ள நிரல்பகுதிகளின் வெளியீடு என்னவாக இருக்கும்?

```
int ctr = 1;
for( ; ctr < 10; ctr++ )
 {
    cout << ctr;
    ctr = 1;
}
```

```
for(it ctr =1; ; ctr++)
      cout << ctr;
```

நிபந்தனைக் கோவை இல்லாத காரணத்தால் int இனத்தின் வரம்பெல்லை –32768, +32767 ஆகியவற்றுக்கு இடையேயான மதிப்புகளைத் தொடர்ந்து காட்டிக் கொண்டிருக்கும். மடக்கு 'முடிவிலா மடக்கு' ஆகிவிடும்.

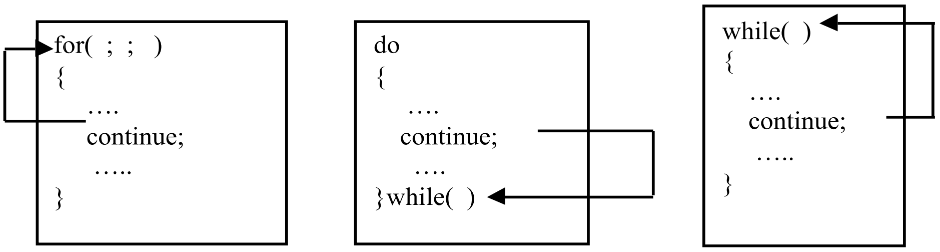
3.5.3. continue

ஒரு மடக்கின் உடற்பகுதியில் இடம்பெறும் continue கட்டளை, மடக்கினை அடுத்த சுழற்சிக்கு இட்டுச்செல்லும். மடக்கின் உடற்பகுதியில் continue கட்டளைக்குப்பின் இடம்பெறும் கட்டளைகள் நிறைவேற்றப்படா.

```
//Program– 3. 20
# include <iostream.h>
# include <conio.h>

void main()
{
  int i = 1,sum = 0;
  for(;i<10;i++)
  {
      if( i % 2 == 0)
       {
         sum += i;
         continue;
       }
       cout <<i;
   }
   cout << "\nSum of evennos.."<<sum;
   getch();
 }
```

கட்டுப்பாடு for() மடக்கின் மிகுப்புப் பகுதிக்கு எடுத்துச் செல்லப்படும்.

பல்வேறு மடக்குகளில் continue செயல்படும் விதத்தைக் கீழே காண்க:



3.5.4. break

பரிசோதிப்பு நிபந்தனை ‘தவறு’ ஆகும்போது, மடக்கின் செயல்பாடு நிறுத்தப்படுகிறது. ஆனால் சில சூழ்நிலைகளில், பரிசோதிப்பு நிபந்தனை என்னவாக இருந்தாலும் குறிப்பிட்ட தருணத்தில் மடக்கினை முடித்துவைக்க நேரலாம். எடுத்துக்காட்டாக, Program - 3.21-ஐ நோக்குங்கள்:

```
//Program - 3.21
# include <iostream.h>
# include <conio.h>

void main()
{
   clrscr();
   int a[] = {1,2,3,4,5,6,7,8,9};
   int search_item = 7;
   for(int x=0; x<9;x++)
  {
       if (a[x] == search_item)
       {
           cout << "\nItem found at position ..” << x;
           break;
        }
   }
    cout << '\n' << "value of index position is ..” << x;
    getch();
}
```

இதன் வெளியீடு இவ்வாறு அமையும்:

Item found at position .. 6
Value of index position is..6

✓ break கட்டளை, நிரலின் கட்டுப்பாட்டை மடக்குக்கு வெளியே எடுத்துச் செல்கிறது. x-ன் மதிப்பு 6 -ஆக இருக்கும் போது, break கட்டளை மடக்கினை முறிக்கிறது.

✓ break கட்டளை அது இடம்பெறும் மடக்கினைவிட்டு வெளியேறச் செய்யும்.
✓ break கட்டளை அது இடம்பெறும் மடக்கினைவிட்டு தாவச் செய்யும்.

பின்னலான மடக்குகள் (Nested Loops)

ஒரு மடங்கின் உடற்பகுதியில் இன்னொரு மடக்கு பின்னலாய் அமையலாம். கீழேயுள்ள நிரல் Program- 3.22-ஐ நோக்குக.

```
// nesting of loops – Program- 3.22
# include <iostream.h>
# include <conio.h>

void main()
{
   clrscr();
   for(int i = 1; i <= 3; i++)
   {
       int j = 1;
       while(j <= i)
       {
         cout << "* ";
         j++;
       }
       cout << '\n';
  }
  getch();

}
```

இதன் வெளியீடு:

```
*
* *
* * *
```

மடக்குகளின் செயல்பாடு இவ்வாறாக அமைகிறது:

```
for(int i = 1; i <= 3; i++)
 {
   int j = 1;
   while(j <= i)
   {
     cout << "* ";
     j++;
   }
   cout << '\n';
 }
```

பின்னல் மடக்குகளின் மடக்குச் செயல் இவ்வாறு அமைகிறது:

for ..loop	while loop
i = 1	ஒருமுறை நிறைவேற்றப்படும் 1 (j<= I)
i = 2	இருமுறை நிறைவேற்றப்படும் (j = 1 .. 2)
i = 3	மும்முறை நிறைவேற்றப்படும் (j = 1.. 3)

கீழேயுள்ள நிரலின் வெளியீடு என்னவாக இருக்கும்?

```
# include <iostream.h>
# include <conio.h>

void main()
{
   clrscr();
  int i = 1, j = 1;
  while(i <= 3)
  {
      cout << '\n';
      for(i=1;i<=j;i++)
       cout << '*';
      i++;
      j++;
  }
   getch();
}
```

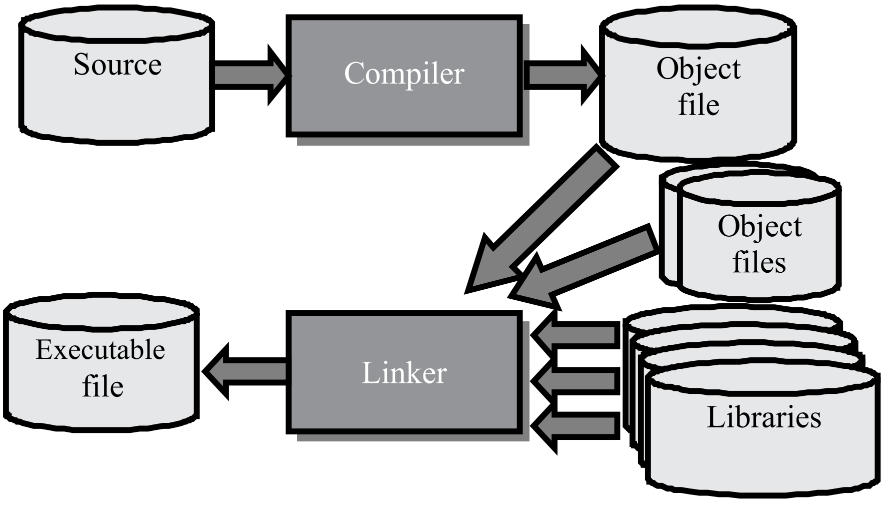
வெளியீடு?

பின்னல் மடக்குகளை அமைப்பதற்கான விதிமுறைகள்:

1. வெளி-மடக்கு, உள்-மடக்கு இரண்டும் ஒரே கட்டுப்பாட்டு மாறியை வைத்துக் கொள்ள முடியாது. காரணம் அது தருக்கப் பிழைகளை ஏற்படுத்தும்.
2. உள்-மடக்கு முழுமையும் வெளி-மடக்கின் உடற்பகுதிக்குள் அமைந்திருக்க வேண்டும்.

3.6 நிரல் உருவாக்கம்

ஒரு கணிப்பொறி மொழியின் இலக்கணத்தைப் பயன்படுத்தி, உயர்நிலை மொழியில் நிரல்கள் எழுதப்படுகின்றன. உயர்நிலை மொழியில் எழுதப்படும் நிரல் மூலக் குறிமுறை (Source Code) எனப்படுகிறது. மூலக் குறிமுறையை எந்திரத்துக்குப் புரியும் வடிவில் மொழிமாற்றம் செய்ய வேண்டும். எந்திர மொழி வடிவில் மாற்றப்பட்ட நிரல் இலக்குக் கோப்பு (Object file) எனப்படுகிறது. நிரல்பெயர்ப்பிகள் (Compilers) மூலக் குறிமுறையிலிருந்து இலக்குக் கோப்பு களை உருவாக்குகின்றன. நிரல்பெயர்ப்பிகள் என்பவை, மூலக் குறிமுறை யிலிருந்து எந்திர மொழிக்கு மாற்றம் செய்யும் மொழிபெயர்ப்பு நிரல்கள் ஆகும். நிரல்பெயர்ப்பி, மொழியின் இலக்கணத்தைச் (கட்டளை அமைப்பை) சரிபார்க்கிறது. பிழையில்லாத மூலக் குறிமுறையிலிருந்தே இலக்குக் கோப்பு உருவாக்கப்படுகிறது. இலக்குக் கோப்புடன் தேவையான நூலகக் கோப்புகள் தொடர்புறுத்தப்பட்டு, ஓர் இயக்குநிலைக் கோப்பு (executable file) உருவாக்கப் படுகிறது. செயல்பாடுகளின் இந்த வரிசை முறையைப் படம் 3.1-ல் காண்க.



படம் 3.1 நிரல் செயலாக்கம்

பயிற்சி வினாக்கள்:

1. கீழ்க்காணும் அறிவிப்புகளைச் சரியானது, பிழையானது எனக்குறிப்பிடுக. பிழையானது எனில், காரணம் கூறுக.

அறிவிப்பு	சரியானது/பிழையானது	காரணம்
int A; a;		
char name(10);		
float f, int;		
double d, float f;		
int 1choice, _2choice		

2. கீழ்க்காணும் நிரலில் பிழை திருத்தி, சரியான நிரலை எழுதுக:

```
include <iostream.h>
include <conio.h>

void main()
{
  int N1,n1;
  cin << ‘ \nEnter two number s ..’;
  result := N1 * n1;
  cout << ‘ \n’ << Result;
}
```

3. கீழ்க்காணும் தகவல்களைப் பொருத்தமான அறிவிப்புக் கூற்றுகளாக எழுதிக் காட்டுக:

அ) 8/3 என்னும் கணக்கீட்டின் விடையை இருத்தி வைக்கவும்.

ஆ) Emp_Name என்னும் மாறியில் “Kalam” என்னும் தொடக்க மதிப்பிருத்துக.

இ) Y-yes, N-no எனக் குறிப்பிட்டுப் பயனரிடமிருந்து உள்ளீடு பெறுக.

4. கீழ்க்காணும் நிரல் பகுதிகளில் உள்ள பிழைகளைச் சுட்டுக் காட்டுக:

அ)
```
int a = 10; b = 5;
   if a > b
cout << a;
```

ஆ) if (a<b) && (a<0)

```
            cout << " a is negative and ..."
```

இ) char option = 'Y';

```
        do while option == 'y'
        {
            cout << ' * ';
            .................
        }
```

ஈ) for (int i = 1; i < 10; i ++)

```
            cout << i * 2;
```

உ) do

```
        {
            cout << ' * ';
        }while (cout << " \n Continue <y/n> ..."; cin >> ch; ch == 'y');
```

5. கீழ்க்காணும் நிரல்கள்/ நிரல்பகுதிகளின் வெளியீடு என்னவாக இருக்கும்?

```
// 5 a.
# include iostream.h>
# include <conio.h>

void main()
{
     int feet;
    const int inch_conversion = 12;
    clrscr();
     cout << "\nEnter feet ...";
     cin >> feet;
     cout << "\nConverted
                          to inches ..."
            << feet * inch_conversion;
}

input – 7 for feet
```

```
// 5 b.
# include <iostream.h>
# include <conio.h>

void main()
{
     int i = 1, sum = 0;
    clrscr();
     while(i++ <= 5)
     {
            cout << '\n' << i;
             s += i;
      }
     cout << "\nValue of the
                 variable i after
                 executing  the
                 while loop .."
                 << i << "\nSum:..."
                 << s;
}
```

```
// 5 c
# include <iostream.h>
# include <conio.h>

void main()
{
  int i = 1, sum = 0;
  clrscr();
  while(++i <= 5)
  {
        cout << '\n' << i;
        sum += i;
  }
  cout << '\n' << i << '\t' << sum;
  getch();
}
```

```
// 5 d
# include <iostream.h>
# include <conio.h>
void main()
{
  int i = 1, sum = 0;
  clrscr();
  for(i = 1; i <= 5; i++)
  {      cout << '\n' << i;
         sum += i;
  }
  cout << '\n' << i << '\t' << sum;
  for(i = 1; i <= 5; ++i)
  {
         cout << '\n' << i;
         sum += i;
  }
  cout << '\n' << i << '\t' << sum;
 }
```

```
//5e
# include <iostream.h>
# include <conio.h>

void main()
{
  int i = 1, j = 1;
  clrscr();
  do
  {
        while (j<=i)
        {
           cout << '#';
           j++;
        }
        cout << '\n';
        i++;
        j = 1;
  } while(i<= 5);
  getch();
}
```

```
// 5 f
# include <iostream.h>
# include <conio.h>

void main()
{
  int num = 1784, s= 0, d = 0, x;
  x = num;
  clrscr();
  for(;num > 0;)
  {     d = num % 10;
        s += d;
        num = num / 10;
  }
  cout << "\nThe sum of digits of "
       << x << "is : "
       << s;
  getch();
}
```

```
//5 g
# include <iostream.h>
# include <conio.h>
void main()
{
  clrscr();
  for(int i = 1,s = 0; ; i++)
  {
        if (i%2 == 0)
           continue;
        s += i;
        if ( i > 9)
           break;
  }
  cout << "\nThe sum is ...." <<
s;
  getch();
```

```
// 5 h
# include <iostream.h>
# include <conio.h>

void main()
{
  clrscr();
  for(int i = 1,x = 0;i <= 5; i++)
    x = x + i%2==0 ? i*1 : i * -1;
  cout << x;
  getch();
}
```

```
//5 j
# include <iostream.h>
# include <conio.h>

void main()
{
  clrscr();
  do
  {
    cout << "\ndo loop ...";
  } while (0);
  getch();
}
```

```
//5 k
# include <iostream.h>
# include <conio.h>

void main()
{
  clrscr();
  int i = 0;
  for(i = -5; i >= 5; i–)
        cout << "Bjarne Stroustrup";
  cout << "\nReturning to Edit Window..";
  getch();
}
```

```
//5 l
# include <iostream.h>
# include <conio.h>

void main()
{
  clrscr();
  int month = 5;
  if (month++ == 6)
     cout << "\nMay ...";
  else if (month == 6)
     cout << "\nJune ...";
  else if (–month == 5)
     cout << "\nMay again ..";
}
```

```
// 5 m
# include <iostream.h>
# include <conio.h>
void main()
{ int day = 3;
  switch (day)
  {
    case 0 : cout << "\nSunday ..";
    case 1 : cout << "\nMonday ..";
    case 2 : cout << "\nTuesday ..";
    case 3 : cout << "\nWednesday .. ";
    case 4 : cout << "\nThursday ..";
    case 5 : cout << "\nFriday ..";break;
    case 6 : cout << "\nSaturday ..";
  }
}
```

```
// 5 n
# include <iostream.h>
# include <conio.h>

void main()
{
  clrscr();
  int bool = 2,b =4;
  while(bool)
  {
    cout << bool << '\t' << ++b << '\n';
    bool–;
    b–;
  }
  getch();
}
```

6. நிரல் எழுதுக:

அ) a மெய்யெண், b முழுஎண். a^b கண்டறியவும். (while() மடக்கு பயன்படுத்துக).

ஆ) கொடுக்கப்பட்ட எண்ணின் தொடர்பெருக்கல் (factorial) கண்டறியவும். (for() மடக்கு பயன்படுத்துக).

இ) n வரையுள்ள ஃபைபோனாசித் தொடரைக் கண்டறியவும். ஃபைபோனாசித் தொடர்: 0, 1, 1, 2, 3, 5, 8, 12, 18, 20, 32, ...

ஈ) கீழ்க்காணும் தோரணிகளை வெளியீடாகத் தருக. (தனித் தனி நிரல் எழுதுக)

1				
1	2			
1	2	3		
1	2	3	4	
1	2	3	4	5

4			
3	4		
2	3	4	
1	2	3	4

A			
B	C		
D	E	F	
G	H	I	J

7. ஒரு மாதத்தின் தேதியை உள்ளீடாகப் பெற்று, கீழ்காணும் செய்திகளை வெளியிடுக. switch() பயன்படுத்துக.

அ) day 1 எனில், “1st day in the month” என வெளியிடுக.

ஆ) day 2/22 எனில், “2nd / 22nd day in the month” என வெளியிடுக

இ) day 3/23 எனில் “ 3rd / 23rd day in the month” என வெளியிடுக.

ஈ) day 4,5,......14, 15,16........ எனில் 4th/5th....... day in the month” என வெளியிடுக.

பாடம் 4

செயற்கூறுகள்
(Functions)

4.1 முன்னுரை

சி++ நிரல்களின் கட்டமைப்புக் கூறுகளாகத் திகழ்பவை செயற்கூறுகள் (Functions) ஆகும். ஒரு நிரலின் செயல்பாட்டுப் பகுதிகளாக அவை விளங்குகின்றன. ஒரு நிரலைச் செயல்படுத்துவதற்கான தொடக்கப் புள்ளி main() செயற்கூறாகும். செயற்கூறுகள்,

- ✓ நிரலின் நீளத்தைக் குறைக்கின்றன.
- ✓ குறிமுறையின் மறுபயனாக்கத்துக்கு உதவுகின்றன.

```
// To print X!/F! – Program - 4.1
# include < iostream.h >
# include <conio.h >
void main ( )
{  int x, f,xfact=1, ffact = 1;
clrscr ( );
cout << “\nEnter values ...”;
cin >> x >> f;   for (int a =1; a < = x; a
++)
xfact * = a;
for (a = 1; a< = f; a++)
ffact * = a;
cout << xfact/ffact;
getch();
}
```

```
// To print X!/F! using functions
// Program - 4.2
# include <iostream.h >
# include <conio.h >
int fact (int num)
{
int factorial = 1;
for (int a=1; a<= num; a++)
 factorial *= a;
return factorial;
}
void main ( )
{   int x, f;
    clrscr ( );
   cout<<“\nEnter values…”;
    cin >> x >> f;
    cout << fact (x)/fact(f);
  }
```

Program - 4.1 நிரலில் தொடர்பெருக்கலைக் கணக்கிடும் கட்டளைகள் திரும்பத் திரும்ப இரண்டுமுறை இடம்பெற்றுள்ளன. Program-4.2 நிரலில் தேவையானபோது fact (int num) என்னும் செயற்கூறு செயல்படுத்தப் பட்டுள்ளது. எனவே, செயற்கூறுகள் கீழ்க்காணும் வகையில் உதவுகின்றன:

- ✓ நிரல் குறிமுறையின் மறுபயனாக்கம் (fact( ) செயற்கூறு இருமுறை அழைக்கப்பட்டுள்ளது)
- ✓ ஒரு செயற்கூறினை தனியாக நிரல்பெயர்த்து (Compiling) வைத்துக் கொண்டால் ஒன்றுக்கு மேற்பட்ட நிரல்கள் அதனைப் பகிர்ந்துகொண்டு பயன்பெற முடியும்.

ஒரு செயற்கூறின் பல்வேறு பகுதிகளைக் காட்டும் அதன் பொதுவான கட்டமைப்பைக் காண்க:

```
// To print X!/F! using functions
// Program - 4.3
# include <iostream.h >
# include <conio.h >
int fact (int num)
{
  int factorial = 1;
  for (int a=1; a<= num; a++)
        factorial *= a;
  return factorial;
}

void main ( )
{   int x, f;
    clrscr ( );
    cout<<"\nEnter values…";
    cin >> x >> f;
    cout << fact (x)/fact(f);
}
```

int fact (int num) → **அளபுருக்களுடன் கூடிய முன்வடிவு**

return factorial; → return **கூற்று, செயற்கூறின் முடிவைச் சுட்டுகிறது. நிரலின் கட்டுப்பாட்டைசெயற்கூறு அழைப்புக் கட்டளைக்கு அடுத்ததாக எடுத்துச் செல்கிறது.**

cout << fact (x)/fact(f); → **செயற்கூறினைச் செயல்படுத்தும் அழைப்புக் கூற்றுகள்**

4.2 செயல்கூறின் முன்வடிவு (Function Prototype)

செயற்கூறுகள், நிரலில் பயன்படுத்தப்படுவதற்கு முன்பாக அறிவிக்கப்பட வேண்டும். செயற்கூறு முன்வடிவின் மூலமாக ஒரு செயற்கூறு அறிவிக்கப் படுகிறது. எடுத்துக்காட்டாக, Program - 4.4 நிரலைக் காண்க.

```
//Program -4.4
# include <iostream.h>
   void fun (char name [ ]) ;        --> செயற்கூறு முன்வடிவு (செயற்கூறின் அறிவிப்பு)

  void main ( )
  {
    char n [ ] = { "C++ programming…."};
    fun (n);
  }
 void fun (char name[] )             --> செயற்கூறின் வரையறை
{ cout << name; }
```

முன்வடிவு நிரல்பெயர்ப்பிக்குக் கீழ்க்காணும் தகவல்களை வழங்குகிறது:

1. செயலுருபுகளின் (arguments) எண்ணிக்கையும் அவற்றின் தரவினமும்-(char name[]) என்பது ஒரு செயலுருபு).
2. திருப்பியனுப்பும் மதிப்பின் தரவினம்.(மேற்கண்ட எடுத்துக்காட்டில் fun() என்னும் செயற்கூறின் தரவினம் void என்பதால், திருப்பியனுப்பும் மதிப்பு எதுவும் இல்லை என்பது பொருள். Program-4.2 -ஐ மீண்டும் நோக்குக. fact() என்னும் செயற்கூறு திருப்பியனுப்பும் தரவினம் int ஆகும்.

செயற்கூறு முன்வடிவின் பொதுவான தொடர் அமைப்பு:

<தரவினம்> <செயற்கூறின் பெயர்> <செயலுருபுகள்>;

எடுத்துக்காட்டு:

```
void fun(char);
int max(int, int);
int max(int a, int b);
```

செயற்கூறு முன்வடிவின் முதன்மையான பயன்பாடு, செயற்கூறினுக்குத் தேவையான தரவுகளைச் சரிபார்த்துக்கொள்ள நிரல்பெயர்ப்பிக்கு உதவுவதே. ஒரு செயற்கூறினை அறிவிக்கும்போதும் வரையறுக்கும்போதும், முன்வடிவுடன் கூடிய ஒரு வார்ப்புரு (Template) பயன்படுத்தப்படுகிறது. ஒரு செயற்கூறு அழைக்கப்படும்போது, சரியான செயலுருபுகள் அனுப்பப்பட்டுள்ளனவா என்பதையும், திருப்பி அனுப்பும் மதிப்புச் சரியாக உள்ளதா என்பதையும உறுதி செய்துகொள்ள நிரல்பெயர்ப்பி இந்த வார்ப்புருவைப் பயன்படுத்திக் கொள்கிறது. செயலுருபுகள் அல்லது திருப்பியனுப்பும் இனம் முன்வடிவில் உள்ளவற்றுடன் பொருந்தவில்லை எனில், நிரல்பெயர்ப்பின்போது, பிழையென நிரல்பெயர்ப்பி சுட்டிக் காட்டும்.

int max (int, int) **என்னும் முன்வடிவு சரியானதா?**

ஒரு செயற்கூறு அறிவிப்பில், செயலுருபுகளின் பெயர்கள் மாதிரி மாறிகளே. எனவே அவற்றைக் குறிப்பிட வேண்டியது கட்டாயமில்லை. முன்வடிவில் குறிப்பிடப்படும் மாறிகள் வெறும் இடம் உணர்த்திகளாகவே (place holders) குறிப்பிடப்படுகின்றன.

செயற்கூறின் வரையறையில் செயலுருபுகளின் பெயர்கள்(மாறிகள்) தேவைப்படுகின்றன. காரணம், செயற்கூறினுள் செயலுருபுகள் எடுத்தாளப் படுகின்றன.

```
void fun(char name[ ])
{
    cout<<name;
}
int fact(int num)
{
    int f =1;
    for(int a = 1;a <= num; a++)
        f *= a;
    return f;
}
```

செயலுருபுகள் name, num ஆகியவை செயற்கூறினுள் எடுத்தாளப்படுவது காண்க.

4.3 ஒரு செயற்கூறினை அழைத்தல்

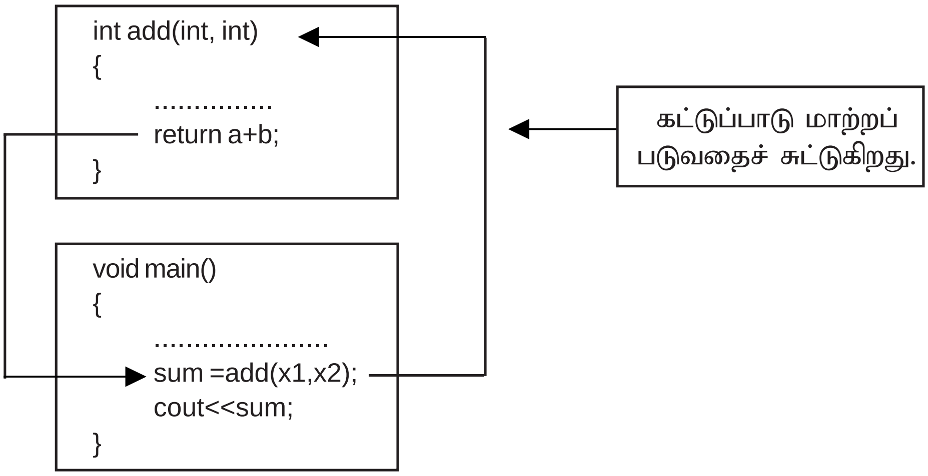
ஒரு செயற்கூறினை அதன் பெயர் குறிப்பிட்டு, வேறொரு செயற்கூறிலிருந்து அழைக்க முடியும். அதாவது, செயல்படுத்த முடியும். செயற்கூறின் பெயருடன் மெய்யான அளபுருக்கள் காற்புள்ளியால் பிரிக்கப்பட்டுப் பிறை அடைப்புக் குறிகளுக்குள் தரப்படும். எடுத்துக்காட்டு:

```
// Program - 4.5
# include <conio.h>
# include <iostream.h>

int add (int a, int b)

{ return a + b;}
void main ( )
{ int x1, x2, sum = 0
  cin >> x1 >> x2;

   sum = add (x1, x2);
  cout << sum;
}
```

முறையான அளபுருக்கள் (Formal Parameters) — int add (int a, int b)

மெய்யான அளபுருக்கள் (Actual Parameters) — sum = add (x1, x2);

ஒரு செயற்கூறின் செயல்பாடுகள்:



4.4 செயற்கூறினுக்கு அளபுருக்களை அனுப்பிவைத்தல்

செயற்கூறினை அழைக்கும் கூற்று, செயலுருபுகள்(Arguments) அல்லது அளபுருக்கள் (Parameters) மூலமாக செயற்கூறோடு தகவல் பரிமாற்றம் செய்கிறது.

அழைப்புக் கூற்றிலிருந்து செயற்கூறினுக்கும் செயற்கூறிலிருந்து அழைப்புக் கூற்றினுக்கும் இடையே தரவுகளின் பாய்வுக்கு அளபுருக்கள் வழித்தடங்களாய் அமைகின்றன.

சி++ மொழியில் செயலுருபுகள் கொண்ட செயற்கூறுகளை, இருவகையில் செயல்படுத்த முடியும்.

- ✓ மதிப்பு மூலம் அழைத்து (Call by value)
- ✓ குறிப்பு மூலம் அழைத்து (Call by reference)

4.4.1 மதிப்பு மூலம் அழைத்தல் (Call by Value)

இந்தவகை அழைப்பின்போது அழைக்கப்பட்ட செயற்கூறு, தனக்கு அனுப்பி வைக்கப்படும் செயலுருபுகளின் மதிப்புகளை இருத்தி வைக்க, புதிய மாறிகளை உருவாக்குகிறது. மெய்யான அளபுருக்களின் (அழைப்புக் கூற்றில் குறிப்பிடப்படும் அளபுருக்கள்) மதிப்புகளை, முறையான அளபுருக்களில் (செயற்கூறின் தலைப்பில் குறிப்பிடப்படும் அளபுருக்கள்) நகலெடுக்கும். இவ் வாறு, செயற்கூறு தனக்கென அளபுருக்களின் நகலை உருவாக்கிப் பயன் படுத்திக் கொள்கிறது. எடுத்துக்காட்டு நிரல் Program -4.5-ஐ நினைவு கூர்க.

```
// Program - 4.5
# include <iostream.h>
# include <conio.h>
int add (int a, int b)
{ return a + b;}
void main ( )
{ int x1, x2, sum;
cin >> x1 >> x2;
sum = add (x, x2);
cout << sum;
}
Assume x 1 = 5, x2 = 7
```

```
main()          add()

 x1 = 5          a = 5
 x2 = 7          b = 7
 sum =
Assume address of the variables :

 x1  = Oxf1      address   data
 x2 = Oxf3       Oxf1       5
 a   = Oxf7      Oxf2
 b   = Oxf9      Oxf3       7
 sum = Oxf6      Oxf4
                 Oxf5
                 Oxf6       12
                 Oxf7       5
                 Oxf8
                 Oxf9       7
```

மெய்யான அளபுருக்களான x1, x2 ஆகியவையும், முறையான அளபுருக்களான a, b ஆகியவையும் வெவ்வேறு நினைவக இருப்பிடங்களில் இருத்தப்பட்டுள்ளன என்பதைக் கவனித்தீர்களா? ஆக, மதிப்பு மூலம் அழைக்கும் முறையில், எப்போதுமே, தரவுகள் அழைப்புக் கூற்றிலிருந்து, செயற்கூறின் வரையறைக்குப் பாயும்.

```
// Program - 4.6
// To exchange values
#include <iostream.h>
#include <conio.h>
# include <iomanip.h>
void swap (int n1, int n2)
{   int temp;
    temp = n1;
    n1 = n2;
    n2 = temp;
    cout << '\n'<<n1<<'\t'<<n2<<'\n';
}
```

```
void main ( )
{
   int m1 = 10, m2 = 20;
   clrscr ( );
   cout <<"\n Values before invoking swap"
        << m1 << '\t' << m2;
  cout << "\n Calling swap..";
  swap (m1, m2);
  cout << "\n Back to main.. Values are"
       << m1 << '\t' << m2;
  getch ( );
}
```

வெளியீடு:

```
Values before invoking swap     10     20
Calling   swap ..
20        10
Back to main... values are      10     20
```

m1, m2 ஆகிய மாறிகளின் மதிப்புகள் இடம் மாறியது, main() செயற்கூறில் பிரதிபலிக்கவில்லை. ஏன் என எண்ணிப் பார்த்தீர்களா?

செயலுருபுகள் மதிப்பு மூலமாக அனுப்பிவைக்கப்படும்போது, அழைக்கப்பட்ட செயற்கூறு, செயலுருபுகள் எந்தத் தரவினங்களைச் சார்ந்தவையோ அதே தரவினங்களில் புதிய மாறிகளை உருவாக்குகிறது. செயலுருபுகளின் மதிப்புகள் புதியதாய் உருவாக்கப்பட்ட மாறிகளில் நகலெடுக்கப்படுகின்றன. எனவே, முறையான அளபுருக்களான இந்த மாறிகளில் செய்யப்படும் மாற்றங்கள், மெய்யான அளபுருக்களில் பிரதிபலிப்பதில்லை.

மதிப்பு மூலம் அழைத்தல் முறையில், முறையான அளபுருக்களில் செய்யப்படும் மாற்றங்கள் மெய்யான அளபுருக்களில் பிரதி பலிப்பதில்லை.

4.4.2 குறிப்பு மூலம் அழைத்தல் (Call by Reference)

இந்த முறையில், அழைக்கப்படும் செயற்கூறின் செயலுருபுகள் - முறையான அளபுருக்கள்- அழைக்கும் செயற்கூறிலுள்ள மெய்யான அளபுருக்களின் மாற்றுப் பெயர்களாகச் (alias) செயல்படுகின்றன. செயற்கூறு தனது சொந்தச் செயலுருபுகளின் மீது செயல்படும்போது உண்மையில், மூலத் தரவுகளின் மீதே செயல்படுகிறது என்பது இதன் பொருளாகும். Program - 4.6 - ஐ நினைவு கூர்க. குறிப்புவகை அளபுருக்களைப் பயன்படுத்தி, swap () செயற்கூறினைத் திருத்தி எழுதுவோம்.

```
//Program - 4.7
// To exchange values

# include <iostream.h>
#include <conio.h>
void swap (int &n1, int &n2)
{
  int temp;
  temp = n1;
  n1 = n2;
  n2 = temp;
  cout<<'\n'<< n1
        <<'\t'<<n2<<'\n';
}
```

```
void main ( )
{
   int m1 = 10, m2 = 20;
   clrscr();
   cout<<"\nValues before swap call"
         << '\t' << m1 << '\t' << m2;
   swap(m1,m2);
   cout<<"\n Calling swap..";
   cout<<"\n Back to main.Values are"
         << '\t' << m1 << '\t'<< m2;
   getch ( );
}
```

வெளியீடு:

```
Values before invoking swap        10      20
Calling    swap ..
20         10
Back to main... values are         20      10
```

முறையான அளபுருக்களில் செய்த மாற்றங்கள் மெய்யான அளபுருக் களில் பிரதிபலித்துள்ளன. காரணம், முறையான அளபுருக்கள் குறிப்புவகை என்பதால் அவை மெய்யான அளபுருக்களின் நினைவக இடங்களையே சுட்டு கின்றன.

கீழேயுள்ள விளக்கங்களைக காண்க:

main()	swap()
m1=10	n1=10
m2=20	n2=20
	temp

m1, m2 ஆகிய மாறிகளின் நினைவக முகவரிகள் முறையே Oxf1, Oxf4 என்க.

m1 = Oxf1 = 10

m2 = Oxf4 = 20

முறையான அளபுருக்களின் குறிப்பு (reference) என்பதை இவ்வாறு விளக்கலாம்:

m1 = 10;

n1 என்பது, m1-ன் குறிப்பு என்பதை,

int &n1 = m1;

எனக் குறிப்பிடலாம். இதன் பொருள், m1-ன் மாற்றுப்பெயர் n1 என்பதாகும். அதாவது, m1, n1 இரண்டுமே ஒரே நினைவக இருப்பிடத்தையே குறிக்கின்றன. எனவே, மேற்கண்ட கூற்றுகளின் பொருள்,

n1 = m1 = Oxf1 = 10
n2 = m2 = Oxf4 = 20

எனக் கொள்ளலாம்.

முகவரி	**மாற்றத்துக்கு முன்**	**மாற்றத்துக்குப் பின்**
Oxf1 (n1, m1)	10	20
Oxf4 (n2, m2)	20	10

ஆக, முறையான அளபுருக்களில் செய்யப்படும் மாற்றங்கள் மெய்யான அளபுருக்களில் எதிரொலிக்கும்.

> குறிப்பு மூலம் அழைத்தல் முறையில், முறையான அளபுருக்களில் செய்யப்படும் எந்த மாற்றமும் மெய்யான அளபுருக்களில் பிரதி பலிக்கும்.

கீழேயுள்ள நிரலை இயக்கிப் பார்க்கவும்:

```
// Reference variables
// Program -4.8
# include <iostream.h>
# include <conio.h>
void main ( )
{
    int num1 = 10, & num2 = num1;
    num2 ++;
    cout << num1;
}
```

இந்த நிரலின் வெளியீடு 11 எனக் கிடைக்கும்.

குறிப்பின் உதவியால் num1, num2 ஆகிய இரண்டு மாறிகளும் ஒரே நினைவக இருப்பிடத்தைச் சுட்டுமாறு செய்ய முடிகிறது. எனவே, num1- ல் செய்யப்படும் மாற்றம் num2- ல் பிரதிபலிக்கிறது.

மெய்யான அளபுருக்களுக்கான விதிமுறைகள்

1. மதிப்பு வகையிலான முறையான அளபுருக்களுக்கு மாறிலி, மாறி அல்லது கோவையின் வடிவில் மெய்யான அளபுருக்களை அனுப்பிவைக்க முடியும்.

   எடுத்துக்காட்டாக,

   int add (int n1, int n2) என்ற முன்வடிவைக் கொண்ட ஒரு செயற்கூறினுக்கான அழைப்புக் கூற்று இவ்வாறு அமையலாம்:

   x = add (5, 10);
   x = add (a1, a2) [a1, a2 ஆகியவை int மாறிகள்]

2. குறிப்பு வகையிலான முறையான அளபுருக்களுக்கு, மெய்யான அளபுருக்களை மாறிகளின் வடிவிலேயே அனுப்பிவைக்க முடியும்.

   எடுத்துக்காட்டாக,

   int add(int &n1, int &n2) என்ற முன்வடிவு கொண்ட ஒரு செயற்கூறினுக்கான அழைப்புக்கூற்று,

   x= add (a1, b1) (a1,b1 ஆகியவை int மாறிகள்) என்று அமைய வேண்டும்.

   x = add (a1+b1, a1);
   x = add (5,10) ;

   என்றெல்லாம் அமைவது பிழையாகும்.

```
// Program - 4.9
//To print 5 stars per row and 5 such rows
# include <iostream.h>
# include <conio.h>

void fun_starts(int &i)
{
   int j = 5;
   for (i= 1; i <= j; i++)
      cout << ' ' <<'*';
}

void main()
{
   int mi = 1;
   clrscr( );
   for (; mi<= 5; mi++)
   {
      cout << '\n';
      fun_starts (mi);
   }
   getch();
}
```

மேற்கண்ட நிரல், குறிப்புரையில் குறிப்பிட்டுள்ளபடி வெளியீட்டைத் தராத காரணம் என்னவென்று புரிகிறதா? இந்த நிரல்,

* * * * *

என்று ஒரு வரி விடையை மட்டுமே தரும். ஐந்து வரி விடையைத் தராது.

காரணம்: i என்னும் மாறி mi என்ற மாறியின் குறிப்பு (reference) ஆகும். முதல் முறை செயற்கூறு நிறைவேற்றப்படும்போது i-ன் மதிப்பு 6 ஆகிவிடுகிறது. அதேநேரம் mi- ன் மதிப்பும் 6 ஆகிவிடும். எனவே, main() -ல் உள்ள for( ) மடக்கு ஒரேயொரு முறை மட்டுமே செயல்படுகிறது.

4.3.4 முன்னியல்புச் செயலுருபுகள் (Default Parameters)

சி++ மொழியில் ஒரு செயற்கூறின் முன்வடிவில் முறையான அளபுருக்களில் முன்னியல்பு மதிப்புகளை இருத்திவைக்க முடியும்.

எடுத்துக்காட்டு:

```
// Program - 4.10
// formal parameters with default values
# include <iostream.h>
# include <conio.h>
float power (float n, int p = 1)
{
  float prd = 1;
  for (int i = 1;  i<= p; i++)
      prd *= n;
  return prd;
}

void main ( )
{
   clrscr ( );
   int x = 4, b = 2;
   cout << "\n Call statement is power(b, x)..."
         << power (b, x);
   cout << "\n Call statement is power(b).. "
         << power (b);
   getch ( );
}
```

வெளியீடு:

Call statement is power (b,x) ... 16
Call statement is power(b) ... 2

power(b, x) என்னும் அழைப்புக் கூற்றில் தொடக்க மதிப்பிருத்தல் இவ்வாறு இருக்கும்:

n = b, p = x

power (b) என்னும் இரண்டாவது அழைப்புக் கூற்றில், n என்ற மாறியில் 2 என்னும் மதிப்பு இருத்தப்படும். மெய்யான அளபுரு இல்லையென்பதால் p என்னும் மாறியில் முன்னியல்பு மதிப்பான 1 இருக்கும்.

குறிப்பு:

- ✓ மாறியில் தொடக்க மதிப்பிருத்தும் வடிவில் முன்னியல்பு மதிப்பு தரப்பட்டுள்ளது.
- ✓ ஒரு செயற்கூறின் அழைப்புக் கூற்றில் சில அளபுருக்களை விட்டு விடவோ அல்லது அளபுருக்கள் இல்லாமலே அழைக்கவோ 'முன்னியல்புச் செயலுருபுகள்' வழி வகுக்கின்றன.
- ✓ முன்னியல்பு மதிப்புகள் செயற்கூறு முன்வடிவில் செயலுருபுப் பட்டியலில் இறுதியாகவே இடம்பெற வேண்டும். பட்டியலின் தொடக்கத்திலோ, நடுவிலோ இடம்பெறக் கூடாது.

கீழேயுள்ள நிரலை இயக்கிப் பார்க்கவும்:

```
//Program - 4.11
# include <iostream h>
# include <conio.h>
int area (int side1 = 10, int side2=20)
{
   return (side1 * side2);
}

void main( )
{
    int s1 = 4, s2 = 6;
    clrscr( );
    cout << area (s1, s2) << '\n';
    cout << area (s1) << '\n';
    cout << area (s2) << '\n';
    getch( );
}
```

```
வெளியீடு:
 24
 80
120
மாறியில் மதிப்பிருத்தல்:
முதல் அமைப்பில்:
side1 = s1, side2 = s2
2வது அமைப்பில்:
side1 = s1, side2 = 20
3-வது அமைப்பில்:
side1 = s2, side2 = 20
```

கீழ்க்காணும் நிரலின் வெளியீடு என்னவாக இருக்கும்?

```
// Program - 4.12
// arguments with default values

# include <iostream.h>
# include <conio.h>

void print (int times, char ch = ' * ')
{
   cout << '\n';
   for (int i = 1, i < = timers; i ++)
       cout << ch;
}
void main ( )
{
    clrscr ( );
     print (50);
     print ('A', 97);
     print ( );
}
```

விடை:

print (50)	-	times என்னும் செயலுருபு 50 என்னும் மதிப்பை ஏற்கும். எனவே, 50 முறை * வெளியிடப்படும்.
print ('A', 97)	-	times என்னும் மாறியில் 'A' என்னும் மதிப்பு இருத்தப் படும். (char மதிப்பு int மதிப்பாக உள்ளுறை இனமாற்றம் நடைபெறும்). எனவே times-ல் 65 என்னும் மதிப்பு இருக்கும்.

97 என்னும் மதிப்பு ch-ல் இருத்தப்படும். இங்கே, int மதிப்பு char மதிப்பாக இனமாற்றம் பெறும். எனவே ch-ல் ‘a’ இருத்தப்படும்.

மெய்யான அளபுருக்கள், முறையான அளபுருக்களுடன் ஒன்றுக்கு ஒன்று என்ற அடிப்படையில் பொருத்தப்படும்.
எனவே ‘a’ என்ற எழுத்து 65 முறை காட்டப்படும்.

print() - மெய்யான செயலுருபுகள் இல்லையென்பதால் முறையான அளபுருக்கள் முன்னியல்பு (default) மதிப்புகளை எடுத்துக் கொள்ளும். எனவே, * குறி, 50 முறை காட்டப்படும்.

4.5 மதிப்பினைத் திருப்பியனுப்புதல்

எந்த மதிப்பையும் திருப்பியனுப்பாத செயற்கூறு void என அறிவிக்கப் படுகிறது. ஒரு செயற்கூறினை அறிவிக்கும் போது அதன் தரவினம் வெளிப் படையாக குறிப்பிடப்படவில்லையெனில், அச்செயற்கூறின் இனம் int எனக் கொள்ளப்படும். எடுத்துக்காட்டாக,

```
int add (int, int);
add (int, int);
```

இரண்டு முன்வடிவுகளிலும் திருப்பியனுப்பும் மதிப்பு int ஆகும். காரணம் சி++ மொழியில் முன்னியல்பாக ஒரு செயல்கூறு திருப்பியனுப்பும் தரவினம் int ஆகும்.

கீழ்க்காணும் அட்டவணையைக் காண்க:

வரிசை எண்	**செயற்கூறு முன்வடிவு**	**திருப்பியனுப்பும் இனம்**
1	float power (float, int)	float
2	char choice ( )	char
3	char * success ( )	pointer to character
4	double fact (int)	double

4.5.1 குறிப்பு மூலம் திருப்பி அனுப்புதல்

குறிப்பு அல்லது மாற்றுப்பெயர் மாறிகள்:

```
// Program -  4.13
# include <iostream.h>
# include <conio.h>
void main ( )
{    int i = 5;
     int  &count = i ;
     cout << "\nCount: " << count;
     count ++;
     cout << "\ni: " << i;
     getch ( );
}
```

வெளியீடு:

```
Count : 5
i:6
```

count மாறி, 5 என்னும் மதிப்பை எப்படிப் பெற்றது?

count மாறியின் மதிப்பை மிகுக்கும்போது i-ன் மதிப்பு எவ்வாறு அதிகரித்தது? count மற்றும் i ஆகிய மாறிகள் நினைவகத்தில் ஒரே தரவினையே சுட்டிக்கொண்டுள்ளன என்பதே இதற்குக் காரணம். இதேபோல, மூல மாறியில் ஏற்படும் மாற்றம் குறிப்பு மாறியில் எதிரொலிக்கும்.

இந்தக் கோட்பாட்டின் அடிப்படையில், கீழ்க்காணும் நிரலின் வெளியீடு என்னவாக இருக்கும் என்பதைக் கண்டறிக:

```
// Program - 4.14
# include <iostream h>
# include <conio.h>
int &maxref (int &a, int &b)
{
   if (a>b)
       return a;
   else
       return b;
}
void main ( )
{ int x = 20, y = 30, max = 0;
     max = maxref (x,y);
      cout << "/n Maximum is:"<< max;
}
```

வெளியீடு:

Maximum is : 30

மேற்கண்ட நிரலில், maxref () என்னும் செயற்கூறு ஓர் int இன மாறிக்குரிய குறிப்பினைத் (reference) திருப்பியனுப்புகிறது. maxref(x,y) என்னும் செயற்கூறு அழைப்பு, a, b ஆகியவற்றில் எது பெரிதோ அதன் குறிப்பைத் திருப்பியனுப்பும். எனவே, max என்னும் மாறி y-ன் மதிப்பைப் பெறுகிறது.

கீழ்க்காணும் நிரலின் வெளியீடு என்னவாக இருக்கும்?

```
// Program 4.15
# include <iostream h>
# include <conio.h>
int & maxref (int &a, int &b)
{
     if (a > b),
        return a;
    else
        return b;
}
void main ( )
{
    int x = 20, y = 30, max = 0;
    maxref (x,y) = -1;
    cout << "\n Value of x is : " << x;
    cout << "\n Value of y is: " <<y;
    getch ( );
}
```

வெளியீடு:

Value of x is : 20
Value of y is : -1

குறிப்பு:

1. ஒரு குறிப்பினைத் திருபியனுப்பும் செயற்கூறின் அழைப்புக் கூற்று ஒரு மதிப்பிருத்தும் கூற்றின் இடப்பக்கம் இடம்பெற முடியும்.

2. மேற்கண்ட எடுத்துக்காட்டில், y-ல் -1 இருத்தப்படும். காரணம், maxref() செயற்கூறு அழைப்பு, b என்னும் மாறிக்கான குறிப்பைத் திருப்பியனுப்பும் main() செயற்கூறில் b-க்கு இணையான மாறி y ஆகும். எனவே, மதிப்பிருத்து கூற்றின் இடப்பக்கம் b இடம்பெறும். வலப்பக்கம் -1 உள்ளது.

3. ஒரு குறிப்பு இனச் செயற்கூறின் முறையான அளபுருக்கள் எப்போதுமே குறிப்பு இன அளபுருக்களாகவே இருக்க வேண்டும். அதாவது,

   int & maxref (int a, int b);

   என வரையறுப்பின், நிரல்பெயர்ப்பில் பிழை கிட்டும். காரணம் a, b ஆகிய மாறிகளின் செயல்பாட்டெல்லை maxref () செயற்கூறினுக்குள் மட்டுமே.

4.6 inline செயற்கூறுகள்

செயற்கூறுகளினால் ஏற்படும் நன்மைகளை ஏற்கெனவே பட்டிய லிட்டுள்ளோம்:

✓ நிரல் குறிமுறையின் மறுபயனாக்கம், நினைவகத்தைச் சிக்கனமாகப் பயன்படுத்த உதவும்; நிரலின் நீளத்தைக் குறைக்கும்.

இத்தகைய நன்மைகள் ஒருபுறமிருக்க, ஒரு செயற்கூறினை அழைக்கும் கூற்று, நிரல்பெயர்ப்பியை செயற்கூறு வரையறுக்கப்பட்டுள்ள இடத்துக்குச் தாவச் செய்து, பிறகு அழைப்புக் கூற்றுக்கு அடுத்திருக்கும் ஆணைக்குத் தாவச் செய்கிறது. இது செவ்வனே நிறைவேற அடுக்கங்களை (stacks) உரு வாக்கிக் கையாள வேண்டிய கூடுதல் வேலை நிரல்பெயர்ப்பிக்கு உள்ளது. செயல்கூறு அழைப்பு, திருப்பியனுப்பல், செயலுருபுகள் ஆகியவை தொடர்பான தனிச்சிறப்பு ஆணைகள் சிலவற்றைச் சேமித்து வைப்பதற்கு இத்தகைய அடுக்கங்கள் தேவைப்படுகின்றன. இதனால் நிரலின் செயல்படு வேகம் குறைகிறது. எனவே சில சூழ்நிலைகளில், குறிப்பாகச் செயற்கூறு சிறிதாக (குறைந்த எண்ணிக்கையிலான கட்டளைகள்) இருக்கும்போது,

நிரல்பெயர்ப்பியானது செயற்கூறு அழைப்புக்கூற்று இருக்குமிடத்தில், அந்தச் செயற்கூறின் கட்டளைகளை நிரப்பிப் பதிலீடு செய்துவிடும். இந்த வசதி, ***செயற்கூறினைப் பதிலிடல்*** எனப்படுகிறது. அத்தகைய செயற்கூறுகள் inline செயற்கூறுகள் எனப்படுகின்றன.

> inline செயற்கூறு மூல நிரலில் சாதாரணச் செயற்கூறு போன்றே தோற்றமளிக்கும். ஆனால் நிரல்பெயர்ப்பின்போது (compiling time) செயற்கூறின் கட்டளைகள் முழுமையும் அழைப்புக் கூற்றுக்குப் பதிலாக அப்படியே நிரலில் செருகப்பட்டுவிடும். inline செயற் கூறுகள் வேகமாகச் செயல்படும், ஆனால் அதிக நினைவக இடத்தை எடுத்துக்கொள்ளும்.

கீழேயுள்ள எடுத்துக்காட்டுகளைக் காண்க:

```
// Program - 4.16
// inline functions

# include <iostream.h>
# include <conio.h>

inline float convert_feet(int x)
{
  return x * 12;
}

void main()
{
  clrscr();
  int inches = 45;
  cout << convert_feet(inches);
  getch();
}
```

```
// working of Program - 4.16
// inline functions

# include <iostream.h>
# include <conio.h>

void main()
{
    clrscr();
    int inches = 45;
    cout << inches * 12;
    getch();
}
```

மேற்கண்ட எடுத்துக்காட்டில் உள்ளபடி, convert_feet (inches) என்னும் அழைப்புக்கூற்று, return கூற்றில் உள்ள (inches * 12) என்னும் கோவையால் பதிலீடு செய்யப்படும்.

ஒரு செயற்கூறினை inline செயற்கூறாக அறிவிக்க வேண்டும் எனில், Program-4.16- ல் காட்டியுள்ளபடி செயற்கூறின் தலைப்பில் inline என்னும் சிறப்புச் சொல்லை சேர்த்துக்கொள்ள வேண்டும்.

குறிப்பு: inline என்னும் சிறப்புச்சொல், நிரல்பெயர்ப்பிக்குத் தரப்படும் கோரிக்கை ஆகும். சில வேளைகளில் நிரல்பெயர்ப்பி, இந்தக் கோரிக்கையைப் புறக்கணித்துவிட்டுச் சாதாரண செயற்கூறாகவே கருதிக்கொள்ளும்.

4.7 மாறிகளின் வரையெல்லை (scope) விதிமுறைகள்

'வரையெல்லை' என்பது ஒரு மாறியின் அணுகியல்பைக் (accessibility) குறிக்கிறது. சி++ மொழியில் நான்கு வகையான வரையெல்லைகள் உள்ளன. அவை:

1. உள்ளமை வரையெல்லை (Local scope)
2. செயல்கூறு வரையெல்லை (Function scope)
3. கோப்பு வரையெல்லை (File scope)
4. இனக்குழு வரையெல்லை (Class scope)

4.7.1 உள்ளமை வரையெல்லை (Local Scope)

```
// Program - 4.17
// to demonstrate local variable
# include < iostream.h
# include <conio.h>
void main( )
{
        int a, b;
        a = 10;
        b = 20;
        if (a > b)
       {
              int temp; // local to this if block
              temp = a;
              a = b;
              b = temp;
        }
       cout << '\n Descending order…';
       cout << '\n" <<a << '\n" <<b;
       getch( );
}
```

- உள்ளமை மாறி, ஒரு தொகுதிக்குள் (Block) வரையறுக்கப்படுகிறது.
- ஓர் உள்ளமை மாறியின் வரையெல்லை அது வரையறுக்கப் பட்டுள்ள தொகுதிக்குள் மட்டுமே.
- ஓர் உள்ளமை மாறியை அது அறிவிக்கப்பட்டுள்ள தொகுதிக்கு வெளியிலிருந்து அணுக முடியாது.

Program -4.18 உள்ளமை மாறியின் வரையெல்லையை விளக்குகிறது.

```
//Program - 4.18
# include <iostream.h>
# include <conio.h>
void main ( )
{
    int a, b;
    a = 10
    b = 20;
    if (a > b)
    {
        int temp;
        temp = a;
        a= b;
        b = temp;
    }
    cout << a << b << temp;
    getch ( );
}
```

இந்த நிரலை நிரல்பெயர்க்கும் போது, நிரல்பெயர்ப்பி ஒரு பிழை சுட்டும் செய்தியைக் காட்டும்.

Error in line no. 13

The variable temp is not accessible

உள்ளமை மாறிகளின் வாழ்நாள் அவை அமைந்துள்ள கட்டளைத் தொகுதி செயல்பட்டு முடியும்வரை மட்டுமே. அவற்றின் கட்டளைத் தொகுதிகள் செயல்பட்டு முடிந்தவுடன் உள்ளமை மாறிகள் அழிக்கப்படுகின்றன.

- உள்ளமை மாறிகள் அவை அறிவிக்கப்பட்டுள்ள கட்டளைத் தொகுதிக்கு வெளியே அறியப்படுவதில்லை. ஒரு கட்டளைத் தொகுதி என்பது நெளிவு அடைப்புக்குறிகளுக்குள் அமைந்திருக்கும்.
- உள்ளமை மாறிகள் அவை அறிவிக்கப்பட்டுள்ள கட்டளைத் தொகுதி செயல்படும்போது மட்டுமே நிலவும்.
- நிரலின் கட்டுப்பாடு ஒரு கட்டளைத் தொகுதிக்குள் நுழையும்போது, அதன் உள்ளமை மாறிகள் உருவாக்கப்படுகின்றன. வெளியேறும்போது அவை அழிக்கப்படுகின்றன.

Program- 4.19 நிரலில் உள்ளமை மாறிகளை அடையாளங் காட்டுக. அவற்றின் வரையெல்லையைக் குறிப்பிடுக.

```
// Program - 4.19
# include <iostream.h>
void main ( )
{
    int flag = 1; a = 100;
    while (flag)
    {
        int x = 200;
        if (a > x)
        {  int j;
           -------
        }
        else
        { int h;
           ------
        }
    }
}
```

உள்ளமை மாறிகள்	வரையெல்லை
1. x	x: while( ) தொகுதிக்குள்
2. j	j : if (a > x) { } தொகுதிக்குள் மட்டும்
3. k	k :else { } தொகுதிக்குள் மட்டும்

4.7.2 செயற்கூறு வரையெல்லை (Function Scope)

ஒரு செயற்கூறினுள் அறிவிக்கப்படும் மாறிகளின் வரையெல்லை, அந்த செயற்கூறின் கட்டளைத் தொகுதி மற்றும் அத்தொகுதிக்குள் அமைந்துள்ள உட்-தொகுதிகள் அனைத்துக்கும் விரிகிறது.

Program-4.19- ல் உள்ள flag என்னும் மாறியை main() செயற்கூறு முழுமை யும் அணுக முடியும். அதனுள் இருக்கும் while(), if() ஆகிய உட்-தொகுதி களுக்குள்ளும் அணுக முடியும்.

செயற்கூறு வரையெல்லை கொண்ட ஒரு மாறியின் வாழ்நாள், அந்தச் செயற்கூறுத் தொகுதியின் வாழ்நாள் வரையாகும். ஒரு செயற்கூறின் முறையான அளபுருக்களின் வரையெல்லை செயற்கூறு வரையெல்லை கொண்டவை.

4.7.3 கோப்பு வரையெல்லை (File Scope)

அனைத்துக் கட்டளைத் தொகுதிகளுக்கும் செயற்கூறுகளுக்கும் மேலாக (குறிப்பாக main() செயற்கூறினுக்கு மேலே) அறிவிக்கப்படும் மாறி, கோப்பு வரையெல்லை கொண்டதாகும். கோப்பு வரையெல்லை கொண்ட மாறியின் வரையெல்லை அந்த நிரலின் முழுமையும் விரிகிறது. அதன் வாழ்நாள் அந்த நிரல் செயல்பட்டு முடியும் வரை நீடிக்கும்.

```
// Program - 4.20
// To demonstrate the scope of a variable
// declared at file level
# include <iostream.h>
# include <conio.h>
int i = 10;
void fun()
{
   cout << i;
}

void main()
{
     cout << i;
     while (i)
     {
         ......
         ......
     }
}
```

4.7.4 வரையெல்லை செயற்குறி (Scope Operator)

வரையெல்லை செயற்குறி ஒரு மாறியின் மறைந்து கிடக்கும் வரையெல்லையை வெளிக் கொணரும். கீழேயுள்ள நிரலை நோக்குக.

```
// Program - 4.21
# include <iostream.h>
# include <conio.h>

int num = 15;

void main()
{
  clrscr();
  int num = 5;
  num = num + ::num;
  cout << num << '\t' <<++::num;
  getch();
}
```

num என்னும் மாறி main()-க்கு உள்ளும், வெளியே கோப்பு வரையெல்லை கொண்டதாயும் அறிவிக்கப்பட்டுள்ளதைக் கவனித்தீர்களா? ::num எனக் குறிப்பிடப்பட்டுள்ளதை நோக்கினீர்களா? :: என்பது வரையெல்லை தீர்மானிப்புச் செயற்குறி எனப்படுகிறது. கோப்பு மட்டத்தில் அறிவிக்கப்பட்டுள்ள மாறிகளைக் குறிப்பிட இச்செயற்குறி பயன்படுகிறது. உள்ளமை மற்றும் கோப்பு வரையெல்லை கொண்ட மாறிகள் ஒரே பெயரில் நிலவும் சூழ்நிலைகளில் இச்செயற்குறி உதவும்.

4.7.5 இனக்குழு வரையெல்லை

பாடம் 6-ல் இதுபற்றி விளக்கப்படும்.

பயிற்சி வினாக்கள்

1. கொடுக்கப்பட்ட விளக்கங்களின் அடிப்படையில் கீழ்க்காணும் செயற்கூறு முன்வடிவுகளைக் கட்டமைக்கவும்.

(அ) Procedural -fuction()

-செயலுருபுகளை ஏற்காத, எந்த மதிப்பையும் திருப்பியனுப்பாத செயற்கூறு.

விடை:

void Procedural-function (void);
அல்லது, void Procedural-function( );

(ஆ) Manupulative-function()

- ஒரு double இனச் செயலுருபை ஏற்று, int மதிப்பைத் திருப்பியனுப்பும் செயற்கூறு.

விடை

(i) int Manipulative-function(double) ; அல்லது
(ii) int Manipulative-function(double d); அல்லது
(iii) Manipulative-function (double)

(இ) fun-default()

- இரண்டு செயலுருபுகளை ஏற்கும். அவற்றுள் ஒன்று முன்னியல்பு int மதிப்பைக் கொண்டிருக்கும். மற்றது float மதிப்பாகும். எந்த மதிப்பையும் திருப்பியனுப்பாது.

விடை:

void fun-default (float, int num = 10);

(ஈ) return-reference-fun()

-இரண்டு int இனச் செயலுருபுகளை ஏற்று, int இனக் குறிப்பைத் திருப்பியனுப்பும்.

விடை:

int & char-reference-fun(int&, int&);

(உ) multi-arguments()

-இரண்டு float இனச் செயலுருபுகளை ஏற்கும். முதலாவது செயலுருபு pi-யின் மாறா மதிப்புடையது. இரண்டாவது செயலுருபு குறிப்பு இனத்தைச் சார்ந்தது. எந்த மதிப்பையும் திருப்பியனுப்பாது.

விடை:

void multi-arguments (float const pi, float & a);

2. கீழ்க்காணும் செயற்கூறு முன்வடிவுகளில் பிழைகளைச் சுட்டுக:

அ. float average (a, b);
ஆ. float prd (int a, b);
இ. int default -arg (int a=2, int b);
ஈ. int fun (int, int, double = 3.14);
உ. void strings(char[ ]);

3. கீழேயுள்ள செயற்கூறினை அழைப்பதற்குத் தேவையான அனைத்தும் கொண்ட main() செயற்கூறினை வரையறுத்துக் காட்டுக.

```
void  line (int times, char ch)
{
    cout << '\n';
    for (int i =1; i <=times; i++)
            cout << ch;
        cout << '\n';
}
```

4. இந்த நிரலில் பயன்படுத்தப்பட்டுள்ள மாறிகளின் வரையெல்லைகளைக் குறிப்பிடுக:

```
# include <iostream.h>
float a, b ; void f1 (char);
int main ( )
{
      char ch;
       ............
      {
           int i = 0;
           ...........
      }
}
void f1(char g)
{
      short x, y ;
       ...............
}
```

விடை:

a,b - கோப்பு வரையெல்லை

ch - செயற்கூறு வரையெல்லை -main()

i - அதன் தொகுதிக்குள் மட்டும்

x, y, g - செயற்கூறு வரையெல்லை - f1()

5. கீழ்க்காணும் நிரல்களிலுள்ள பிழைகளைச் சுட்டுக:

அ)

```
include <iostream.h>

xyz (int m, int n)
{
      int m = 10;
      n = m * n;
      return n;
}
void main( )
{
      cout << xyz(9, 27);
}
```

விடை:
செயற்கூறின் தொகுதிக்குள் m என்னும் மாறியை அறிவிக்க இயலாது.

ஆ.
```
# include <iostream.h>
void xyz ( );
void main ( )
{ int x = xyz ( ) ; }
void xyz ( )
{ return '\0' ; }
```

விடை:
void இனமாக அறிவிக்கப்பட்டுள்ள செயற்கூறினுள் return கட்டளை இடம்பெறக் கூடாது. எனவே, செயற்கூறு அழைப்பு, ஒரு கோவையின் பகுதியாக இடம்பெற முடியாது.

இ.
```
# include <iostream.h>
void counter (int & a)
{ ++ a;}
void main ( )
{counter (50); }
```

விடை:
முறையான அளபுரு, குறிப்பு வகையாக இருப்பதால் மெய்யான அளபுருவை மதிப்பு வடிவில் அனுப்பி வைக்க இயலாது.

6. கீழ்க்காணும் நிரல்களின் வெளியீடு என்னவாக இருக்கும்?

அ.
```
# include <iostream.h>
int val = 10;
divide (int);
void main( )
{
     int val = 5;
     val = divide (::val/val);
     cout << :: val<<val;
}
divide (int v)
{
    return v/2;
}
```

விடை : 101

ஆ.

```
# include <iostream.h>
divide (int v)
{ return v / 10;}
void main( )
{   int val = -1;
    val = divide (400) = = 40;
    cout << "\In Val." << val;
}
```

விடை : 1

(i) divide(400) என்ற அழைப்பு 40-ஐ விடையாகத் தரும்

(ii) divide(400)==40 என்பது 40==40 எனக் கொள்ளப்பட்டு, நிபந்தனை 'சரி' என்பதால் val-ன் மதிப்பு 1 ஆகும்.

இ.

```
# include <iostream.h>
int incre (int a)
{ return a++; }
void main( )
{    int x = 10;
     x = incre (x);
     cout << x;
}
```

விடை : 10

ஈ.

```
# include <iostream.h>
# include <iostream.h>
void line( )
{
     static int v = 5;
     int x = v - -;
     while (x)
     {cout << ' * '; x --;
     }
     cout << '\n';
}
void main( )
{     clrscr( );
      for (int i = 1; i < = 5; i ++)
            line( );
      getch( );
}
```

விடை :

```
* * * * *
* * * *
* * *
* *
```

உ.
```
# include <iostream.h>
first (int i)
{ return i++; }
second (int x)
{ return x —; }
void main ( )
{
    int val = 50;
    val = val * val/val
    val = second (val);
    val = first (val);
    cout << "\n Val: " << val;
}
```
விடை : Val : 50

7. நிரல்கள் எழுதுக:

(அ) float cube (int, int, int) என ஒரு செயற்கூறினை வரையறுக்கவும். cube() - ன் செயல்பாட்டைச் சோதிக்க main() செயற்கூறினை எழுதுக.

(ஆ) unsigned long int factorial(int) என ஒரு செயற்கூறினை வரையறுக்கவும். ஓர் எண்ணின் தொடர்பெருக்கல் மதிப்பு அறியும் முறை: 5-ன் தொடர்பெருக்கல் =1x2x3x4x5. factorial (n)-ன் மதிப்பை அறிய main( ) செயற்கூறினை எழுதுக.

(இ) char odd-even-check(int) என்னும் செயற்கூறினை வரையறுக்கவும். இச்செயற்கூறு, தரப்படும் எண் இரட்டைப்படை எனில் 'E' எனவும், இல்லையேல் 'O' எனவும் விடை தரவேண்டும். இச்செயற்கூறினை இயக்கி, உரிய செய்தியை வெளியிட main() செயற்கூறினை எழுதிக் காட்டுக.

(ஈ) int prime(int) என்னும் செயற்கூறினை வரையறுக்கவும். இச்செயற்கூறு, தரப்படும் எண் பகா எண் எனில் 1எனவும், இல்லையேல் -1 எனவும் விடை தரவேண்டும். இச்செயற்கூறினை இயக்கி, உரிய செய்தியை வெளியிட main() செயற்கூறினை எழுதிக் காட்டுக.

பாடம் 5

கட்டமைப்புத் தரவினம் – அணிகள்

(Structured Data Type - Arrays)

5.1. முன்னுரை

சி++ மொழியில் ***அணி*** என்பது தருவிக்கப்பட்ட தரவினம் ஆகும். ஒரே தரவினத்தைச் சேர்ந்த பல மதிப்புகளைக் கொண்டிருக்கும்.

தரவுகள் அதிகமாக இருக்கும்போது, ஒரு தரவுத் தொகுதியில் ஒவ்வோர் உறுப்பாக அணுகிச் செயலாக்குவது கடினமான பணியாக இருக்கும். எடுத்துக்காட்டாக, கீழ்க்காணும் சூழ்நிலைகளை நோக்குக:

1. கொடுக்கப்பட்ட எண்களின் தொகுதியில் பெரிய எண்ணைக் கண்டறிதல்:

அ) தொகுதியில் இரண்டு எண்கள் இருப்பின், ஒப்பீடு இவ்வாறு இருக்கும்:

```
if (a > b)
        max = a;
else
        max = b;
```

ஆ) தொகுதியில் மூன்று எண்கள் இருப்பின், ஒப்பீடு இவ்வாறு இருக்கும்:

```
if (a > b) && (a > c)
        max = a;
else if (b > c)
        max = b;
else
        max = c;
```

இ) தொகுதியில் நான்கு எண்கள் இருப்பின், ஒப்பீடு இவ்வாறு இருக்கும்:

```
if (a > b && a > c && a > d)
        max = a;
else if (b > c && b > d)
        max = b;
else if (c > d)
        max = c;
else
        max = d;
```

எண்ணிக்கை அதிகம் ஆக ஆக ஒப்பீடுகளும் அதிகமாவதைக் கவனித்தீர்களா? தரவுகளைச் செயலாக்க மேற்கண்ட வழிமுறைகளைப் பின்பற்றினால், அதிகமான தரவுகளைக் கையாள்வது எளிதாக இருக்காது என்பதில் ஐயமில்லை.

இப்போது, கீழேயுள்ள கட்டளைத் தொகுதியை நோக்குங்கள்:

```
int a [4] = { 10, 40, 30, 20}; max = 0 ; i = 0;
                for (; i < 4; i ++)
                if a [i] > max
                        max = a [i] ;
                cout << "\n The largest number is" << max;
```

மேற்கண்ட கட்டளைத் தொகுதி, கொடுக்கப்பட்ட எண்களின் பட்டியலில் பெரிய எண்ணைத் தீர்மானிக்கிறது. அதாவது, 10, 40, 30, 20 ஆகிய எண்களில் 40 என்ற எண்ணை வெளியிடுகிறது. if கூற்றை நோக்கினீர்களா? அதிகமான தரவுகளை எளிதாகக் கையாள, ஒரே தரவினத்தைச் சேர்ந்த உறுப்புகளின் தொகுதி ஓர் அணியாக அறிவிக்கப்பட்டுள்ளது.

ஆக, அணி என்பது ஒரே தரவினத்தைச் சேர்ந்த மாறிகளின் திரட்டு ஆகும். அணியின் உறுப்புகளை ஒரு பொதுப்பெயரால் குறிப்பிட இயலும்.

அணியில் இருவகை உண்டு:

1) ஒருபரிமாணம்: குறிப்பிட்ட எண்ணிக்கையில் ஒரே இன உறுப்புகளைக் கொண்டது.

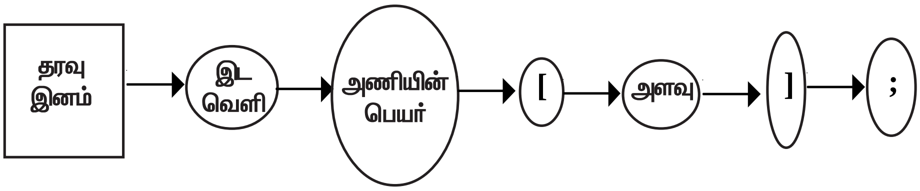
2) பலபரிமாணம்: குறிப்பிட்ட எண்ணிக்கையில், ஒருபரிமாண அணிகளை உறுப்புகளாகக் கொண்டது.

5.2. ஒருபரிமாண அணி (Single Dimensional Array)

ஒரே இனத் தரவுகளின் பட்டியல்களைக் கையாள ஒருபரிமாண அணிகள் உகந்தவை ஆகும். ஒருபரிமாண அணி இவ்வாறு அறிவிக்கப்படுகிறது:

int num _array [5];

கட்டளை அமைப்பு:



படம் 5.1 ஒருபரிமாண அணி

அணியின் உறுப்பெண்ணிக்கை எப்போதும் நேர்ம (positive) எண்ணாக இருக்கும்.

int num_array[5];

என்னும் அறிவிப்பின் பொருள், ‘num_array என்பது ஐந்து முழுஎண் மதிப்பு களைக் கொண்ட ஓர் ஒருபரிமாண அணி’ என்பதாகும். அணியின் ஒவ்வோர் உறுப்பையும் அணியின் பெயர், அணியில் உறுப்பின் இட இருப்பு (position)- இவற்றின் மூலம் அணுக முடியும். எடுத்துக்காட்டாக,

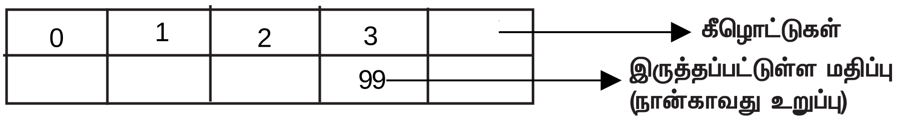
num_array[3] = 99;

என்னும் கட்டளை, அணியின் நான்காவது உறுப்பில் 99-ஐ இருத்துகிறது. num_array என்பது அணியின் பெயர், [3] என்பது கீழொட்டு (Subscript) அல்லது இட இருப்பு (position) ஆகும்.

num_array- க்கு ஒதுக்கப்பட்டுள்ள நினைவக இடம் 10 பைட்டுகள் ஆகும். காரணம் அதில் ஐந்து int உறுப்புகள் உள்ளன. (ஓர் int மதிப்புக்கு 2 பைட்டுகள் தேவை. எனவே 5 x 2 = 10 பைட்டுகள் தேவை).

நினைவக ஒதுக்கீடு இவ்வாறு இருக்கும்:

num_array - அணியின் பெயர்



அணியின் கீழொட்டு எப்போதுமே 0- வில் தொடங்கும். எனவே num_array என்னும் அணி மாறியின் கீழொட்டுகள் 0 விலிருந்து 4 வரை இருக்கும். num_array[5] என்னும் கூற்றுப் பிழையானது, காரணம் ஏற்கத்தகு கீழொட்டுகள் 0 முதல் 4 வரை மட்டுமே. அணி அறிவிப்பின் சரியான பிற வடிவங்கள்:

i) int array [100];

ii) float exponents [10];

iii) char name [30];

iv) const i = 10;
double val [i];

v) int days [ ] = { 1, 2, 3, 4, 5, 7};

iv)- வது எடுத்துக்காட்டில், val என்னும் அணியின் உறுப்பெண்ணிக்கை i என்னும் மாறிலி மூலம் குறிப்பிடப்பட்டுள்ளது. (v)- வது எடுத்துக்காட்டில், days என்னும் அணியின் உறுப்பெண்ணிக்கை 7 என்பது மறைமுகமாக உணர்த்தப்பட்டுள்ளது. எவ்வாறென ஊகிக்க முடிகிறதா? ஆம், {1, 2, 3, 4, 5, 6, 7} எனத் தொடக்க மதிப்பிருத்து கூற்றின் மூலம் தீர்மானிக்கப்படுகிறது. 7 உறுப்புகள் அணியின் அளவைத் தீர்மானிக்கிறது. இப்போது, Program - 5.1ஐ நோக்குங்கள். அணியில் நிகழ்த்தப்படும் அடிப்படைச் செயல்பாடுகளை அது விளக்குகிறது.

அணியைக் கையாள்வதற்கான எடுத்துக்காட்டுகள்:

cin >> number[4]	-	அணியின் ஐந்தாவது உறுப்புக்கு மதிப்பைப் பெற்றுக் கொள்கிறது.
cout<< number[subscript]	-	subscript- இல் குறிப்பிடப்படும் உறுப்பினைத் திரையில் காட்டும்.
number[3] = number[2]	-	மூன்றாவது உறுப்பின் உள்ளடக்கத்தை அணியின் நான்காவது உறுப்பில் இருத்தும்.
number[3] ++	-	நான்காவது உறுப்பில் இருத்தப்பட்டுள்ள மதிப்பு ஒன்று மிகுக்கப்படும்.
number [++a] = 10	-	++a - ஆல் குறிப்பிடப்படும் அணி உறுப்பில் 10 என்னும் மதிப்பு இருத்தப்படும்.

கீழேயுள்ள நிரல்களை இயக்கி வெளியீட்டை நோக்குக:

```
// Program - 5.1
// arrays and basic manipulations on
// arrays
# include <iostream.h>
# include <conio.h>

int a[5],ctr = 0, sum;

void main()
{
     for(;ctr<5;ctr++)
     {
         cout << "\nEnter value ..";
         cin >> a[ctr];
     }
    // generating sum of all the elements
    // stored in array
    for(ctr = 0, sum = 0; ctr<5;ctr++)
        sum+= a[ctr];
    clrscr();
    // display values stored in array
    for(ctr = 0; ctr < 5; ctr++)
        cout <<'\t' << a[ctr];
    cout << "\nSum of the elements ..." << sum;
    getch();
}
```

```
// try out
// Program – 5.2

# include <iostream.h>

# include <conio.h>

char ch [ ] = {'a', 'b', 'c', 'd', 'e', 'f'};
void main ( )
{
     for (int i = 0; i < 5; i++)
          cout << ch[ i];

    for (j=4; j>=0; j—)
          cout << ch [j];

    getch();
}
```

வெளியீடு:

abcdeffedcba

```
// try out
// Program - 5.3
# include <iostream.h>
# include <conio.h>

void main ( )
{
      int even [3] = {0, 2, 4}; int reverse [3];
      for (int i=0, int j = 2; i<3; i++, j —)
              reverse [j] = even [i];
      clrscr ( );
       for (i=0; i<3, i++)
          cout '\t' << even [i] << '\t' << reverse [i] << '\n';
       getch ( );
}
```

வெளியீடு:

0	4
1	2
4	0

```
// try out
//Program - 5.4
# include <iostream.h>
# include <conio.h>
void main ( )
{
      int x[5] = {1,2,3,4,5}, y [5] = {5,4,3,2,1},
     result [5] = { 0,0,0,0,0 };
      int i= 0;
      while (i++ < 5)
          result [i] = x [i] - y [i];
      clrscr ( );
      cout << "\n The contents of the array are: \n";
      i = 0;
      do
      {
            cout << '\t' << x [i]
                       << '\t' << y [i]
                       << '\t' <<  result [i]<<'\n';
           i++;
     } while (i<5);
     getch ( );
}
```

வெளியீடு:

The contents of the array are:

1	-1	0
2	4	-2
3	3	0
4	2	2
5	1	4

```
//Try out
//Program - 5.5
# include <iostream.h>
# include <conio.h>
void main()
{
   int vector[] = {2, 4, 8, 10, 0};
   for(int i=4; i>2; i—)
       vector [i]= vector[i-1];
   clrscr();
   cout << "\n Elements of array before insertion \n";
   for (i= 0; i<5; i++)
      cout << vector[i];
   vector [2] = 6;
   cout << "\n Elements after insertion \n";
   for (i= 0; i<5; i++)
      cout << vector[i];
   getch();
}
```

வெளியீடு:

Elements of array before insertion
248810
Elements after insertion
246810

ஓர் அணியிலுள்ள தரவுகளை ஏறுமுகமாக அல்லது இறங்குமுகமாக வரிசைப்படுத்த முடியும். இச்செயல்பாடு ***வரிசையாக்கம்*** (sorting) எனப்படுகிறது.

5.3 சரங்கள் (Strings)

சரங்கள், மதிப்புருக்கள் (literals) எனவும் அழைக்கப்படுகின்றன. அவை ஒருபரிமாண char அணியாகக் கருதப்படுகின்றன. எண்வகை அணிகளை அறிவிப்பது போன்றே சரங்களை அறிவிக்க முடியும். எடுத்துக்காட்டாக,

(i) char name [10];

(ii) char vowels [ ] = { ‘a’, ‘e’, ‘i’, ‘o’, ‘u’};

(iii) char rainbow[ ] = “VIBGYOR”;

ஒரு char அணியை (ii), (iii) ஆகியவற்றில் உள்ளதுபோலத் தொடக்க மதிப்பிருத்தி அறிவிக்கலாம். (ii)- ல் உள்ளது போன்ற ஒரு char அணியைச் சரமாகக் கையாள வேண்டுமெனில் அதன் இறுதி உறுப்பாக '\0'(NULL) குறியுருவைக் குறிப்பிட வேண்டும்.

```
// Program - 5.6
// Reading values into an array of characters

# include <iostream.h>
# include <stdio.h>
# include <conio.h>
# include <string.h>
void main()
{
   clrscr();
   char name [30], address [30], pincode[7];
   cout << "\n Enter name ...";
   cin >> name;
   cout << "\n Enter address...";
   gets (address);
   cout << "\n Enter pincode ...";
   cin.getlin e (pincode, 7,'#');
   clrscr();
   cout << "\n Hello  " << name;
   cout << "\n Your address is ..." << address;
   cout << "\n Pincode  ....";
   cout.write(pincode, sizeof(pincode));
   getch();
}
```

மேற்கண்ட எடுத்துக்காட்டில் (Program - 5.6), name, address, pincode ஆகிய மாறிகளுக்கான மதிப்புகள் cin, gets(), getline() ஆகியவற்றின் மூலம் பெறப் படுகின்றன. cin, வெற்று இடவெளி அல்லது நகர்த்தி திரும்பலை (Carriage Return - Enter Key) சரத்தின் ஈற்றாக எடுத்துக் கொள்ளும்.

எடுத்துக்காட்டாக,

cin >> name;

அ) உள்ளீடாக K V N Perumal எனக் கொடுத்தால், name- ல் K மட்டுமே இருத்தப்பட்டிருக்கும். காரணம் K- ஐ அடுத்து ஓர் இடவெளி விடப்பட்டதால் அதுவே சரத்தின் ஈற்றாக எடுத்துக் கொள்ளப்படும்.

ஆ) name- ன் மதிப்பாக K.V.N.Perumal என உள்ளீடு தந்தால், name-ல் K.V.N.Perumal என்ற முழுப்பெயரும் இருத்தப்படும்.

சர மதிப்பின் அங்கமாக இடவெளிகளையும் எடுத்துக்கொள்ள வேண்டுமெனில், stdio.h என்னும் கோப்பில் வரையறுக்கப்பட்டுள்ள gets() என்னும் செயல்கூறினைப் பயன்படுத்த வேண்டும். அல்லது அடிப்படை உள்ளீட்டுத் தாரை istream- ன் உறுப்புச் செயற்கூறான getline()- ஐப் பயன்படுத்த வேண்டும்.

கட்டளையமைப்பு:

gets() : gets(char அணியின் பெயர்) அல்லது gets (char *)

getline() : cin.getline(char *, எழுத்துகளின் எண்ணிக்கை, வரம்புக்குறி);

சரத்தின் உள்ளடக்கத்தைத் திரையில் காட்ட இரண்டு வழிமுறைகள் உள்ளன:

1. cout <<name; - பிற மாறிகளின் மதிப்பை வெளியிடுவதைப் போன்றதே

2. cout.write (pincode, 7); அல்லது cout.write (pincode, sizeof(pincode));

write() என்பது அடிப்படை வெளியீட்டுத் தாரை அதாவது ostream- ன் உறுப்புச் செயற்கூறாகும். ஓர் இனக்குழுவின் அனைத்து உறுப்புச் செயற்கூறுகளையும், அந்த இனக்குழுவில் உருவாக்கப்பட்ட ஓர் இனப்பொருளின் (object/instance) மூலமாக அணுக வேண்டும். write() செயற்கூறினுக்குத் தேவையான இரண்டு அளபுருக்கள் சரத்தின் பெயர், காட்டப்பட வேண்டிய எழுத்துகளின் எண்ணிக்கை. எடுத்துக்காட்டு:

```
//Program - 5.7
# include <iostream.h>
# include <conio.h>
void main()
{
   clrscr();
   char name[] = "Tendulkar";
   int i=1;
   while (i<10)
   {
       cout.write (name, i);
       cout << '\n';
       i++;
   }
   getch();
}
```

வெளியீடு:

T
Te
Ten
Tend
Tendu
Tendul
Tendulk
Tendulka
Tendulkar

string.h கோப்பில் வரையறுக்கப்பட்டுள்ள, சரங்களைக் கையாளும் செயற்கூறுகளின் விளக்கத்தை அட்டவணை 5.1-ல் காண்க:

வரிசை எண்	செயற்கூறு	கட்டளை அமைப்பு	பயன்பாடும் திருப்பியனுப்பும் மதிப்பும்
1	strlen()	strlen(char*)	சரத்திலுள்ள எழுத்துகளின் எண்ணிக்கையைத் தரும். (எ.டு): char name[ ] = “Lasya”; strlen(name),5 என்ற விடையைத் தரும்.
2	strcpy()	strcpy(char *, char*)	மூலச் சரத்தை இலக்குச் சரத்தில் படியெடுக்கும் (எ.டு) strcpy(name, petname)
3	strcmp()	strcmp(char*, char*)	இரண்டு சரங்களை ஒப்பிட்டு, இரண்டு சரங்களும் நிகர் எனில், 0 என்ற விடையைத் தரும். அகர வரிசைப்படி முதல்சரம் இரண்டாவது சரத்துக்கு முன்பு இருப்பின்1என்னும் விடையைத் தரும். பின்பு இருப்பின் -1 என்ற விடை கிடைக்கும். (எ-டு) strcmp (“Abc”, “Abc”) என்பது 0 என்னும் விடை தரும். strcmp (“Abc”, “abc”) என்பது -1 என்னும் விடைதரும். strcmp (“abc”, “Abc”) என்பது 1 என்னும் விடை தரும்.

அட்டவணை 5.1. சரச் செயற்கூறுகள்

சரங்களை ஓர் ஒருபரிமாண char அணி போலவே ஒவ்வோர் எழுத்தாக எடுத்தாள முடியும். எடுத்துக்காட்டு:

```
// Program - 5.8
# include <iostream.h>
# include <conio.h>
void main()
{
    clrscr();
    char name[] = "Pascal", reverse[7];
    int i= 0;
    while (name[i] != '\0')
        i++;
    reverse[i] = '\0';
    —i;
    int j = 0;
    while (i>= 0)
        reverse [i—] = name [j++];
    cout << "\n The contents of the string are: "<< name;
    cout << "\n The contents of the reversed string ..."
         << reverse;
    getch();
}
```

கீழ்க்காணும் நிரலின் வெளியீடு என்னவாக இருக்கும்?

```
//Program - 5.9
# include <iostream.h>
# include <conio.h>
# include <string.h>
main()
{
   char word[] = "test";
   int i=0, k = 4, j = 0;
   clrscr();
   while (i < 4)
   {
       j = 0;
       while (j<=i)
          cout << word [j++];
       cout << '\n';
       i++;
   }
   return 0;
}
```

5.4. இருபரிமாண அணி (Two-Dimensional Array)

இருபரிமாண அணி என்பது, ஒருபரிமாண அணிகளை உறுப்புகளாகக் கொண்ட ஓர் அணியாகும். எடுத்துக்காட்டாக, marks[3] [4] என்னும் அணி, 3 கிடக்கைகள், 4 நெடுக்கைகள் கொண்ட ஓர் அட்டவணையைக் குறிக்கும்.

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3

int marks[3] [4] = {90, 70, 80, 0, 75, 95, 65, 0, 80, 90, 90, 0};

என்னும் இருபரிமாண அணி,

	0	1	2	3
0	90	70	80	0
1	75	95	65	0
2	80	90	90	0

என்னும் அட்டவணையை உருவாக்கும்.

குறிப்பு:

- √ ஓர் இருபரிமாண அணியின் உறுப்பெண்ணிக்கை, கிடக்கை/நெடுக்கை எண்ணிக்கைகளின் பெருக்குத் தொகையாகும். மேலே கண்ட எடுத்துக்காட்டில் marks என்னும் அணி 12 (3x4) உறுப்புகளைக் கொண்டுள்ளது.
- √ அணி உறுப்புகளைக் குறிக்கும் கீழொட்டுகள் (subscripts) எப்போதுமே 0-வில் தொடங்குகின்றன. கிடக்கைகளைக் குறிக்கும் கீழொட்டு 0 முதல் 2 வரையும், நெடுக்கைகளுக்கு 0 முதல் 3 வரையும் இருக்கும்.
- √ இருபரிமாண அணியின் உறுப்பு marks[கிடக்கை] [நெடுக்கை] எனக் குறிப்பிடப்படும். எடுத்துக்காட்டாக,

  marks [0] [3] = marks[0] [0] + marks[0] [1] + marks[0] [2];

  என்னும் கட்டளை, முதல் கிடக்கையிலுள்ள மதிப்பெண்களை, அதாவது 90, 70, 80 ஆகியவற்றின் கூட்டுத் தொகையைக் கணக்கிடும்.

ஓர் இருபரிமாண அணி இவ்வாறு அறிவிக்கப்படுகிறது:

< தரவினம் > < அணிப்பெயர்> [< கிடக்கை>] [< நெடுக்கை>];

எடுத்துக்காட்டு:

1. int a[3] [2] - a என்னும் அணி 3 கிடக்கைகள், 2 நெடுக்கைகள் கொண்டது.
2. cons int i = 5;
   float num [i] [3]; - num என்னும் இருபரிமாண அட்டவணை 5 கிடக்கைகள், 3 நெடுக்கைகள் கொண்டது.
3. short fine ['A'] ['E'] - fine என்னும் இருபரிமாண அட்டவணை 65 கிடக்கைகள், 69 நெடுக்கைகள் கொண்டது.

குறிப்பு:

அணியின் பரிமாணங்கள் (கிடக்கைகள்/நெடுக்கைகள்) இவ்வாறெல்லாம் இருக்க முடியும்:

1. int மாறிலிகள்
2. முழுஎண் அல்லது வரிசையெண் சார்ந்த மாறிலிக் குறிப்பெயர்கள்
3. char மாறிலிகள்
4. enum குறிப்பெயர்கள்

5.4.1. இருபரிமாண அணிகளின் நினைவக இருப்பிடம்

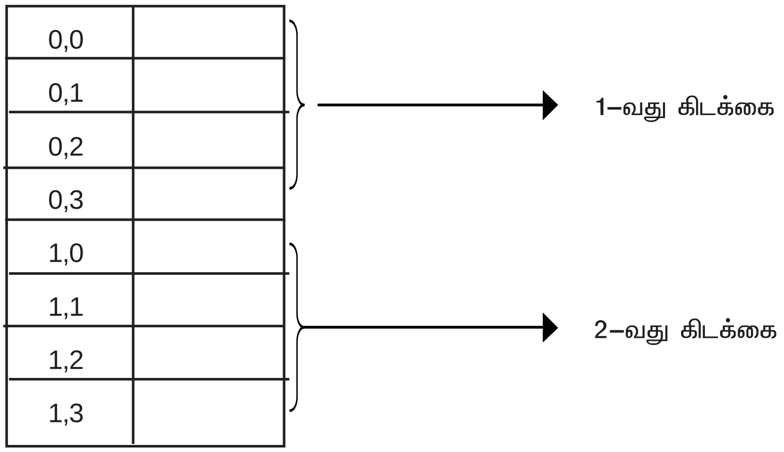
ஓர் இருபரிமாண அணியின் உறுப்புகள் நினைவகத்தில் தொடர்ச்சியான இருப்பிடங்களில் இருத்திவைக்கப்படுகின்றன. உறுப்புகளின் மதிப்புகள் கீழ்க்காணும் முறைகள் ஒன்றில் இருத்தப்படுகின்றன.

1. கிடக்கை வாரியாக: கிடக்கை - முதன்மை (row - major) என்று பெயர்
2. நெடுக்கை வாரியாக: நெடுக்கை - முதன்மை (column - major) என்று பெயர்

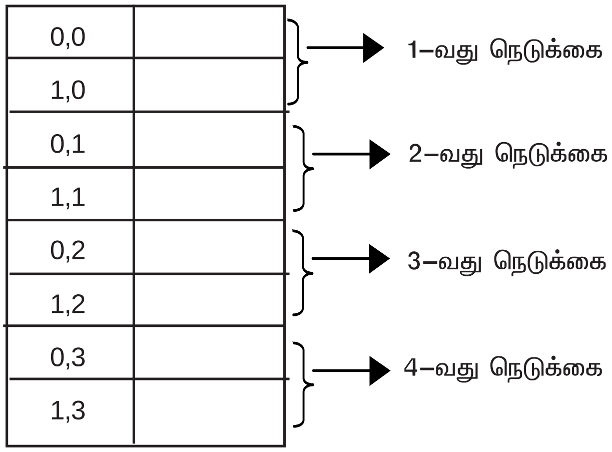
எடுத்துக்காட்டு:

int sales[2][4] என்ற அணியின் உறுப்புகள் இவ்வாறு அமைகின்றன:

கிடக்கை–முதன்மை வரிசை



நெடுக்கை – முதன்மை வரிசை



sales[1][0] என்ற உறுப்பின் இட இருப்பைக் கிடக்கை-முதன்மை வரிசையிலும், நெடுக்கை - முதன்மை வரிசையிலும் கவனித்தீர்களா?

இருபரிமாண அணியின் நினைவகக் கொள்ளளவு இவ்வாறு கணக்கிடப்படுகிறது:

உறுப்பெண்ணிக்கை x ஓர் உறுப்புக்குத் தேவையான நினைவக அளவு.

எடுத்துக்காட்டாக, int sales[2][4] என்ற அணியின் அளவு இவ்வாறு கணக்கிடப்படுகிறது:

உறுப்பெண்ணிக்கை = கிடக்கைகள் x நெடுக்கைகள் = 2 x 4 = 8

∴ 8 x 2 (ஓர் int மதிப்புக்கு 2 பைட்டுகள்)
∴ அளவு = 16 பைட்டுகள்

float num [4][6] என்ற அணியின் அளவு என்னவாக இருக்கும்?

விடை :

4 x 6 x 4 = 96 பைட்டுகள்

கீழ்க்காணும் அணியை நோக்குக:

int num[4][3] = {78, 7, 6, 4, 5, 8, 9, 7, 6, 1, 2, 3};

num	0	1	2
0	8	(7)	6
1	4	(5)	8
2	(9)	7	6
3	1	(2)	3

num அட்டவணையில் வட்டமிடப்பட்ட மதிப்புகளின் உறுப்புகளை எழுதிக் காட்டுக:

விடை:

num[0][1] - (7)

num[1][1] - (5)

num[2][0] - (9)

num[3][1] - (2)

கீழே அறிவிக்கப்பட்டுள்ள அணிகளில், உறுப்புகளின் எண்ணிக்கையைத் தீர்மானிக்கவும்:

அ) int array[10] [12]; விடை : 120 உறுப்புகள்

ஆ) int x[ ] [2] = {0, 1, 1, 2, 2, 3} விடை : 6 உறுப்புகள் (கிடக்கை-3, நெடுக்கை-2)

தொடக்க மதிப்புகளோடு அணியை அறிவிக்கும்போது, முதல் பரிமாணத்தைக் குறிப்பிடுவது கட்டாயமில்லை.

```
        #include <iostream.h>
        # include <conio.h>
        #include iomanip.h>
        void accept (int s [3] [4], int (total)
        { int r = O;  c = O;
                for (; r < 3; r ++)
                {cout << "In Month: " << r+1;
                    for (; c < 4; c++)
                        {cout << 'In' << c+1 <<
"Quarter..";
```

செயல்கூறுகளுக்கு, அணிகளைச் செயலுருபுகளாக அனுப்பிவைக்க முடியும். அணியின் பெயரை மட்டும் மெய்யான அளபுருவாகக் குறிப்பிட்டால் போதும். பரிமாணங்களைக் குறிப்பிடத் தேவையில்லை.

அணி அளபுருக்கள் இயல்பாகவே குறிப்புவகை அளபுருக்கள் போலச் செயல்படுகின்றன. காரணம், அணியின் பெயர், நினைவகத்தில் அவ்வணி இருத்தப்பட்டுள்ள பகுதியின் தொடக்க முகவரியைக் குறிக்கிறது. பிற மாறி களின் பெயர்கள் அவ்வாறில்லை. எனவே அணியின் பெயரை செயலுருபாகக் குறிப்பிடுவது, ஒரு முகவரியை முறையான அளபுருவுக்கு அனுப்பி வைக்கிறோம் என்று பொருள் (குறிப்புவகை அளபுருவைப் போல).

```
// Program - 5.10
#include <iostream.h>
# include <conio.h>

void accept (int s[3][4], int &total)
{
   int r = 0, c = 0;
   for (; r < 3; r++)
   {
       cout << "\n Month: " << r+1;
       for (c = 0; c < 4; c++)
       {
            cout << '\n' << c+1 << " Quarter..";
            cin >> s[r][c];
            total += s[r][c];
       }
   }
}

void display (int d[3][4], int total)
{
   int r, c;
   clrscr ( );
   cout << "\nSales figures for 3 months & their
                              respective quarters..";
   for (r = 0; r < 3; r++)
   {
        cout << "\n Month ..." << r+1;
        for (c = 0; c < 4; c ++)
          cout << '\t' << d[r][c];
   }
   cout << "\n Total sales .." << total;
}
void main()
{
    clrscr();
    int sales[3][4], t = 0;
    accept(sales,t);
    display(sales,t);
    getch();
}
```

இப்போது கீழேயுள்ள நிரலை நோக்குக:

```
// Program - 5.11
# include <iostream.h>
# include <conio.h>
void accept(int a)
{
   cout << "\n Enter a number ..";
   cin >> a;
}

void display(int a)
{
   cout << '\n' << a;
}

void main()
{
   int num [2][2] ={{0,0},{0,0}}, r = 0, c = 0;
   clrscr ( );
   for (; r < 2; r++)
       for (; c < 2; c++)
           accept(num[r][c]);
   clrscr();
   for (r = 0; r < 2; r++ )
       for (c = 0; c < 2; c++)
           display (num[r][c]);
   getch();
}
```

accept() செயல்கூறினுக்கு உள்ளீடாக 1, 2, 3, 4 ஆகிய மதிப்புகள் தரப்படுவதாகக் கொள்க. விடை:

0
0
0
0

எனக் கிடைக்கும். num அணியின் உறுப்புகளில் 1, 2, 3, 4 ஆகிய மதிப்புகள் ஏன் இருத்தப்பட வில்லை என எண்ணிப் பார்த்தீர்களா?

இந்த எடுத்துக்காட்டில், void accept() செயல்கூறினுக்கு அளபுருவாக தனித்தனி உறுப்புகள் அனுப்பிவைக்கப் படுகின்றன. எனவே, அவை மதிப்பு வகை அளபுருவாகவே எடுத்துக் கொள்ளப்படும். குறிப்புவகை அளபுருவாகக் கருதிக் கொள்ளப்படாது.

குறிப்பு:

அணியின் பெயர் மட்டுமே அணியின் தொடக்க முகவரியைக் குறிக்கிறது.

இப்போது, மேற்கண்ட நிரலில் accept() செயல்கூறினைச் சற்றே மாற்றி எழுத வேண்டும். 1, 2, 3, 4 என அதே உள்ளீடுகளைத் தந்து, வெளியீடாக,

1
2
3
4

எனப் பெறமுடியும்.

5.4.2. அணிக்கோவை (Matrix)

அணிக்கோவை என்பது m x n எண்களை m கிடக்கைகளிலும் n நெடுக்கைகளிலும் ஒரு செவ்வக அணி வடிவில் கொண்டிருக்கும். அணிக்கோவைகளை இருபரிமாண அணிகளாக வரையறுக்க முடியும்.

Program-5.12 நிரல் இரண்டு அணிக்கோவைகளுக்கான மதிப்புகளை உள்ளீடாகப் பெற்று அவ்வணிக் கோவைகளின் நிகர்ப்பாட்டை (equality) பரிசோதித்து விடை தருகிறது.

5.5. சரங்களின் அணி

சரங்களின் அணி என்பது ஓர் இருபரிமாண char அணி ஆகும். அணி வரையறுப்பில் உள்ள முதல் சுட்டெண் (கிடக்கை எண்ணிக்கை) சரங்களின் எண்ணிக்கையைக் குறிக்கும். இரண்டாவது சுட்டெண் (நெடுக்கை எண்ணிக்கை) சரங்களின் உச்ச அளவு நீளத்தைக் குறிக்கும். எடுத்துக்காட்டு:

```
char day-names [7] [10]   =   {“Sunday”,
                               “Monday”,
                               “Tuesday”,
                               “Wednesday”,
                               “Thursday”,
                               “Friday”,
                               “Saturday”};
```

இந்த அணியின் உறுப்புகளிலுள்ள மதிப்புகள் அட்டவணை 5.1-இல் உள்ளவாறு இருத்தப்பட்டிருக்கும்.

```
//Program - 5.12
# include <iostream.h>
# include <conio.h>

void accept(int mat[3][3])
{
   clrscr();
   int r = 0, c = 0;
   for (; r < 3; r++)
   {
       cout << "\n Enter elements for row.." << r;
       for (c=0; c < 3; c++)
            cin >> mat[r][c];
   }
}

void main()
{
    int m1[3][3], m2[3][3];
    accept (m1);
    accept (m2);
    int i=0, j = 0, flag = 1;
    for (; i < 3; i++)
    {
        for (; j < 3; j ++)
            if (m1[i][j] != m2[i][j])
            {
                flag = 0;
                break;
            }

        if (flag == 0)
            break;
   }
   if (flag)
      cout << "\n The matrices are equal ...";
   else
      cout << "\n The matrices are not equal..";
  getch();
}
```

	0	1	2	3	4	5	6	7	8	9	
0	S	u	n	d	a	y	\0				day-names [0]
1	M	o	n	d	a	y	\0				day-names [1]
2	T	u	e	s	d	a	y	\0			day-names [2]
3	W	e	d	n	e	s	d	a	y	\0	day-names [3]
4	T	h	u	r	s	d	a	y	\0		day-names [4]
5	F	r	i	d	a	y	\0				day-names [5]
6	S	a	t	u	r	d	a	y	\0		day-names [6]

அட்டவணை 5.1 நினைவகத்தில் அணியின் உறுப்புகள்

அணியிலுள்ள தனித்தனிச் சரங்களைக் கையாள day-names[0] என முதல் சுட்டெண் (கிடக்கை எண்) மட்டும் குறிப்பிட்டால் போதும். ஒரு சரத்திலுள்ள ஒரு குறிப்பிட்ட எழுத்தை அதாவது அணியின் ஒரு தனி உறுப்பைக் கையாள day-names[0][5] என இரண்டு சுட்டெண்களையும் (கிடக்கை/நெடுக்கை எண்களை) குறிப்பிட வேண்டும்.

ஒவ்வொரு சரத்தின் இறுதியிலும் வெற்றுக் குறியுருவைச் (null character - '\0') சேர்ப்பது கட்டாயமில்லை (optional). நாம் சேர்க்காவிட்டாலும் சி++ நிரல்பெயர்ப்பி தானாகவே இணைத்துக் கொள்ளும்.

பயிற்சி வினாக்கள்

1. கீழ்க்காணும் கட்டளைகள் பிழைசுட்டுவது ஏன்?

அ) int a[5.5];

- அணியின் பரிமாணம் முழுஎண்ணாகவே இருக்கவேண்டும்.

ஆ) float f[3] = {1.0, 2.0};

- இது பிழை சுட்டாது. ஆனால் அணியில் குறிப்பிடப்பட்டுள்ள உறுப்பு எண்ணிக்கையைவிடக் குறைவான மதிப்புகளே தரப் பட்டுள்ளன. இவ்வாறு குறைவாக இருக்குமெனில், மீதி உறுப்புகளில் 0 மதிப்பு தாமாகவே இருத்தப்பட்டுவிடும்.

இ) float num[A];

- அணியின் பரிமாணம் வெளிப்படையாக அறிவிக்கப்பட வேண்டும். இதில், A என்னும் மாறிக்கு மதிப்பு எதுவும் கிடையாது. இக்கூற்றையே,

```
float num['A']
```

அல்லது,

```
const A=10; float num[A];
```

என மாற்றி எழுதலாம்.

ஈ) char a[3] [ ] = { "one", "two", "three"};

- தொடக்க மதிப்புடன் அறிவிக்கப்படும் ஓர் இருபரிமாண அணியில் முதல் பரிமாணத்தை விட்டுவிடலாம். இரண்டாவது பரிமாணம் கட்டாயமாக இடம்பெற வேண்டும். எனவே மேற்கண்ட கூற்றை,

```
char a[ ] [6] = {"one", "two", "three"};
```

என மாற்றி எழுதலாம்.

உ) char ch[1] = 's';

-char அணியை சரமாக அறிவிக்க வேண்டுமெனில்,

```
char ch[2] = "s";
```

என அறிவிக்கலாம். அல்லது char அணியாகவே அறிவிக்க வேண்டுமெனில்,

```
char ch[1] = {'s'};
```

என அறிவிக்க வேண்டும்.

ஊ)

```
char test [4];
test = {"abc"};
```

- ஓர் அணியில் இவ்வாறு மதிப்பிருத்த முடியாது. அறிவிக்கும்போதே தொடக்க மதிப்பிருத்தி,

```
char test[4] = "abc";
```

எனக் குறிப்பிடலாம். அல்லது,

```
char test [4];
strcpy(test, "abc");
```

எனச் சரமதிப்பை, சர மாறியில் படியெடுக்கலாம்.

எ) int num[ ] = {1,2,3}, num2[3];
num2 = num;

-அணிகளைக் குழுவாக அறிவித்து மதிப்பிருத்த வழியில்லை.

int num1[5], num2[10], num3[15];

என வெறுமனே குழுவாக அறிவிக்கலாம். தொடக்க மதிப்பிருத்தலும் செய்ய வேண்டியிருப்பின் தனித்தனியேதான் செய்ய வேண்டும்.

ஏ) int num[3];
cout << num;
cin >> num;

- இது போன்ற அணி உள்ளீட்டு/ வெளியீட்டுச் செயல்பாடுகளுக்கு அனுமதியில்லை. ஒவ்வோர் உறுப்பாகவே கையாள முடியும்.

cout << num [1];
cin >> num[2];

என அமைக்கலாம்.

ஐ) int roaster = { 1,2,3,4};

- roaster என்னும் மாறி ஒரேயொரு மதிப்பையே ஏற்கும். எனவே,

int roaster = 10; அல்லது,
int roaster [ ] = {1,2,3,4} ;

என அமைக்கலாம்.

2. கீழ்க்காணுமாறு தொடக்க மதிப்பிருத்தப்பட்டபின் அணியின் உள்ளடக்கம் என்னவாக இருக்கும்?

அ) int rate[ ] = {30, 40, 50};

ஆ) char ch[b] = {“bbbbb\0”}; -b என்பது இடவெளியைக் குறிக்கிறது.
ch[0] = ‘C’; ch[4] = ‘T’; ch[3] = ‘A’;

இ) char product-list [ ] [5] = { “Nuts”, “Bolts”, “Screw”};

விடைகள்:

அ) rate [0] = 30, rate[1] =40, rate[2] = 50

ஆ) ch[0] = ‘C’, ch[1] = ‘ ’, ch[2]= ‘ ’, ch[3] = ‘A’,
ch[4] = ‘T’, ch[5] = ‘\0’

இ) product-list[0] = “Nuts\0”, product-list[1] = “Bolts\0”,
product-list[2] = “Screw\0”.

3. கீழ்க்காணும் நிரல்களின் வெளியீடு என்னவாக இருக்கும்?

அ)

```
# include <iostream.h>
void main ( )
{
      char ch [ ] = “END”;
      cout << ch;
}
```

விடை: END

ஆ)

```
# include <iostream.h>
void main ( )
{
    int a [ ] = {1,2,3,4,5};
    for (int i = 0, i < 4, i++)
          a[i+1] = a[i];
   for (i= 0; i<5; i++)
        cout << ‘\n’ << a[i];
}
```

விடை:

அணியின் தொடக்க உள்ளடக்கம்:

a[0] = 1, a[1] = 2, a[2] = 3, a[3] = 4, a[4] = 5

முதல் for() மடக்கு செயல்படும் போது,

i = 0 : a[1] = a[0]
i = 1 : a[2] = a[1]
i = 2 : a[3] = a[2]
i = 3 : a[4] = a[3]

எனவே, வெளியீடு,

1
1
1
1
1

என அமையும்.

இ)

```
# include <iostream.h>
# include <conio.h>
void main( )
{
    char name[ ] = “Jerry”; int k=5;
    for (int i = 0;  i < 3; i ++, k —)
          name [k] = name[i];
    cout << name;
    getch( );
}
```

விடை:

for() மடக்கு செயல்படும்போது,

i = 0 : k = 5, name[5] = name[0];
i = 1 : k = 4, name[4] = name[1];
i = 2 : k = 3, name[3] = name[2];

எனவே, வெளியீடு, JerreJ எனக் கிடைக்கும்.

4. நிரல்கள் எழுதுக:

1) int - array என்னும் ஓர் int அணியை 10, 20, 30, 40, 50 ஆகிய தொடக்க மதிப்புகளுடன் வரையறுக்கவும். இந்த அணி உறுப்புகளின் மதிப்புகளைக் கூட்டி விடையைக் காட்டவும்.

2) 10 மதிப்புகளை இருத்திவைக்க கூடிய ஓர் int அணியை அறிவிக்கவும். அணியின் உறுப்புகளுக்குப் பயனரிடமிருந்து மதிப்புகளைப் பெறுக. அம்மதிப்புகளை முன்பின் வரிசையில் திரையிடுக.

3) word என்னும் சர மாறியில் (char அணி) பயனரிடமிருந்து ஒரு சொல்தொடரைப் பெற்று இருத்துக. while() மடக்கு switch() கூற்று ஆகியவை பயன்படுத்தித் தொடரிலுள்ள உயிரெழுத்துகளைக் (vowels) கணக்கிட்டு விடையை வெளியிடுக. எடுத்துக்காட்டாக,

char word[ ] = “The Vowel count AEIOU aeiou”;
Vowel count is 14

4) MATRIX[3][3] என்னும் அணிக்கோவையை உருவாக்கவும். மூலைவிட்ட உறுப்புகளின் மதிப்புகளையும் அவற்றின் கூட்டுத் தொகையையும் வெளியிடுக.

5) matrixA [4][4], matrixB [4][4] ஆகிய இரு அணிக்கோவைகளுக்கு மதிப்புகளை உள்ளீடாகப் பெறுக. இரண்டு அணிக்கோவை களையும் கூட்டி sum-matrix[4][4] என்னும் அணிக்கோவையில் இருத்தி, அதன் உள்ளடக்கத்தை வெளியிடுக.

எடுத்துக்காட்டு:

matrixA	**matrixB**	**sum-matrix**
1 2 3 4	9 8 7 6	10 10 10 10
5 6 7 8	5 4 3 2	10 10 10 10
9 1 2 3	1 9 8 7	10 10 10 10
4 5 6 7	6 5 4 3	10 10 10 10

பாடம் 6

இனக்குழுக்களும் பொருள்களும்
(Classes and Objects)

6.1 இனக்குழுக்கள் – ஒரு முன்னுரை

சி++ மொழியின் மிக முக்கியமான பண்புக்கூறு ‘இனக்குழு’ என்பதாகும். சி++ மொழியை உருவாக்கிய ஜேன் ஸ்ட்ரௌஸ்ட்ரப் தம் மொழிக்கு முதலில் ‘இனக்குழுவுடன் கூடிய சி’ (C with Classes) என்றுதான் பெயர் சூட்டினார் என்கிற தகவல், இனக்குழுவின் முக்கியத்துவத்தை எடுத்தியம்புகிறது. பயனர் வரையறுக்கும் தரவினங்களுள் ‘இனக்குழு’ என்பது, பயனர்கள் புதிய தரவினங்களை உருவாக்கவும் நடைமுறைப்படுத்தவும் ஒரு புதிய வழியைத் திறக்கிறது. வேறுபட்ட இனத்தரவுகளை ஒன்றாகச் சேர்த்துவைக்க இனக் குழுக்கள் ஒரு புதிய வழிமுறையை வழங்குகின்றன.

```
class student
{
        char name[30];
        int rollno, marks1, marks2, total_marks;
}
```

மேற்கண்ட இனக்குழுவில் அறிவிக்கப்பட்டுள்ள rollno, marks1, marks2, total_marks ஆகிய தரவு மாறிகள் student என்னும் இனத்தின் பண்புக்கூறுகளை வெளிப்படுத்துகின்றன. இந்தத் தரவுகளைக் கையாளக் கூடிய செயற்கூறுகளை வரையறுத்து, student இனக்குழுவை மேலும் விரிவுபடுத்த முடியும். இத்தகைய செயற்கூறுகளை ***வழிமுறைகள்*** (methods) என்றும் கூறுவர். காரணம், தரவு களின் மீது நிகழ்த்தக் கூடிய பல்வேறு செயல்பாடுகளை (தரவு மதிப்புகளை உள்ளீடாக பெறுதல், தரவுகள் பங்குபெறும் கணக்கீடுகள் போன்றவை) அவை வரையறை செய்கின்றன.

சுருங்கக் கூறின்,

தரவுகளையும் அவற்றோடு தொடர்புடைய செயற்கூறுகளையும் ஒன்றாகச் சேர்த்துவைக்க இனக்குழு வகை செய்கிறது.

6.2 இனக்குழுவை வரையறுத்தல்

ஓர் இனக்குழுவின் வரையறுப்பு இருபகுதிகளைக் கொண்டது:

1) இனக்குழு அறிவிப்பு

2) இனக்குழுவின் செயற்கூறு வரையறைகள்

```
// Program - 6.1
# include <iostream.h>
# include <conio.h>
class student
{
       private :
         char name[30];
         int rollno, marks1, marks2 ,total_marks;
       protected:
          void accept()
          {
              cout<<"\n Enter data name, roll no, marks 1 and
                                                  marks 2.. ";
              cin>>name>>rollno>>marks1>>marks2;
          }
          void compute()
          {
              total_marks = marks1+ marks2;
          }
          void display()
          {
                     cout<<"\n Name "<<name;
                     cout<<"\n Roll no "<<rollno;
                     cout<<"\n Marks 1.. "<<marks1;
                     cout<<"\n Marks 2.. "<<marks2;
                     cout<<"\n Total Marks.. "<< total_marks;
          }
        public:
        student()
        {
              name[0]='\0';
              rollno=marks1=marks2=total_marks=0;
              cout<<"\n Constructor executed ... ";
         }
         void execute()
         {
              accept();
              compute();
              display();
         }
};
void main()
{
       clrscr();
       student stud;
       stud.execute();
}
```

இனக்குழு அறிவிப்பின் வடிவம்:

பொதுவடிவம்	**மேற்கண்ட எடுத்துக்காட்டில் உள்ள இனக்குழு வடிவம்**

class class-name
{
private:
variable declaration
function declaration
 	
protected:
variable decl.
function decl.
 	
public:
variable decl.
function decl.
};
	class student
	{ **private;**
	char name [10];
	int rollno, marks1, marks2, total_marks;
	
	protected:
	void accept( );
	void compute( );
	void display( );
	
	public:
	student( );
	void execute( );
	};

✓ class என்னும் சிறப்புச் சொல், இது பயனர் வரையறுக்கும் class என்னும் தரவு இனம் என்பதை உணர்த்துகிறது.

✓ இனக்குழுவின் உடற்பகுதி நெளிவு அடைப்புக் குறிகளால் அடக்கப்பட்டு ஓர் அரைப்புள்ளியுடன் முற்றுப் பெறுகிறது.

✓ இனக்குழுவின் உடற்பகுதி மாறிகள் மற்றும் செயற்கூறுகளின் அறிவிப்புகளைக் கொண்டுள்ளது.

✓ இனக்குழுவின் உடற்பகுதி மூன்று அணுகியல்பு வரையறுப்பிகளைக் (வெளிக்காட்டும் சிட்டைகள்) கொண்டுள்ளது: private, public, protected.

✓ private என்னும் அணுகியல்பைக் குறிப்பிடுவது கட்டாயமில்லை. எவ்வித அணுகியல்பும் குறிப்பிடப்படவில்லை எனில் இனக்குழுவின் உறுப்புகள் இயல்பாக private என்றே கருதிக்கொள்ளப்படும்.

✓ private என அறிவிக்கப்பட்ட உறுப்புகளை அந்த இனக்குழுவுக்குள்ளே தான் அணுக முடியும்.

✓ protected என அறிவிக்கப்பட்ட உறுப்புகளை அந்த இனக்குழுவுக்குள்ளும், அந்த இனக்குழுவின் அடிப்படையில் தருவிக்கப்பட்ட இனக்குழுவுக்குள்ளும் எடுத்தாள முடியும்.

✓ public என அறிவிக்கப்பட்ட உறுப்புகளை அந்த இனக்குழுவுக்கு வெளியேயும் எடுத்தாள முடியும்.

6.3 தரவு அருவமாக்கம் (Data Abstraction)

தரவுகளையும் செயற்கூறுகளையும் ஒன்றுசேர்த்து ஒற்றை உரு பொருளாய்க் கட்டிவைப்பதை ***உறைபொதியாக்கம்*** (Encapsulation) என்று அழைக்கிறோம்.

private என வகைப்படுத்தப்பட்ட தரவுகளையும் செயல்கூறுகளையும் இனக்குழுவுக்கு வெளியிலிருந்து அணுக முடியாது. இதனைத் ***தரவு மறைப்பு*** (Data Hiding) என்கிறோம். பிற இனக்குழுக்களின் பொருள்களும் (objects), உறுப்புகளும் (members) வரம்புக்குட்பட்ட முறையில் அணுகுமாறு கட்டுப்பாடு விதிக்கும் நுட்பம் ***தரவு அருவமாக்கம்*** (Data Abstraction) எனப்படுகிறது. 'தரவு மறைப்பு' மூலமாக 'தரவு அருவமாக்கம்' சாத்தியமாகி உள்ளது என்றும் கூறலாம்.

தரவு மறைப்பு என்பது பொருள்நோக்கு நிரலாக்கத்தின் (Object Oriented Programming-OOP) அடிப்படையான பண்புக் கூறாகும்.

private	அதே இனக்குழுவின் உறுப்புகளும் நட்புச் செயற்கூறுகள் (friend functions) எனப்படும் சில சிறப்புச் செயற்கூறுகளும் மட்டுமே அணுக முடியும்.
protected	அதே இனக்குழு மற்றும் அதன் தருவிக்கப்பட்ட இனக்குழுக்களின் உறுப்புகளால் அணுக முடியும்.
public	அதே இனக்குழுவின் உறுப்புகளும் பொருள்களும் மட்டுமின்றி வெளி இனக்குழுக்களின் உறுப்புகளாலும் அணுக முடியும்.

6.4 தரவு உறுப்புகளும் உறுப்புச் செயற்கூறுகளும்

இனக்குழு இருவகை உறுப்புகளைக் கொண்டது. ஒன்று ***தரவு உறுப்புகள்*** (Data Members). மற்றது ***உறுப்புச் செயற்கூறுகள்*** (member functions). தரவு உறுப்புகள் என்பவை ஓர் இனக்குழுவின் பண்புக்கூறுகளை உணர்த்தும் தரவு மாறிகளைக் குறிக்கின்றன. உறுப்புச் செயற்கூறுகள் என்பவை ஓர் இனக்குழுவில் குறித்த பணிகளை நிறைவேற்றுகின்ற செயற்கூறுகளைக் குறிக்கின்றன. உறுப்புச் செயற்கூறுகளை வழிமுறைகள் (methods) என்றும், தரவு உறுப்புகளைப் பண்புக்கூறுகள் (attributes) என்றும் கூறுவது உண்டு. இப்போது, Program-6.1-ல் வரையறுக்கப்பட்டுள்ள student என்னும் இனக்குழுவை அடிப்படையாகக் கொண்டு தகவல் தரப்பட்டுள்ள அட்டவணை-6.1-ஐ நோக்குக. இனக்குழுக்கள் ***ஆக்கிகள்*** (constructors), ***அழிப்பிகள்*** (destructors) என்னும் தனிச் சிறப்பான உறுப்புச் செயற்கூறுகளையும் கொண்டுள்ளன. ஆக்கிகள்/அழிப்பிகள் பற்றிப் பாடம் 8-ல் விரிவாகப் படிப் போம்.

student	தரவினக் குறிப்பெயர் student என்பதை இனக்குழு ஒட்டு(tag) என்றும் கூறுவர்.
name, rollno, marks1,marks2, total_marks	தரவு உறுப்புகள்
accept() compute() display() execute() student()	உறுப்புச் செயற்கூறுகள் அல்லது வழிமுறைகள்
stud	student இனக்குழுவின் சான்றுரு (instance)/ பொருள் (object) / மாறி (variable)

அட்டவணை 6.1 Program– **6.1 ல் உள்ள** student **இனக்குழு**

6.5 பொருள்களை உருவாக்குதல்

student stud;

என்னும் அறிவிப்புக் கூற்றை நோக்குங்கள். இக்கூற்றின் பொருள் stud என்பது student இனக்குழுவின் சான்றுரு(instance) அல்லது பொருள்(object) என்பதாகும்.

ஓர் இனக்குழு வரையறுக்கப்பட்டபின், அவ்வினத்தில் மாறிகளை அறிவிக்க முடியும். stud என்பது student இனத்தைச் சார்ந்த மாறி. student என்பது class என்னும் தரவினமாகும். சி++ மொழியில் இனக்குழு மாறிகள் ***பொருள்கள்*** (objects) என்று அறியப்படுகின்றன. அடிப்படைத் தரவினங்களில் மாறிகளை அறிவிப்பதைப் போன்றே இனக்குழுவில் பொருள்கள் அறிவிக்கப் படுகின்றன. இனக்குழு வரையறுப்பு முடிகின்ற நெளிவு அடைப்புக் குறியை ஒட்டி, மாறிகளின் பெயரைக் குறிப்பிட்டும் பொருள்களை உருவாக்க முடியும்.

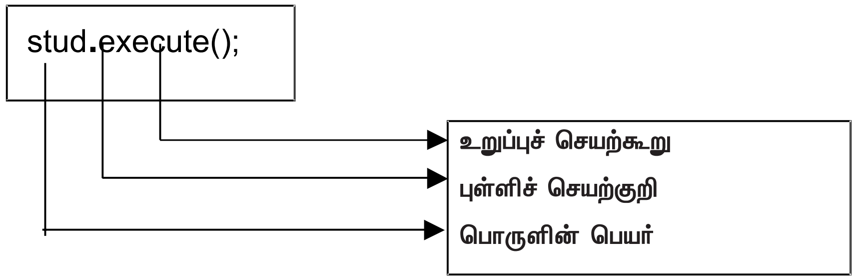
```
class student
{
        private:


          protected:


          public:

}stud;// stud is an object
```

```
class student
{
          private:
          ........
          protected:
          .......
           public:
};
void main()
{
   student s, s1[5];
}
the variables s and s1 are objects
or instances of the class stduent
```

படம் 6.1 பொருள்களை உருவாக்குதல்

6.6 இனக்குழு உறுப்புகளை அணுகுதல்

இனக்குழுவின் உறுப்புகளைப் புள்ளிச் செயற்குறி மூலம் அணுக முடியும். எடுத்துக்காட்டாக, student இனக்குழுவின் execute() செயற்கூறினை அழைக்கும் கூற்று இவ்வாறு அமையும்:



ஓர் இனக்குழுவின் private தரவுகளை அதே இனக்குழுவின் உறுப்புச் செயற்கூறுகளும், நட்புச் செயல்கூறுகள் எனப்படும் சில சிறப்புச் செயற் கூறுகளுமே அணுக முடியும்.

எடுத்துக்காட்டில் தரப்பட்டுள்ள student இனக்குழுவில் name, marks1, marks2, total_marks ஆகிய தரவு உறுப்புகளை, accept(), display(), compute() ஆகிய உறுப்புச் செயற்கூறுகள் மட்டுமே அணுக முடியும். இனக்குழுவுக்கு வெளியே அறிவிக்கப்படும் பொருள்கள், private அல்லது protected என வரையறுக்கப்பட்டுள்ள உறுப்புகளையோ, செயற்கூறுகளையோ அணுக முடியாது.

public என அறிவிக்கப்பட்டுள்ள உறுப்புச் செயற்கூறுகளை அதே இனக்குழுவின் பொருள்களால் அணுக முடியும்.

stud.execute();

என்னும் அழைப்புக் கூற்று ஏற்கத் தக்கதே. காரணம், execute() என்பது public என அறிவிக்கப்பட்டுள்ள ஓர் உறுப்புச் செயற்கூறு. எனவே, இனக் குழுவுக்கு வெளியே அறிவிக்கப்பட்டுள்ள stud என்னும் பொருள் மூலமாக அணுக முடியும். ஆனால், stud.accept(), stud.compute(), stud.display(), stud.marks1- என்றெல்லாம் அணுக முயன்றால், நிரல்பெயர்ப்பி "not accessible" என்னும் பிழைசுட்டும் செய்தியைத் தரும். இரண்டு எண்களைக் கூட்டிச் சொல்லும் பணியை விளக்கும் Program- 6.2 இன்னோர் எடுத்துக் காட்டாகும். இதிலுள்ள இனக்குழு, மூன்று முழுஎண் மாறிகளையும், எண்களைப் பெற்றுக் கூட்டித் தருகின்ற செயற்கூறினையும் கொண்டுள்ளது. sum என்னும் மாறி public என்னும் அணுகியல்பு கொண்டுள்ளதால், s என்னும் பொருள் அதனை அணுக முடிகிறது.

```
//Program - 6.2
# include <iostream.h>
# include <conio.h>

class add
{
       private:
              int a,b;
       public:
              int sum;

              void getdata()
              {
                     a=5;
                     b=10;
                     sum = a+b;
              }
};

void main()
{
       add s;
       s.getdata();
       cout<<s.sum;
}
```

6.7 இனக்குழுவின் செயற்கூறுகளை வரையறுத்தல்

```
class add
{
       int a,b;
       public:
       add()
       {
           a = 0;               ----> வழிமுறை–1
           b = 0;
       }
       void display();
};
void add::display()
{
       int sum;
       sum = a+b;               ----> வழிமுறை–2
       cout<<sum;
}
```

வழிமுறை 1-ல், add() என்னும் உறுப்புச் செயற்கூறு, add இனக்குழுவுக்குள்ளேயே அறிவிக்கப்பட்டு, வரையறுக்கப்பட்டுள்ளது.

வழிமுறை 2-ல், display() என்னும் உறுப்புச் செயற்கூறு, add இனக்குழுவுக்குள் அறிவிக்கப்பட்டு, இனக்குழுவுக்கு வெளியே வரையறுக்கப்பட்டுள்ளது.

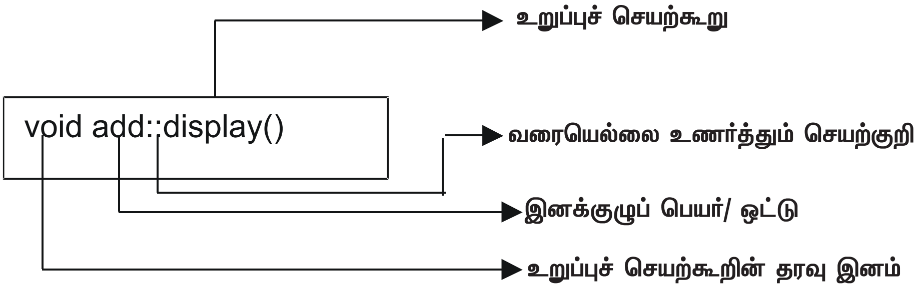
இனக்குழுவின் உறுப்புச் செயற்கூறுகளை இவ்வாறு இரு வகையாகவும் வரையறுக்கலாம். இனக்குழுவுக்குள் வரையறுக்கப்படும் செயற்கூறுகள் inline செயற்கூறுகளைப்போல் இயங்குகின்றன.

இனக்குழுவுக்கு வெளியே வரையறுக்கப்படும் செயற்கூறுகள்,

```
type class_name: : function_name();
```

என்னும் முன்வடிவைக் கொண்டுள்ளன.

எடுத்துக்காட்டு:



function_name()- க்கு முன்பாக இடம்பெற்றுள்ள class_name:: (add::) என்னும் உறுப்பைக் குறிக்கும் முன்னொட்டு, அச்செயற்கூறு, class_name- க்கு உரியது என்பதைச் சுட்டுகிறது. செயற்கூறின் தலைப்பில் குறிப்பிடப்பட்டுள்ள இனக்குழுவுக்குள் அடங்கியது என்னும் வரையெல்லையைக் குறிக்கிறது.

உறுப்புச் செயற்கூறுகள் தனிச்சிறப்பான பண்பியல்புகளைக் கொண்டுள்ளன. நிரல் உருவாக்கத்தில் இவை அடிக்கடி பயன்படுத்திக்கொள்ளப்படுகின்றன.

✓ பல இனக்குழுக்கள் தம் செயற்கூறுகளுக்கு ஒரே பெயரைப் பயன்படுத்திக்கொள்ள முடியும். செயற்கூறினுக்கு முன்னொட்டாக இடம்பெறும் இனக்குழுப் பெயர் அதன் வரையெல்லையைக் குறிக்கும்.

- ✓ உறுப்புச் செயற்கூறுகள் இனக்குழுவின் private தரவுகளை அணுக முடியும். உறுப்பில்லாத செயற்கூறுகள் அணுக முடியாது.
- ✓ ஓர் உறுப்புச் செயற்கூறு இன்னோர் உறுப்புச் செயற்கூறினை நேரடியாக அழைக்க முடியும். புள்ளிச் செயற்குறி தேவையில்லை. (இவ்வாறு அமைவது உறுப்புச் செயற்கூறுகளின் பின்னலமைப்பு எனப்படுகிறது).
- ✓ உறுப்புச் செயற்கூறுகள், ஏற்கத்தகு சி++ தரவு இனச் செயலுருபுகளை ஏற்கும். பொருள்களையும் செயலுருபுகளாக அனுப்பிவைக்க முடியும்.
- ✓ உறுப்புச் செயற்கூறு திருப்பியனுப்பும் மதிப்பு, இனக்குழுவின் பொருளாகவும் இருக்கலாம்.
- ✓ உறுப்புச் செயற்கூறுகள் static இனமாகவும் இருக்க முடியும்.

6.8 பொருள்களுக்கு நினைவக ஒதுக்கீடு

இனக்குழு வரையறுப்பின் ஒரு பகுதியாக உறுப்புச் செயற்கூறுகள் வரையறுக்கப்பட்டிருக்கும்போதே அவை உருவாக்கப்பட்டு நினைவகத்தில் இருத்தப்படுகின்றன. அந்த இனக்குழுவைச் சார்ந்த அனைத்துப் பொருள்களுமே ஒரே உறுப்புச் செயற்கூறினையே பயன்படுத்திக் கொள்வதால், ஒவ்வொரு பொருள் உருவாக்கப்படும்போதும் உறுப்புச் செயற்கூறுகளுக்குத் தனித்தனி நினைவக இடம் ஒதுக்கப்படுவதில்லை. உறுப்பு மாறிகளுக்குத் தேவையான நினைவக இடம் மட்டும் ஒவ்வொரு பொருளுக்கும் தனித்தனியே ஒதுக்கப்படுகிறது. வெவ்வேறு பொருள்களுக்கு வெவ்வேறு இடம் ஒதுக்கப்படுவது கட்டாயமாகும். காரணம், வெவ்வேறு பொருள்களின் உறுப்பு மாறிகள் வெவ்வேறு மதிப்புகளைக் கொண்டிருக்கும்.

கீழேயுள்ள இனக்குழு அறிவிப்பை நோக்குக:

```
class product
{
        int code, quantity;
        float price;
        public:
                void assign_data();
                void display();
};
void main()
{
        product p1, p2;
}
```

இதிலுள்ள உறுப்புச் செயற்கூறுகள் assign_data(), display() ஆகியவை நினைவகத்தில் பொதுவிடப் பகுதியில் இருத்தப்படுகின்றன. அதாவது p1, p2 ஆகிய இரண்டு பொருள்களும் பொதுவிடப் பகுதியிலுள்ள கட்டளைகளை அணுக முடியும்.

p1, p2 ஆகிய இரண்டு பொருள்களின் நினைவக ஒதுக்கீட்டைக் காண்க:

பொருள்	தரவு உறுப்புகள்	நினைவக ஒதுக்கீடு
p1	code, quantity, price	8 பைட்டுகள்
p2	code, quantity, price	8 பைட்டுகள்

அட்டவணை 6.2 பொருள்களுக்கான நினைவக ஒதுக்கீடு

ஓர் இனக்குழுவின் உறுப்புச் செயற்கூறுகள், Program- 6.3- ல் விளக்கப் பட்டுள்ளதுபோல, உறுப்பில்லாத வேறெந்தச் செயற்கூறுகளையும் போலவே செயலுருபுகளைக் கையாள முடியும்.

6.9 static தரவு உறுப்புகள்

இனக்குழுவின் ஒரு தரவு உறுப்புக்கு static என்னும் தகுதியை அளிக்க முடியும். static தரவு உறுப்பு-

✓ அந்த இனக்குழுவில் முதல் பொருள் உருவாக்கப்படும்போது, 0 என்ற தொடக்க மதிப்பைப் பெறும். அதன்பிறகு தொடக்க மதிப்பிருத்தப்படுவதில்லை.

✓ நினைவகத்தில், உறுப்பு மாறிக்கு ஒரேயொரு நகல் மட்டுமே இருக்கும் (பொது இடத்தின் ஒரு பகுதியாக). அதே இனக்குழுவின் பிற பொருள்களும் நினைவகத்திலுள்ள அந்தவொரு நகலையே பகிர்ந்துகொள்கின்றன.

✓ அதன் வரையெல்லையும் அணுகியல்பும் அந்த இனக் குழுவுக்குள்ளேதான். ஆனால் அதன் வாழ்நாள், நிரல் செயல்பட்டு முடியும்வரை இருக்கும்.

static உறுப்பு மாறி பயன்படுத்தப்பட்டுள்ள ஒரு நிரலைக் காண்க (Program-6.4):

```
// Program - 6.3
#include<iostream.h>
#include<conio.h>
class product
{
      int code, quantity;
      float price;

      public:
      void assign_data( int c, int q, float p)
      {
         code = c;
         quantity = q;
         price = p;
      }

      void display()
      {
          cout<<"\n Code : "<<code;
          cout<<"\n Quantity :"<<quantity;
          cout<<"\n Price :  "<< price;
      }
};

void main()
{
      product p;
      p.assign_data( 101, 200, 12.5);
      p.display();
}
```

```
// Program - 6.4
// To demonstrate the use of static member variables

#include<iostream.h>
#include<conio.h>
class simple_static
{
        int a,b,sum;
        static int count;
        public:
        void accept()
        {
                cout<<"\n Enter values.. ";
                cin>>a>>b;
                sum = a+b;
                count++;
        }
        void display()
        {
                cout<<"\n The sum of two numbers … "<<sum;
                cout<<"\n This is addition… "<<count;
        }
};
static_simple.count = 0;
void main()
{
    simple_static p1,p2,p3;
    p1.accept();
    p1.display();
    p2.accept();
    p2.display();
    p3.accept();
    p3.display();
}
```

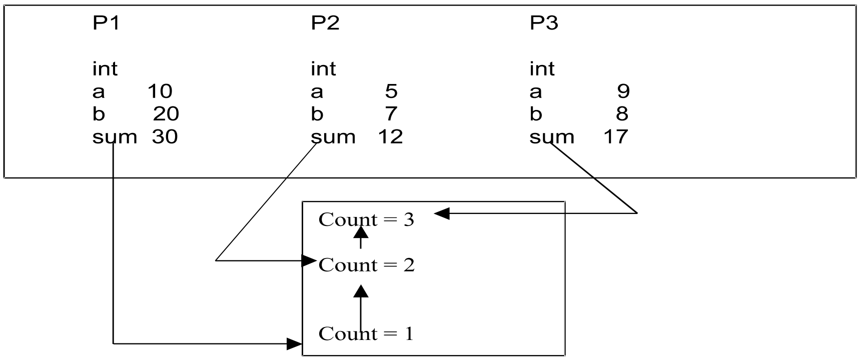
வெளியீடு:

Enter values …… 10 20
The sum of two numbers ………… 30
This is addition 1

Enter values……… 5 7
The sum of two numbers……………12
This is addition 2

Enter values……….. 9 8
The sum of two numbers ……………17
This is addition 3

static மாறியான count ஒரேயொரு முறை மட்டும் 0 எனத் தொடக்க மதிப்பிருத்தப்படுகிறது. ஒவ்வொரு முறை இரண்டு எண்களின் கூட்டுத்தொகை கணக்கிடப்படும்போதெல்லாம் count-ன் மதிப்பு ஒன்று மிகுக்கப்படுகிறது. accept() செயற்கூறு மும்முறை செயல்படுத்தப்படுவதால் count-ன் மதிப்பும் மும்முறை மிகுக்கப்பட்டு 3 என்னும் மதிப்பைப் பெறுகிறது. அனைத்துப் பொருள்களும் count- ன் ஒரேயொரு நகலை மட்டுமே பகிர்ந்துகொள்வதால், count- ன் மதிப்பு 3 ஆகிறது. இது, படம் 6.2-ல் விளக்கப்பட்டுள்ளது.



படம் 6.2 static **உறுப்பு மாறி** – count

static உறுப்பு மாறியின் தொடக்க மதிப்பு இனக்குழுவுக்கு வெளியே இருத்தப்படுகிறது.

6.10 பொருள்களின் அணிகள்

கீழ்க்காணும் இனக்குழு வரையறுப்பையும், அதன் நினைவக ஒதுக்கீட்டையும் நோக்குக:

```
class product
{
    int code,quantity;
    float price;
    public :
   void assign_Data();
   void display();
} p[3];

void main()
{
  p[0].assign_Data();
  p[0].display();
}
```

code quantity price	p[0]
code quantity price	p[1]
code quantity price	p[2]

பயிற்சி வினாக்கள்

I. பிழைகளைக் கண்டறிந்து திருத்துக:

```
class x
{
        public:
        int a,b;
        void init()
        {
            a =b = 0;
        }
        int sum();
        int square();
};
int sum()
{
        return a+b;
}
```

```
int square()
{
        return sum() * sum()
}
```

விடை:

int x : : sum() மற்றும் int x : : square()

II.

```
#include<iostream.h>
class simple
{
        int num1, num2 , sum = 0;
    protected:
        accept()
        {
                cin>>num1>>num2;
        }
    public:
        display()
        {
                sum = num1 + num2;
        }
};
void main()
{   simple s;
    s.num1=s.num2=0;
    s.accept();
    display();
}
```

விடை:

1) sum என்னும் உறுப்பு மாறியில் அறிவிப்பின்போது தொடக்கமதிப்பு இருத்த முடியாது.

2) num1, num2 ஆகிய உறுப்பு மாறிகள் private என்பதால் அவற்றை main()- லிருந்து அணுக முடியாது.

3) s.accept() என்று அழைக்க முடியாது. காரணம் அது protected என வரையறுக்கப்பட்டுள்ளது.

4) display() என்னும் செயற்கூறினை ஒரு பொருளின் மூலம் அழைக்க வேண்டும்.

III.

```
#include<iostream.h>
#include<conio.h>
class item
{
        private:
        int code,quantity;
        float price;
        void getdata()
        {
                cout<<"\n Enter code, quantity, price ";
                cin>>code>>quantity>>price;
        }
        public:
        float tax = 0;
        void putdata()
        {
                cout<<"\n Code : "<<code;
                cout<<"\n Quantity : "<<quantity;
                cout<<"\n Price :  "<<price;
                if( quantity >100 )
                        tax = 2500;
                else
                        tax =1000;
```

```
                    cout<<" \n Tax :"<<tax;
            }
};

void main()
{ item i;   }
```

மேற்கண்ட நிரலின் அடிப்படையில் கீழ்க்காணும் அட்டவணையை நிறைவு செய்க:

சான்றுரு i- ன் நினைவக ஒதுக்கீடு	private தரவு உறுப்புகள்	public தரவு உறுப்புகள்	i மூலம் அணுக முடிகிற செயற்கூறு களும் தரவு உறுப்புகளும்

IV.

1) கீழ்க்காணும் வரையறுப்புகளுடன் employee என்னும் ஓர் இனக்குழுவை வரையறுக்கவும்:

employee இனக்குழுவின் private உறுப்புகள்:
empno - முழுஎண்
ename - 20 எழுத்துகள்
basic - float
netpay, hra, da - float
calculate()- basic+hra+da என்பதைக் கணக்கிட்டு ஒரு float இனமதிப்பைத் திருப்பியனுப்பும் செயற்கூறு.
employee இனக்குழுவின் public உறுப்புச் செயற்கூறுகள்:
havedata() - empno, ename, basic, hra, da - ஆகிய மதிப்புகளை உள்ளீடாகப் பெற்று, calculate() செயற்கூறினை அழைத்து netpay- ஐக் கணக்கிடச் செய்யும் செயற்கூறு.
dispdata() - தரவு உறுப்புகள் அனைத்தையும் திரையில் காட்டும் செயற்கூறு.

2) கீழ்க்காணும் வரையறுப்புகளுடன் math என்னும் இனக்குழுவை வரையறுக்கவும்:

private உறுப்புகள்:
num1, num2, result - float
init() செயற்கூறு - num1, num2, result ஆகியவற்றில் 0 என்னும் தொடக்க மதிப்பிருத்தும்.

protected உறுப்புகள்:

add() செயற்கூறு - num1, num2 இரண்டையும் கூட்டி, கூட்டுத் தொகையை result- ல் இருத்தும்.

prod() செயற்கூறு - num1, num2 இரண்டையும் பெருக்கி, பெருக்குத் தொகையை result- ல் இருத்தும்.

public உறுப்புகள் :

getdata() செயற்கூறு - num1, num2 ஆகிய மதிப்புகளை ஏற்கும்

menu() செயற்கூறு - 1. Add...
2. Prod...

என்னும் பட்டியைக் (menu) காட்டும்.

1 என்பதைத் தேர்வு செய்தால் add() செயற்கூறினையும், 2-ஐத் தேர்வு செய்தால் prod() செயற்கூறினையும் செயல்படுத்தி, result- ஐத் திரையில் காட்டவேண்டும்.

பாடம் 7

பல்லுருவாக்கம்
(Polymorphism)

7.1 முன்னுரை

polymorphism என்ற சொல், 'பல வடிவங்கள்' (poly-many, morph-shapes) என்னும் பொருளைத் தருகிறது. தமிழில் ***பல்லுருவாக்கம்*** என்கிறோம். சி++ மொழியில், பல்லுருவாக்கம், ***செயற்கூறு பணிமிகுப்பு*** (function overloading), ***செயற்குறி பணிமிகுப்பு*** (operator overloading) ஆகியவற்றின் மூலம் நிறை வேற்றப்படுகிறது. உண்மையில் 'பணிமிகுப்பு' (overloading) என்பது ஒரே பெயர், ஒன்றுக்கு மேற்பட்ட வெவ்வேறு பொருளை உணர்த்துவதைக் குறிக்கிறது. ஆக, 'பணிமிகுத்த செயற்கூறு' என்பது ஒன்றுக்கு மேற்பட்ட தனித்த பொருள் கொண்ட செயற்கூறினைக் குறிக்கிறது. பொருள்நோக்கு நிரலாக்கத்துக்குத் துணைபுரியும் 'செயற்கூறு பணிமிகுப்பு' சி++ மொழியின் பண்புக்கூறுகளில் ஒன்றாகும்.

7.2 செயற்கூறு பணிமிகுப்பு (Function Overloading)

செய்தி அல்லது தரவினை ஒன்றுக்கு மேற்பட்ட வடிவங்களில் செயலாக்கவல்ல செயற்கூறின் திறனையே செயற்கூறு பணி மிகுப்பு என்கிறோம்.

ஒரு நிரலர் கீழ்க்காணும் செயற்கூறுகளை வரையறுக்க விரும்புகிறார் என வைத்துக்கொள்வோம்:

```
area_circle()       // ஒரு வட்டத்தின் பரப்பைக் கணிக்க
area_triangle()     // ஒரு முக்கோணத்தின் பரப்பைக் கணிக்க
area_rectangle()    // ஒரு செவ்வகத்தின் பரப்பைக் கணிக்க
```

வெவ்வேறு வடிவங்களின் பரப்பளவைக் கண்டறிய எழுதப்பட்ட மேற்கண்ட மூன்று முன்வடிவுகளையும், ஒரே பெயரில் (வெவ்வேறு அளபுருக்களுடன்) வரையறுக்க முடியும்.

```
float area(float radius);
float area(float half, float base, float height);
float area(float length, float breadth);
```

இப்போது Program-7.1-ஐ நோக்குக. மூன்று வடிவங்களில் எந்த வடிவத்தின் பரப்பளவையும் கணக்கிட area() என்னும் செயற்கூறினைச் செயல் படுத்தியுள்ளமை காண்க:

```
// Program - 7.1
// to demonstrate the polymorphism - function overloading

#include<iostream.h>
#include<conio.h>

float area ( float radius )
{       cout << "\nCircle …";
        return ( 22/7 * radius * radius );
}
float area (float half, float base, float height)
{       cout << "\nTriangle ..";
        return (half* base*height);
}


float area ( float length, float breadth )
{        cout << "\nRectangle …";
         return ( length *breadth );
}

void main()
{
        clrscr();
        float r,b,h;
        int choice = 0;
        do
        {
                clrscr();
                cout << "\n Area Menu ";
                cout << "\n 1. Circle ... ";
                cout << "\n 2. Traingle ...";
                cout << "\n 3. Rectangle ... ";
                cout <<"\n 4. Exit ... ";
                cin>> choice;
                switch(choice)
                {
                        case 1 :
                                cout << "\n Enter radius ... ";
                                cin>>r;
                                cout<<"\n The area of circle is ... "
                                  << area(r);
                                getch();
                                break;
                        case 2:
                                cout<< "\n Enter base, height ... ";
                                cin>>b>>h;
                                cout<<"\n The area of a triangle is..."
                                     << area (0.5, b, h);
                                getch();
                                break;
                        case 3:
                              cout<< "\n Enter length, breadth.. ";
                              cin>>h>>b;
                              cout<<"\n The area of a rectangle is ..."
                                   << area(h,b);
                              getch();
                              break;
                 }
                 }while (choice <=3);
}
```

மூன்று செயற்கூறுகளின் முன்வடிவுகளைக் கவனித்தீர்களா? அவை:

```
float area(float radius);
float area(float half, float base, flaot height);
float area(float length, float breadth);
```

மூன்று area() செயற்கூறுகளின் வரையறைகளும் ஒன்றுக்கொன்று மாறு பட்டுள்ளன என்று நினைக்கிறீர்களா? ஆம், ஒவ்வொரு செயற்கூறின் முன் வடிவும் செயலுருபுகளின் எண்ணிக்கையில் வேறுபடுகின்றன. முதலாவது முன்வடிவில் ஒரு செயலுருபும், இரண்டாவதில் மூன்று செயலுருபுகளும், மூன்றாவதில் இரண்டு செயலுருபுகளும் உள்ளன. நம்முடைய எடுத்துக் காட்டில் மூன்று செயற்கூறுகளுமே float இனச் செயலுருபுகளைக் கொண் டுள்ளன. இவ்வாறுதான் இருக்க வேண்டும் என்பதில்லை. ஒவ்வொரு முன் வடிவும் வெவ்வேறு தரவினச் செயலுருபுகளைக் கொண்டிருக்கலாம். செயலுருபுகளின் எண்ணிக்கை வெவ்வேறாக இருக்கலாம். செயலுருபுகளின் எண்ணிக்கை ஒன்றாக இருப்பின் அவற்றின் தரவினங்கள் மாறி இருக்க வேண்டும்.

கீழ்க்காணும் முன்வடிவுகளில் செயற்கூறு பணிமிகுப்புக்கான முன்வடிவு களில் பிழையானதைக் கண்டறிய முடியுமா? ஏன் என்று காரணம் கூற முடியுமா?

செயற்கூறு முன்வடிவுகள்:	பிழையான முன்வடிவுகள்:

void fun(int x);
void fun(char ch);
void fun(int y);
void fun(double d);
	void fun(int x);
	void fun(int y);
	இரண்டும் ஒரே எண்ணிக்கை யில் ஒரே இனச் செயலுருபு களைக் கொண்டுள்ளன. எனவே பிழையாகும்.

செயற்கூறு பணிமிகுப்பில் செயற்கூறுகள் எவ்வாறு இயக்கப்படுகின்றன?

நிரல்பெயர்ப்பி, ***மிகச்சிறந்த பொருத்தம்*** (Best Match) என்னும் செயல்நுட்பத்தைப் பின்பற்றுகிறது. இந்தச் செயல்நுட்பத்தின்படி நிரல் பெயர்ப்பி-

✓ செயற்கூறு அழைப்புக் கூற்றுக்கு மிகச் சரியாகப் பொருந்தும் செயற்கூறு முன்வடிவைத் தேடி பார்க்கும்.

✓ மிகச் சரியாகப் பொருந்தும் செயற்கூறு இல்லையெனில், அடுத்துப் பொருத்தமுள்ள செயற்கூறினைத் தேடும். அதாவது, எண்வகைச் செயலுருபை வேறு எண்வகைக்கு இனமாற்றம் செய்து, அதற்கேற்ற செயற்கூறு முன்வடிவு உள்ளதா எனப் பார்க்கும்.

எடுத்துக்காட்டாக, மேற்கண்ட நிரலில் (Program-7.1) float area (float radius) என்னும் முன்வடிவு உள்ளது. இதன்பொருள், அழைப்புக் கூற்று area(r) என்பதில் r என்னும் செயலுருபு float இனத்தைச் சேர்ந்தாக இருக்க வேண்டும். r என்பது int இனமாக இருக்கும் எனில், முழுஎண் வகை இனமாற்றத்தின்படி, int மாறிலி/மாறியை char அல்லது float அல்லது double இனத்தோடு பொருத்திக்கொள்ள முடியும். இச்செயல்நுட்பத்தின் படி area(r) என்னும் அழைப்பு area(float radius) என்னும் செயற்கூறோடு பொருத்தப்பட்டு அது இயக்கப்படும்.

இத்தகைய எண்வகை இனமாற்றம் முழுக்க முழுக்க நிரல்பெயர்ப்பி சார்பானது. வெவ்வேறு நிரல்பெயர்ப்பிகள் வெவ்வேறு வகையில் இனமாற்றம் செய்யலாம். எண்வகை இனமாற்றம் பெரும்பாலும் இவ்வாறு அமையும்:

✓ char தரவினத்தை int/float/double இனமாக மாற்ற முடியும்.

✓ int தரவினத்தை char/float/double இனமாக மாற்ற முடியும்.

✓ float தரவினத்தை int/double/char இனமாக மாற்ற முடியும்.

✓ double தரவினத்தை int அல்லது float இனமாக மாற்ற முடியும்.

Program-7.1 நிரலில் உள்ள area() செயல்கூறினுக்கு அழைப்புக் கூற்று கீழ்க்காணுமாறு அமையுமெனில் விடை என்னவாக இருக்கும்?

செயற்கூறு அழைப்புக் கூற்று	**வெளியீடு**
area (5.0)	
area (0.5, 4.0, 6.0)	
area(3.0, 4.5)	

செயற்கூறு பணிமிகுப்பின் விதிமுறைகள்

1) பணிமிகுத்த செயற்கூறுகள்முறையான அளபுருக்களின் எண்ணிக்கையிலோ, அவற்றின் தரவு இனங்களிலோ வேறுபட்டிருக்க வேண்டும்.

2) பணிமிகுத்த செயற்கூறுகள் திருப்பியனுப்பும் தரவினம் ஒன்றாக இருக்கலாம், வேறுபட்டும் இருக்கலாம்.

3) பணிமிகுத்த செயற்கூறுகளின் முன்னியல்புச் செயலுருபுகளை, அளபுருக்களின் பட்டியலில் ஒரு பகுதியாக சி++ நிரல்பெயர்ப்பி கருதிக் கொள்ளாது.

4) தொடர்பில்லாத இரு செயற்கூறுகளுக்கு ஒரே பெயரைச் சூட்டாமல் இருப்பது நல்லது.

முறையற்ற அறிவிப்புகள், கீழேயுள்ளவாறு செயற்கூறு அழைப்பினில் மோதலை உருவாக்கும்.

```
void fun ( char a, int times)
{
      for (int i=1; i<=times;i++)
                  cout<<a;
}
void fun( char a= ‘*’, int times )
{
      for(int i=1;i<=times;i++)
                  cout<<a;
}
void fun( int times)
{
      for(int i=1; i<=times ;i++)
                  cout<<’@’;
}
void main()
{
      fun ( ‘+’, 60);
      fun(60);
}
```

மேற்கண்ட நிரலை நிரல்பெயர்க்கும்போது இரண்டு பிழைகள் சுட்டப்படும்:

- ✓ conflict between fun(char a, int times) and fun(char a = ‘*’, int times)
- ✓ conflict between fun(char a = ‘*’, int times) and fun (int times)

fun (‘+’, 60) என்னும் அழைப்புக் கூற்றை fun(char a, int times) மற்றும் fun(char a = ‘*’, int times) ஆகிய இரண்டு செயற்கூறுகளுடனும் பொருத்தமுடியும்.

வேறுபாடான அளபுருப் பட்டியல்கள் கொண்ட செயற்கூறுகளின் மூலமாகவே ஒரு செயற்கூறினுக்குப் பணிமிகுக்க வேண்டும். அதாவது, அளபுருப் பட்டியல், அளபுருக்களின் எண்ணிக்கை அல்லது தரவினங்கள் அடிப்படையில் வேறுபட்டிருக்க வேண்டும்.

7.3 செயற்குறிப் பணிமிகுப்பு (Operator Overloading)

‘செயற்குறிப் பணிமிகுப்பு’ என்பது, +, ++, -, --, +=, -=, *, ‹, › போன்ற சி++ செயற்குறிகளுக்குக் கூடுதலான செயல்பாட்டினை வரையறுப்பதைக் குறிக்கிறது.

sum = num1 + num 2;

என்னும் கூற்று, இரண்டு எண்களைக் (int/float/double) கூட்டி, விடையை sum என்னும் மாறியில் இருத்தும் பணிக்கான கூற்றாகக் கருதிக் கொள்ளப்படும். இப்போது, கீழ்க்காணும் கூற்றை நோக்குக:

name = first_name + last_name;

இதில் name, first_name, last_name ஆகியவை மூன்றும் சரங்கள் (char இன அணிகள்) ஆகும். சி++ மொழியில் இவ்வாறு + செயற்குறி மூலம் இரண்டு சரங்களை ஒன்றிணைக்க முடியுமா? இரண்டு சரங்களை இணைக்க + செயற்குறியைப் பயன்படுத்த இயலாது என நிரல்பெயர்ப்பி பிழை சுட்டும். இரண்டு சரங்களை இணைக்க, பயனர் கட்டாயமாக strcat() செயற்கூறினையே பயன்படுத்த வேண்டும். எண் இனத்துக்குப் பயன்படுத்துவது போலவே சரங்களுக்கும் + செயற்குறியைப் பயன்படுத்த முடிந்தால் மிகவும் எளிதாக இருக்கும் அல்லவா? ***செயற்குறிப் பணிமிகுப்பு*** நுட்பம் மூலம் + செயற்குறியின் செயல்பாட்டைச் சரங்களுக்கும் நீட்டிக்க முடியும்.

கீழேயுள்ள எடுத்துக்காட்டைக் காண்க:

```
// Program -7.2 – OPERATOR OVERLOADING
# include <iostream.h>
# include <conio.h>
# include <string.h>

class strings
{
   char s[10];
   public :
   strings()
    {
          s[0] = '\0';
    }

   strings(char *c)
    {
         strcpy(s,c);
    }

   char * operator+(strings x1)
    {
         char *temp;
         strcpy(temp,s);
         strcat(temp,x1.s);
         return temp;
    }
};
```

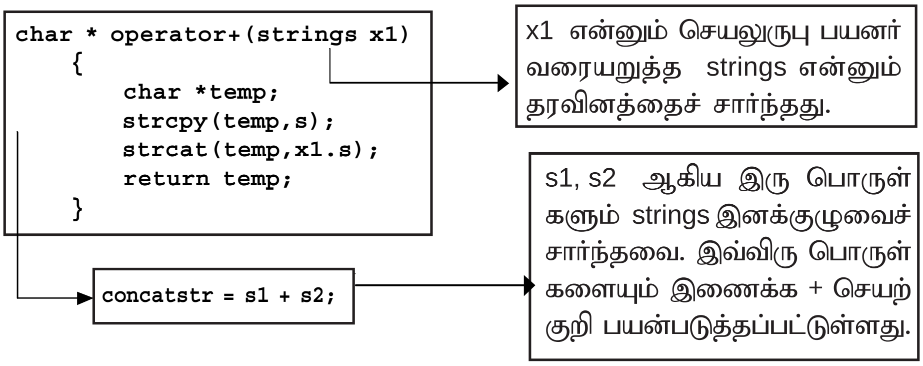
```
void main()
{
    clrscr();
    strings s1("test"),s2(" run\0");
    char *concatstr ;
    concatstr = s1 + s2;
    cout <<"\nConcatenated string ..."
         << concatstr;
    getch();
}
```

concatstr = s1 + s2;

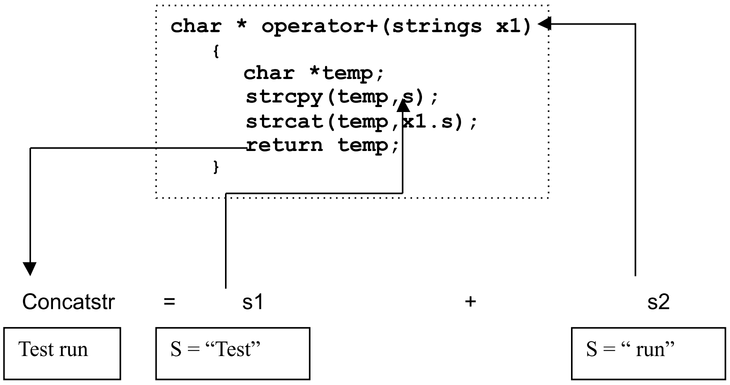
என்னும் கூற்று இரண்டு சரங்களை ஒன்றிணைக்கும்.

char * operator + (strings x1)

என்னும் உறுப்புச் செயற்கூறு மூலமாக, + செயற்குறிக்கு கூடுதலான பணி வரையறுக்கப்பட்டுள்ளது. இச்செயற்கூறு பயனர் வரையறுத்த x1 என்னும் தரவினப் பொருளைச் செயலுருபாக ஏற்கிறது. இதனை இவ்வாறு விளக்கலாம்:



படம் 7.1, மாறிகளுக்கும் அவற்றின் மதிப்புகளுக்கும் இடையேயான தொடர்பினை விளக்குகிறது.



படம் 7.1 மாறிகளுக்கும் மதிப்புகளுக்கும் உள்ள தொடர்பு

'செயற்குறிப் பணிமிகுப்பு' இவற்றையெல்லாம் வழங்குகிறது:

✓ +, *, -, ++, --, >, <, += மற்றும் இதுபோன்ற சி++ மொழியின் அடிப்படையான செயற்குறிகளுக்குப் புதிய வரையறைகளை வழங்குகிறது.

✓ பயனர் வரையறுக்கும் தரவினங்களுக்கும் பணிமிகுத்த செயற்குறிகளை வரையறுக்க முடியும்.

✓ செயற்குறிகளுக்கான செயற்கூறுகள் உறுப்புச் செயற்கூறுகளாக இருக்க வேண்டும். அல்லது நட்புச் செயற்கூறுகளாக (friend functions) இருக்க வேண்டும். (நட்புச் செயற்கூறுகள் இந்த நூலின் வரையெல்லைக்கு அப்பாற்பட்டதாகும்).

✓ ஒரு செயற்குறிக்குத் தரப்படும் புதிய வரையறை அச்செயற் குறியின் மூல வரையறுப்பை மாற்ற முடியாது. எடுத்துக்காட்டாக, மேற்கண்ட நிரலில் + செயற்குறி இரண்டு சரங்களை ஒன்றிணைக்கப் பயன்படுத்தப்பட்டுள்ளது. அதே நிரலில் + செயற்குறியை வழக்கம்போல இரண்டு எண்களைக் கூட்டும் பணிக்கும் பயன்படுத்த முடியும். நிரல்பெயர்ப்பி, பயனர் வரை யறுத்த செயற்குறியின் கட்டளைகளை ஓர் அழைப்புக்கூற்றைப் போலவே செயல்படுத்துகிறது. அதாவது,

```
cout << 5+10;
```

என்னும் கட்டளை 15 எனத் திரையில் காட்டும். (+-ன் மூல வரையறை செயல்படுத்தப்பட்டுள்ளது).

concatstr = s1 + s2;

என்னும் கட்டளை, char * operator+(strings x1) என்னும் உறுப்புச் செயற்கூறினைச் செயல்படுத்தும். காரணம், இதில் + செயற்குறியின் இருபுறமும் இடம்பெற்றுள்ள செயலேற்பிகள் s1, s2 ஆகிய இரண்டும் strings இனக்குழுவின் பொருள்களாகும்.

செயற்குறிப் பணிமிகுப்பில் நடைபெறும் செயல்பாடுகள்:

- ✓ ஒரு புதிய தரவினத்தை வரையறுக்கும் இனக்குழுவை உருவாக்கி அத்தரவினப் பொருள்களைப் பணிமிகுத்த செயற்குறியில் பயன்படுத்திக்கொள்ள வேண்டும்.
- ✓ operator <symbol>() செயற்குறியை இனக்குழுவின் public பகுதியில் அறிவிக்க வேண்டும்.
- ✓ தேவையான செயல்பாடுகளை நிறைவேற்றிக்கொள்ளும் வகையில் செயற்குறிக்கான செயற்கூறினை வரையறுக்க வேண்டும்.

பயனர் வரையறுத்த தரவினப் பொருள்கள் செயற்குறிகளில் எளிமையாகக் கையாளப்பட்டிருப்பதைக் கீழேயுள்ள எடுத்துக்காட்டு விளக்குகிறது. ஓர் இனக் குழுவில் உள்ள தரவு உறுப்புகளின் எதிர்ம மதிப்புகளைக் (negative value) கண்டறிய - செயற்குறியைப் பயன்படுத்தும் முறையை Program - 7.3 விளக்குகிறது.

```
// Program - 7.3
# include <iostream.h>
# include <conio.h>

class negative
{
   int i;
   public :
   void accept()
   {
      cout << "\nEnter a number ...";
        cin >> i;
   }
   void display()
   {
        cout << "\nNumber ..."<<i;
   }
```

```
 void operator-()
  {
      i = -i;
  }
};

void main()
{
      clrscr();
      negative n1,n2;
      n2.accept();
      -n2;
      n2.display();
      getch();
}
```

sum = - num1;

என்பதில், சாதாரண மாறிகளுக்கு - செயற்குறி செயல்படுவது போல, இனக் குழுவின் தரவு உறுப்புகளை void operator-() என்னும் செயற்கூறு எதிர்மம் (Negative) ஆக்குகிறது.

கீழேயுள்ள நிரலை நோக்குக. வினாக்களுக்கு விடை தருக:

```
// Program - 7.4
# include <iostream.h>
# include <conio.h>

class distance
{
   int feet,inches;
   public :
   void distance_assign(int f, int i)
   {
        feet = f;
        inches = i;
   }

   void display()
   {
       cout << "\nFeet   : " << feet
              << "\tInches : " << inches;
   }

   distance operator+(distance d2)
   {
        distance d3;
        d3.feet = feet + d2.feet;
        d3.inches = (inches + d2.inches) % 12;
        d3.feet += (inches + d2.inches)/12;
        return d3;
   }
};

void main()
{
    clrscr();
    distance dist_1,dist_2;
    dist_1.distance_assign(12,11)
    dist_2.distance_assign(24,1);
    distance dist_3 = dist_1 + dist_2;
    dist_1.display();
    dist_2.display();
    dist_3.display();
    getch();
}
```

1. பணிமிகுக்கப்பட்ட செயற்குறிகளை அடையாளம் காண்க.
2. பணிமிகுக்கப்பட்ட உறுப்புச் செயற்கூறின் முன்வடிவை எழுதிக் காட்டுக.
3. பணிமிகுக்கப்பட்ட செயற்குறியில் பயன்படுத்தப்பட்டுள்ள செயலேற்பிகளின் இனம் யாது?
4. பணிமிகுக்கப்பட்ட உறுப்புச் செயற்கூறினை இயக்கும் கூற்றினை எழுதிக் காட்டுக.

பணிமிகுக்கப்பட்ட +=, -= ஆகிய செயற்கூறுகளை Program-7.5 விளக்குகிறது.

```
//Program-7.5
// operator overloading

# include <iostream.h>
# include <conio.h>
# include <string.h>

class library_book
{
   char name[25];
   int code,stock;

  public :

   void book_assign(char n[15],int c,int s)
  {
        strcpy(name,n);
        code = c;
        stock = s;
  }

   void display()
  {
         cout << "\n  Book details ....";
         cout << "\n  1. Name       ....." << name;
         cout << "\n  2. Code       ....." << code;
         cout << "\n  3. Stock      ....." << stock;
  }

   void operator +=(int x)
  {
       stock += x;
  }

   void operator -=(int x)
  {
        stock -= x;
  }
};
```

```
class  library_cdrom
{
   char name[25];
   int code,stock;

   public :

   void cdrom_assign(char n[15],int c,int s)
   {
          strcpy(name,n);
          code = c;
          stock = s;
   }

   void display()
   {
           cout << "\n  CD ROM details ....";
           cout << "\n  1. Name       ....." << name;
           cout << "\n  2. Code       ....." << code;
           cout << "\n  3. Stock      ....." << stock;
   }

   void operator +=(int x)
   {
        stock += x;
   }

   void operator -=(int x)
   {
          stock -= x;
   }
};


void main()
{
     library_book book;
     library_cdrom cdrom;

     book.book_assign("Half Blood Prince",101,55);
     cdrom.cdrom_assign("Know your Basics",201,50);

     char choice,borrow;

     do
     {
           cout << "\nBook,cdrom,exit<b/c/e> ...";
           cin >> choice;
           if (choice != 'e')
             {
                  cout << "\nBorrow/Return <b/r> ...";
                  cin >> borrow;
             }
```

```
switch (choice)
{
   case 'b':
      switch (borrow)
      {
         case 'b' : book += 1;break;
         case 'r' : book -= 1;break;
      }
      book.display();
      break;
   case 'c' :
      switch (borrow)
      {
         case 'b' : cdrom += 1;break;
         case 'r' : cdrom -= 1;break;
      }
      cdrom.display();
      break;

   case 'e' : cout << "\nTerminating ..";
              break;
 }
 } while (choice != 'e');
 getch();
}
```

நூலகத்திலுள்ள இருப்புகள், வழக்கமான +=, -= ஆகிய செயற்குறிகளைப் பயன்படுத்தி வழக்கமான பாணியில் மிக எளிதாக மிகுக்கப்படுவதையும், குறைக்கப்படுவதையும் கவனித்தீர்களா?

```
book   += 1;
book   -= 1;
cdrom  += 1;
cdrom  -= 1;
```

ஒரு செயற்குறிக்குப் புதிய பொருளை வழங்கும் செயல்நுட்பமே செயற்குறிப் பணிமிகுப்பு என்றழைக்கப்படுகிறது.

செயற்குறிப் பணி மிகுப்புக்கான விதிமுறைகள்:

செயற்குறிகளின் பணிமிகுப்புக்குச் சில கட்டுப்பாடுகளும் வரம்புகளும் உள்ளன. அவை:

- ✓ ஏற்கெனவே இருக்கும் செயற்குறிகளுக்கு மட்டுமே பணிமிகுக்க முடியும். புதிய செயற்குறிகளை உருவாக்க முடியாது.
- ✓ பணிமிகுக்கப்பட்ட செயற்குறி ஏற்கின்ற செயலேற்பிகளுள் ஒன்று மட்டுமாவது பயனர் வரையறுத்த தரவினமாக இருக்க வேண்டும்.
- ✓ ஒரு செயற்குறியின் அடிப்படை வரையறையை மாற்றியமைக்க முடியாது. அதாவது, ஒரு செயற்குறியின் செயல்பாட்டை மறுவரையறை செய்ய முடியாது. ஒரு செயற்குறிக்குக் கூடுதலான பணிகளை வரையறுக்க முடியும்.
- ✓ பணிமிகுக்கப்பட்ட செயற்குறிகள் அவற்றின் செயலேற்பிகளைப் பொறுத்தமட்டில் அடிப்படைச் செயற்குறிகளைப் போன்றே செயல்படுகின்றன.
- ✓ இருமச் செயற்குறிகளின் (Binary Operators) பணிமிகுக்கும்போது, அச்செயற்குறியின் இடப்பக்கம் அமையும் பொருள், அது வரையறுக்கப்பட்டுள்ள இனக்குழுவின் பொருளாக இருக்க வேண்டும்.
- ✓ உறுப்புச் செயற்கூறு மூலம் இருமச் செயற்குறியைப் பணிமிகுக்கும்போது, ஒரேயொரு வெளிப்படைச் செயலுருபை மட்டுமே ஏற்கும்.

பயிற்சி வினாக்கள்

I. கீழேயுள்ள பணிகளை நிறைவேற்றச் செயற்கூறு பணிமிகுப்பைப் பயன்படுத்தும் ஒரு நிரலை எழுதுக:

அ) இரண்டு முழுஎண்களில் பெரிய எண்ணைக் கண்டறியவும்

ஆ) மூன்று முழுஎண்களில் பெரிய எண்ணைக் கண்டறியவும்

விடை: செயற்கூறு முன்வடிவுகள் max(int, int), max(int, int, int)

II. கீழ்க்காணும் பணிகளுக்குப் பணிமிகுப்பு நுட்பத்தைப் பயன்படுத்திச் செயற்கூறுகளை எழுதவும்:

அ) float இன மாறியின் மதிப்பை ஒன்று மிகுக்க

ஆ) char இன மாறியின் மதிப்பை ஒன்று மிகுக்க

விடை: செயற்கூறு முன்வடிவுகள் - float incr(float), char incr(char)

III. செயற்கூறு பணிமிகுப்பு மூலம் கீழ்க்காணும் பணிகளை நிறைவேற்ற சி++ மொழியில் ஒரு நிரல் எழுதுக:

அ) x, y ஆகியவை int எனில், x^y மதிப்பைக் கண்டறியவும்.

ஆ) x, y ஆகிய இரண்டும் float எனில், x^y மதிப்பைக் கண்டறியவும்.

விடை: செயற்கூறு முன்வடிவுகள் -

int power(int, int), float power (float, float)

IV. செயற்குறிப் பணிமிகுப்பின் ஆதாயங்கள் யாவை?

V. பணிமிகுக்கப்பட்ட செயற்குறியை வரையறுப்பதில் பின்பற்ற வேண்டிய படிநிலைகளைப் பட்டியலிடுக.

VI. பணிமிகுப்புச் செய்யமுடியாத செயற்குறிகளைப் பட்டியலிடுக.

VII. complex_numbers என்னும் இனக்குழுவின் இரண்டு பொருள்களைக் கூட்டிச் சொல்ல நிரல் எழுதுக. ஒரு கலப்பு எண் என்பது மெய்ப்பகுதி, கற்பனைப் பகுதி என்னும் இரண்டு தரவு உறுப்புகளைக் கொண்டிருக்கும். கீழேயுள்ள வரையறையை நிறைவு செய்து, complex_numbers இனத்திலுள்ள c1, c2 என்னும் இரண்டு பொருள்களைக் கூட்டிச் சொல்ல main() செயல்கூறினை எழுதிக் காட்டுக.

```
class complex_numbers
{
    float x;
    float y;
    public :
     void assign_data(float real, float imaginary);
    void display_Data();
    complex_numbers operator +(complex_numbers n1);
}
```

பாடம் 8

ஆக்கிகளும் அழிப்பிகளும்

(Constructors and Destructors)

8.1 முன்னுரை

ஓர் இனக்குழுவின் ***சான்றுரு*** (instance) பயன்பாட்டுக்கு வரும்போது, ***ஆக்கி*** (constructor) எனப்படும் சிறப்புச் செயற்கூறு இயக்கப்படுகிறது. ஆக்கிச் செயற்கூறு இனக்குழுப் பொருளைத் (class object) தொடங்கிவைக்கிறது - அதாவது, அதன் உறுப்புகளில் தொடக்க மதிப்பு இருத்துகிறது. ஓர் இனக் குழுப் பொருளின் பயன்பாடு முடிவுக்கு வரும்போது ***அழிப்பி*** (destructor) எனப்படும் சிறப்புச் செயற்கூறு இயக்கப்படுகிறது. ஆக்கி, அழிப்பி ஆகிய இரண்டு செயற்கூறுகளுமே இனக்குழுவின் பெயரையே கொண்டுள்ளன. இரண்டு செயற்கூறுகளுமே எந்த மதிப்பையும் திருப்பி அனுப்புவதில்லை. எந்தத் தரவினத்தோடும் தொடர்புடையவை அல்ல.

8.2 ஆக்கிகள் (Constructors)

```
// Program - 8.1
// to determine constructors and destructors

#include<iostream.h>
#include<conio.h>
class simple
{
 private:
        int a,b;
 public:
 simple()
 {
        a= 0 ;
        b= 0;
        cout<< "\n Constructor of class-simple ";
 }

 ~simple()
 {
        cout<<"\n Destructor of class - simple .. ";
 }

 void getdata()
 {
        cout<<"\n Enter values for a and b... ";
        cin>>a>>b;
 }

 void putdata()
 {
     cout<<"\nThe two integers .. "<<a<<'\t'<< b;
     cout<<"\n The sum of the variables .. "<< a+b;
 }
};

void main()
{
 simple s;
 s.getdata();
 s.putdata();
}
```

மேற்கண்ட நிரலைச் செயல்படுத்துகையில் s என்னும் பொருள் உருவாக்கப்படும்போது simple() என்னும் ஆக்கி, தானாகவே இயக்கப்படுகிறது. s- ன் பயன்பாடு முடியும்போது, அதாவது நிரலின் செயல்பாடு முடிவுக்கு வரும்போது ~simple() என்னும் அழிப்பி தானாகவே இயக்கப்படுகிறது.

இந்த நிரலின் வெளியீடு இவ்வாறு இருக்கும்:

```
Constructor of class - simple ..
Enter values for a & b… 5 6
The two integers….. 5   6
The sum of the variables….. 11
Destructor of class - simple …
```

8.3 ஆக்கியின் செயல்பாடுகள்

1) இனக்குழுப் பொருளின் உறுப்புகளில் தொடக்க மதிப்பு இருத்துகிறது.

2) பொருளுக்கு நினைவகத்தில் இடம் ஒதுக்குகிறது.

8.4 ஆக்கியின் பணிமிகுப்பு (Constructor Overloading)

ஆக்கிகள் இனக்குழுவின் சிறப்புச் செயற்கூறுகள் என்பதால், செயற்கூறு பணிமிகுப்பு (Function Overloading) ஆக்கிகளுக்கும் பொருந்தும். Program- 8.2 நிரல் ஆக்கிப் பணிமிகுப்பை விளக்குகிறது.

add() என்னும் ஆக்கி ***அளபுருக்கள் இல்லாத ஆக்கி*** (non-parameterized constructor) ஆகும். இது ***தானமைவு ஆக்கி*** (default constructor) எனப் படுகிறது. வழக்கமாக, தானமைவு ஆக்கிகள் ***நிரல்பெயர்ப்பி உருவாக்கும் ஆக்கிகள்*** (compiler generated constructors) எனக் குறிப்பிடப்படுவதுண்டு. அதாவது, பயனர் வரையறுத்த ஆக்கிகள் எதுவும் இல்லாதபோது கணிப் பொறியே இத்தகைய ஆக்கியை உருவாக்கிக் கொள்கிறது. அளபுருக்கள் இல்லாமல் ஒரு பொருள் அறிவிக்கப்படும்போது, அளபுருக்கள் இல்லாத ஆக்கி இயக்கப்படுகிறது.

```
// Program - 8.2
// To demonstrate constructor overloading

# include<iostream.h>
#include<conio.h>
class add
{
       int num1, num2, sum;
       public:
       add()
       {
              cout<<"\n Constructor without parameters.. ";
              num1= 0;
              num2= 0;
              sum = 0;
       }

       add ( int s1, int s2 )
       {
              cout<<"\n Parameterized constructor... ";
              num1= s1;
              num2=s2;
              sum=NULL;
       }

       add (add &a)
       {
              cout<<"\n Copy Constructor ... ";
              num1 = a.num1;
              num2 = a.num2;
              sum  = NULL;
       }

       void getdata()
       {
              cout<<"Enter data ... ";
              cin>>num1>>num2;
       }
       void addition()
       {
              sum=num1+num2;
       }

       void putdata()
       {
              cout<<"\n The numbers are..";
              cout<<num1<<'\t'<<num2;
              cout<<"\n The sum of the numbers are.. "<< sum;
       }
};

void main()
{
       add a, b (10, 20) , c(b);
       a.getdata();
       a.addition();
       b.addition();
       c.addition();
       cout<<"\n Object a :  ";
       a.putdata();
       cout<<"\n Object b :  ";
       b.putdata();
       cout<<"\n Object c : ";
       c.putdata();
}
```

வெளியீடு:
Constructor without parameters….
Parameterized Constructor...
Copy Constructors…

Enter data .. 5 6

Object a:
The numbers are 5 6
The sum of the numbers are ….. 11

Object b:
The numbers are 10 20
The sum of the numbers are … 30

Object c:
The numbers are 10 20
The sum of the numbers are ….. 30

add(int s1, int s2) என்னும் ஆக்கி ***அளபுரு ஏற்கும் ஆக்கி*** எனப் படுகிறது. இந்த ஆக்கியைச் செயல்படுத்த, இரண்டு int மாறிலிகள் அல்லது மாறிகளுடன் பொருளை அறிவிக்க வேண்டும்.

குறிப்பு: char, float, double அளபுருக்களையும் ஏற்கும். அவை உள்ளுறை இனமாற்றத்தின் காரணமாக int தரவினத்தோடு பொருந்தும் என்பதால் ஏற்றுக் கொள்ளப் படக் கூடியவையே.

எடுத்துக்காட்டு: add a(10, 60) / add a(ivar, ivar)

add (add &a) என்னும் ஆக்கி, ***நகல் ஆக்கி*** (copy constructor) எனப்படுகிறது. நகல் ஆக்கி கீழ்க்காணும் சூழல்களில் இயக்கப்படும்:

1) ஏதேனும் ஓர் உறுப்புச் செயற்கூறினுக்கு ஒரு பொருளை அளபுருவாக அனுப்பிவைக்கும்போது.
எடுத்துக்காட்டு: void add : : outdata (add x);

2) ஓர் உறுப்புச் செயற்கூறு ஒரு பொருளைத் திருப்பியனுப்பும் போது.
எடுத்துக்காட்டு: add getdata();

3) ஒரு பொருள் குறிப்புவகை அளபுருவாக ஆக்கிக்கு அனுப்பி வைக்கப்படும்போது.
எடுத்துக்காட்டு: add a; b(a);

கீழேயுள்ள நிரல் (Program - 8.3) நகல் ஆக்கி எப்போதெல்லாம் இயக்கப் படுகிறது என்பதை விளக்குகிறது:

```
// Program – 8.3
// To demonstrate constructor overloading

# include<iostream.h>
#include<conio.h>
class add
{
       int num1, num2, sum;
       public:
       add()
       {
              cout<<"\n Constructor without parameters.. ";
              num 1 = 0;
              num 2 = 0;
              sum   = 0;
       }

       add ( int s1, int s2 )
       {
              cout<<"\n Parameterized constructor... ";
              num1= s1;
              num2=s2;
              sum=NULL;
       }

       add (add &a)
       {
              cout<<"\n Copy Constructor ... ";
              num1= a.num1;
              num2=a.num2;
              sum = NULL;
       }

       void getdata()
       {
              cout<<"Enter data ... ";
              cin>>num1>>num2;
       }
       void addition(add b)
       {
              sum=num1+ num2 +b.num1 + b.num2;
       }

       add addition()
       {
           add a(5,6);
           sum = num1 + num2 +a.num1 +a.num2;
       }

       void putdata()
       {
              cout<<"\n The numbers are..";
              cout<<num1<<'\t'<<num2;
              cout<<"\n The sum of the numbers are.. "<< sum;
       }
};
```

```
void main()
{
      clrscr();
      add a, b (10, 20) , c(b);
      a.getdata();
      a.addition(b);
      b = c.addition();
      c.addition();
      cout<<"\n Object a :  ";
      a.putdata();
      cout<<"\n Object b :  ";
      b.putdata();
      cout<<"\n Object c.. ";
      c.putdata();
}
```

```
ouput of the above program
Constructor without parameters..
 Parameterized constructor...
 Copy Constructor ... Enter data ... 2 3

 Copy Constructor ...
 Parameterized constructor...
 Parameterized constructor...
 Object a :
 The numbers are..2     3
 The sum of the numbers are.. 35
 Object b :
 The numbers are..0     1494
 The sum of the numbers are.. 0
 Object c..
 The numbers are..10    20
 The sum of the numbers are.. 41
```

நகல் ஆக்கி எத்தனைமுறை இயக்கப்படுகிறது என்பதைக் கவனித்தீர்களா?

Program - 8.3 நிரலில் கீழ்க்காணும் கூற்றுகள் நகல் ஆக்கியை இயக்குகின்றன:

add c(b); // b என்னும் பொருள் செயலுருபாக அனுப்பிவைக்கப் படுகிறது.

a.addition(b); // b என்னும் பொருள் addition() என்னும் உறுப்புச் செயற்கூறினுக்கு செயலுருபாக அனுப்பிவைக்கப்படுகிறது.

b = c.addition(); // addition() என்னும் உறுப்புச் செயற்கூறு ஒரு பொருளைத் திருப்பியனுப்புகிறது.

மேற்கண்ட எடுத்துக்காட்டில் addition() என்னும் செயற்கூறின் பணி மிகுக்கப்பட்டுள்ளது (overloaded). நிரலினுள் எந்த இடத்தில் அறிவிக்கப்படும் செயற்கூறுகளும் பணிமிகுக்கப்பட முடியும் என்பதறிக.

8.5 ஆக்கி வரையறுப்பு மற்றும் பயன்படுத்தலுக்கான விதிமுறைகள்:

1) ஆக்கியின் பெயர் இனக்குழுவின் பெயராகவே இருக்க வேண்டும்.

2) ஆக்கி, அளபுருக்களின் பட்டியலைக் கொண்டிருக்க முடியும்

3) ஆக்கிச் செயற்கூறு, பணிமிகுக்கப்பட முடியும்.

4) ஆக்கி எதையும் பயனர் வரையறுக்காதபோது, நிரல்பெயர்ப்பி ஓர் ஆக்கியை உருவாக்கிக் கொள்ளும்.

5) ஓர் இனக்குழுப் பொருள் உருவாக்கப்படும்போது, ஆக்கியானது தானாகவே இயக்கப்படும். அழைக்க வேண்டியதில்லை.

8.6 அழிப்பிகள் (Destructors)

அழிப்பி என்பது, ஒரு பொருளை உருவாக்கும்போது, ஆக்கியால் பொருளுக்கென ஒதுக்கப்படும் நினைவகப் பகுதியை விடுவிக்கும் ஒரு செயற்கூறாகும். இதுவும் இனக்குழுவின் பெயரையே கொண்டிருக்கும். ஆனால், ~ என்னும் குறியைப் பெயரின் முன்னொட்டாகக் கொண்டிருக்கும்.

எடுத்துக்காட்டு:

```
class simple
{
        ...............
        ...............
        public :
        ~simple()
        {
            ................
        }
}
```

8.7 அழிப்பி வரையறுப்பு மற்றும் பயன்படுத்தலுக்கான விதிமுறைகள்:

1) அழிப்பியின் பெயரானது, ~ என்ற முன்னொட்டுக் குறியுடன் கூடிய இனக்குழுவின் பெயரையே கொண்டிருக்கும்.

2) அழிப்பி, செயலுருபுகளை ஏற்காது.

3) அழிப்பி, எந்த மதிப்பையும் திருப்பி அனுப்பாது.

4) அழிப்பியின் பணிமிகுக்கப்பட முடியாது. அதாவது, ஓர் இனக்குழுவில் ஓர் அழிப்பி மட்டுமே இருக்க முடியும்.

5) பயனர், அழிப்பியை வரையறுக்காதபோது, நிரல்பெயர்ப்பி ஓர் அழிப்பியை உருவாக்கிக் கொள்ளும்.

6) நிரலில் உருவாக்கப்பட்ட ஓர் இனக்குழுப் பொருளின் பயன்பாடு முடிவுக்கு வரும்போது, அழிப்பி தானாகவே இயக்கப்படும். அழைக்க வேண்டியதில்லை.

பயிற்சி வினாக்கள்:

I கீழ்க்காணும் அட்டவணையை நிறைவு செய்க:

வினாக்கள்	ஆக்கி	அழிப்பி
1. எந்த வரையெல்லைக்குள் (scope) அறிவிக்கப்பட வேண்டும்?		
2. பணிமிகுப்பு சாத்தியமா?		
3. எப்போது இயக்கப்படுகிறது? ஓர் இனக்குழுப் பொருளானது....		
4. செயலுருபுகளை ஏற்குமா?		

II கீழ்க்காணும் நிரல்பகுதிகள் பிழைசுட்டுவது ஏன்?

```
class simple
{
   private:
   int x;
   simple()
   {
     x = 5;
   }
};
```

```
class simple
{
   private:
   int x;
   public:
   simple(int y)
   {
     x = y;
   }
};
void main()
{
  simple s;
}
```

```
class simple
{
    private:
    int x;
    public:
    simple(int y)
    {
       x = y;
    }

    simple(int z=5)
    {
       x = z;
    }
};
void main()
{
  simple s(6);
}
```

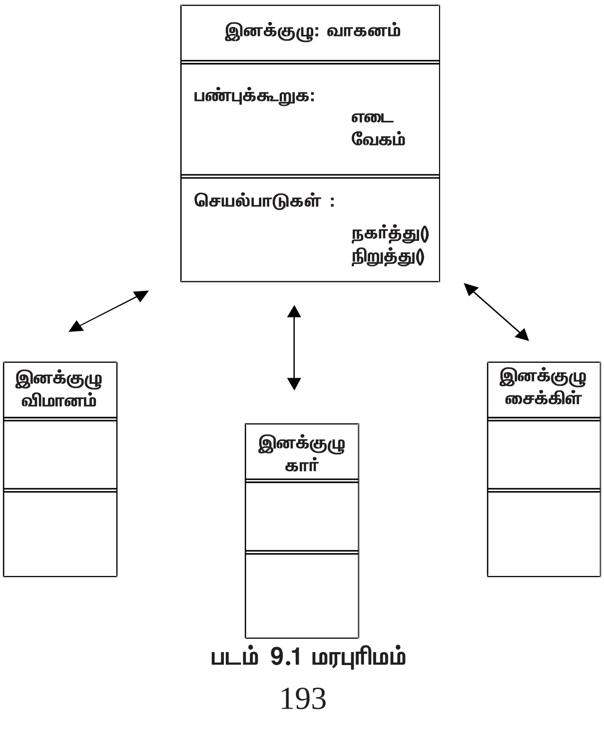
பாடம் 9

மரபுரிமம்

(Inheritance)

9.1 முன்னுரை

ஒரு பொருள்நோக்கு நிரலாக்க மொழியின் சக்திவாய்ந்த பண்புக்கூறு ***மரபுரிமம்*** (Interitance) ஆகும். மரபுரிமம் என்பது ஏற்கெனவே இருக்கும் இனக் குழுக்களின் அடிப்படையில் புதிய இனக் குழுக்களை உருவாக்கும் செயல் முறை ஆகும். இவை முறையே, ***அடிப்படை இனக்குழுக்கள்*** (base classes) எனவும், ***தருவிக்கப்பட்ட இனக்குழுக்கள்*** (derived classes) எனவும் அழைக்கப் படுகின்றன. தருவிக்கப்பட்ட இனக்குழு, அடிப்படை இனக்குழுவின் அனைத்துப் பண்புகளையும் மரபுரிமையாகப் பெற்று விடுகிறது. அதுமட்டுமின்றி, தருவிக்கப் பட்ட இனக்குழுவில் கூடுதலான பண்புக்கூறுகளையும் வழிமுறைகளையும் சேர்த்து அதன் செயலாற்றலை அதிகரிக்க முடியும். 'மரபுரிமம்' என்னும் சொல் நடைமுறை வாழ்க்கையில் நமக்கு மிகவும் அறிமுகமான சொல். குழந்தைகள் தமக்கே உரிய பண்பியல்புகளுடன் பெற்றோரின் பண்புகளை மரபு வழியாகப் பெறுகின்றன. அதுபோலவே, ஓர் இனக்குழுவும் அதன் அடிப்படை (Parent) இனக்குழுவிடமிருந்து பண்புகளை மரபுரிமையாகப் பெறுகிறது.



படம் 9.1 மரபுரிமம்

9.2 மரபுரிமத்தின் பலன்கள்

மரபுரிமம் கீழ்க்காணும் பலன்களை அளிக்கிறது:

1) நிரல் குறிமுறையின் மறுபயனாக்கம் (Reusability of Code);
ஒரு நிறுவனத்தில் பல்வேறு பயன்பாட்டு மென்பொருள்கள் உருவாக்கப்படுகின்றன. ஒரு பயன்பாட்டுக்கென எழுதப்பட்ட குறி முறைத் தொகுதியை, அதே செயலாக்கம் வேண்டப்படுகின்ற வேறொரு பயன்பாட்டில் பயன்படுத்திக்கொள்ள முடியும். இதனால் நிரல் உருவாக்கும் நேரம் வெகுவாகக் குறைகிறது.

2) நிரல் குறிமுறைப் பகிர்வு (Code Sharing);
அடிப்படை இனக்குழுவின் வழிமுறைகளை(methods) தருவிக்கப்பட்ட இனக்குழுக்கள் பகிர்ந்துகொள்ள முடியும்.

3) இடைமுகத்தின் முரண்பாடின்மை (Consistency of Interface);
மரபுரிமையாகப் பெறும் பண்புக் கூறுகளும் வழிமுறைகளும் அழைக் கும் வழிமுறைகளுக்கு ஒரே மாதிரியான இடைமுகத்தையே வழங்கு கின்றன. படம்1-ல் கண்டுள்ளபடி, ***வாகனம்*** என்னும் இனக்குழுவின் பண்புக்கூறுகளும், வழிமுறைகளும், ***விமானம், கார், சைக்கிள்*** ஆகிய மூன்று தருவிக்கப்பட்ட இனக்குழுக்களுக்கும் பொதுவாக அமைந்துள்ளன. இந்த மூன்று தருவிக்கப்பட்ட இனக்குழுக்களுமே முரண்பாடற்ற இடைமுகங்களைக் கொண்டுள்ளன எனலாம்.

9.3 தருவிக்கப்பட்ட இனக்குழுவும் அடிப்படை இனக்குழுவும்

ஓர் இனக்குழுவிலிருந்து பிற இனக்குழுக்கள் தருவிக்கப்படுமாயின் அதனை அடிப்படை இனக்குழு என்கிறோம். அடிப்படை இனக்குழுவின் உறுப்புகளைத் தருவிக்கப்பட்ட இனக்குழு மரபுரிமையாகப் பெறுகிறது.

தருவிக்கப்பட்ட இனக்குழுவை வரையறுக்கும்போது, கீழ்க்காணும் விதிமுறைகளைப் பின்பற்ற வேண்டும்:

அ) class என்னும் சிறப்புச்சொல் இருக்க வேண்டும்.

ஆ) class என்ற சொல்லை அடுத்து, தருவிக்கப்படும் இனக்குழுவின் பெயர் இடம்பெற வேண்டும்.

இ) ஒற்றை முக்காற்புள்ளி (:) இடம்பெற வேண்டும்.

ஈ) private, public அல்லது protected ஆகியவற்றுள் எத்தகைய அணுகியல்புடன் தருவிக்கப்படுகிறது எனக் குறிப்பிட வேண்டும்.

உ) அடிப்படை இனக்குழுவின் பெயரைக் குறிப்பிட வேண்டும்.

ஊ) தருவிக்கப்படும் இனக்குழுவுக்குரிய கூடுதல் பண்பு களையும், வழிமுறைகளையும் வரையறுக்க வேண்டும்.

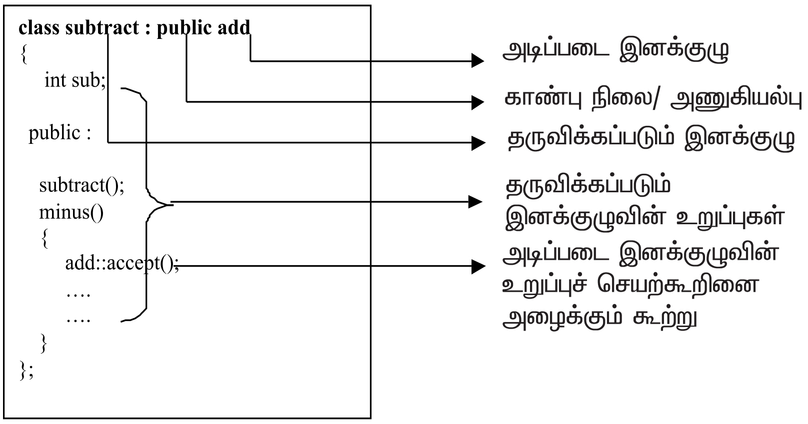
```
// Program - 9.1

#include< iostream.h>
#include<conio.h>
class add
{
      int sum;
      protected:
            int num1, num2;
      public:
      add()
      {
         num1 = num2 = sum  = 0;
         cout<<"\n Add constructor .. ";
      }

      accept()
      {
         cout<<"\n Enter two numbers .. ";
         cin>>num1>>num2;
      }

      plus()
      {
         sum = num1 + num2;
         cout<<"\n The sum of two numbers is .. "<< sum;
      }
};

class subtract :public add
{
      int sub;
      public:
      subtract()
      {
         sub = 0;
         cout<<"\n Subtract constructor .. ";
      }
      minus()
      {
         add::accept();
         sub= num1-num2;
         cout<<"\n The difference of two numbers are ..."
                                                << sub;
       }
};
```

```
void main()
{
      subtract s;
      int choice = 0;
      cout<<"\n Enter your choice ";
      cout<<"\n1. Add..\n2. Subtract ..";
      cin>>choice;
      switch(choice)
      {
            case 1:
                  s.accept();
                  s.plus();
                  break;
            case 2:
                  s.minus();
                  break;
      }
}
```

Program -9.1- ல் add என்பது அடிப்படை இனக்குழு. subtract என்பது தருவிக்கப்பட்ட இனக்குழு. தருவிக்கப்பட்ட இனக்குழு இவ்வாறு வரையறுக்கப் பட வேண்டும்:

class < பெயர்> : < அணுகியல்பு> < அடிப்படை இனக்குழுவின் பெயர்>
{
    தருவிக்கப்படும் இனக்குழுவின் தரவு உறுப்புகள்;
    தருவிக்கப்படும் இனக்குழுவின் உறுப்புச் செயற்கூறுகள்;
}

Program- 9.1- ல் substract என்னும் தருவிக்கப்பட்ட இனக்குழு இவ்வாறு வரையறுக்கப்பட்டுள்ளது:



9.4 காண்புநிலைப் பாங்கு / அணுகியல்பு வரையறுப்பி

(Visibility Mode / Accessibility Specifier)

மரபுரிமத்தின் குறிப்பிடத்தக்க கூறு என்னவெனில், தருவிக்கப்பட்ட இனக்குழுவின் பொருள்களும் உறுப்புகளும் அடிப்படை இனக்குழுவின் உறுப்புகளை எப்போது, எப்படிப் பயன்படுத்திக் கொள்ள வேண்டும் என்பதுதான். இதுவே, ***அணுகியல்பு*** (accessibility) எனப்படுகிறது. private, public, protected என மூன்று ***அணுகியல்பு*** வரையறுப்பிகள் உள்ளன. அணுகியல்பு வரையறுப்பியை ***காண்புநிலைப் பாங்கு*** (visibility mode) என்றும் கூறலாம். முன்னியல்பான காண்புநிலைப் பாங்கு private ஆகும். தருவிக்கப்பட்ட இனக்குழுவில், அடிப்படை இனக்குழு உறுப்புகளின் வரையெல்லை மற்றும் அணுகியல்பை அட்டவணை 9.1 விளக்குகின்றது.

அடிப்படை இனக்குழுவின் உறுப்புகள்	**தருவிக்கப்பட்ட இனக்குழு**		
	private	protected	public
private உறுப்புகள்	மரபுரிமையாகப் பெறப்படுவதில்லை. ஆனாலும் அவை தொடர்ந்து நிலவும்.		
protected உறுப்புகள்	private அணுகியல்பு கொண்ட உறுப்புகளாக மரபுவழி பெறப் படுகிறது.	protected அணுகியல்பு கொண்ட உறுப்புகளாக மரபுவழி பெறப் படுகிறது.	protected அணுகியல்பு தக்க வைத்துக் கொள்ளப் படுகிறது.
public உறுப்புகள்	private அணுகியல்பு கொண்ட உறுப்புகளாக மரபுவழி பெறப்படுகிறது.	protected அணுகியல்பு கொண்ட உறுப்புகளாக மரபுவழி பெறப் படுகிறது.	public அணுகியல்பு கொண்ட உறுப்பு களாகவே பெறப் படுகிறது.

அட்டவணை 9.1 தருவிக்கப்பட்ட இனக்குழுவில் அடிப்படை இனக்குழு உறுப்புகளின் வரையெல்லையும் அணுகியல்பும்

ஓர் அடிப்படை இனக்குழு private என்னும் அணுகியல்புடன் தருவிக்கப் படும்போது, அடிப்படை இனக்குழுவின் public மற்றும் protected உறுப்புகள், தருவிக்கப்படும் இனக்குழுவில் private உறுப்புகளாகவே நிலவுகின்றன.

ஓர் அடிப்படை இனக்குழு protected என்னும் அணுகியல்புடன் தருவிக்கப் படும்போது, அடிப்படை இனக்குழுவின் protected மற்றும் public உறுப்புகள், தருவிக்கப்படும் இனக்குழுவில் protected உறுப்புகளாக நிலவுகின்றன.

ஓர் அடிப்படை இனக்குழு public என்னும் அணுகியல்புடன் தருவிக்கப் படும்போது, அடிப்படை இனக்குழுவின் protected உறுப்புகள், தருவிக்கப்படும் இனக்குழுவில் protected உறுப்புகளாகவும், public உறுப்புகள் public உறுப்பு களாகவும் நிலவுகின்றன.

> அடிப்படை இனக்குழுக்கள் public, protected, private என எப்படித் தருவிக்கப்பட்டாலும் அடிப்படை இனக்குழுவின் private உறுப்புகள் தருவிக்கப்பட்ட இனக்குழுவில் மரபுரிமையாகப் பெறப்படுவதில்லை. தருவிக்கப்பட்ட இனக்குழுவில் அவை நிலவும். ஆனால் அவற்றை அணுக முடியாது.

Program -9.1- ல் add, subtract ஆகிய இனக்குழுக்கள் இவ்வாறு அறிவிக்கப் பட்டுள்ளன:

```
class add
{
     private:
          int sum;
     protected :
          int num1, num2;
     public:
          add();
          accept();
          plus();
};
class subtract : private add
{
     int sub;
     public:
     subtract();
     minus();
};
```

subtract இனக்குழு மரபுவழிப் பெற்ற தரவு உறுப்புகளும், உறுப்புச் செயற்கூறுகளும்:

int num1 & num2 - subtract- ல் இவை private

accept(), plus() - subtract- ல் இவை private

I

```
class subtract : private add
{
        int sub;
  public:
      subtract( );
      minus();
};
```

II

```
class subtract : protected add
{
        int sub;
  public:
      subtract();
      minus();
};
```

III

```
class subtract : private add
{
        int sub;
  public:
      subtract();
      minus();
};
```

அடிப்படை இனக்குழு உறுப்புகளின் அணுகியல்பு அட்டவணை 9.2-ல் தரப்பட்டுள்ளன.

அடிப்படை இனக்குழுவின் ஆக்கிகள் (constructors), தருவிக்கப்பட்ட இனக்குழுவில் மரபுவழி பெறப்படுவதில்லை. ஆனால், தருவிக்கப்பட்ட இனக் குழுவில் ஒரு சான்றுரு (instance) உருவாக்கப்படும்போது முதலில் இயக்கப் படுகின்றன.

class subtract	II	III
private:	private:	private:

sub
num1
num2
accept()
plus()
public:
subtract()
minus();
private mode
of inheritance
	sub
	protected:
	num1. num2
	accept();
	plus();
	public:
	subtract()
	minus()
	protected mode
	of inheritance
		sub
		protected:
		num1. num2
		public:
		accept();
		plus();
		subtract()
		minus();
		public mode
		of inheritance

அட்டவணை 9.2 அடிப்படை இனக்குழு உறுப்புகளின் அணுகியல்பு

Program - 91- ல் அறிவிக்கப்பட்டுள்ள பொருள்களை நோக்குக. இந்த நிரலின் அடிப்படையில் கீழ்க்காணும் அட்டவணையை நிறைவு செய்க:

இனக்குழுவின் பொருள்கள்	**தரவு உறுப்புகள்**	**வழிமுறைகள்/ செயற்கூறுகள்**
add		
subtract		

அட்டவணை 9.3 பொருள்களின் தரவு/ செயற்கூறு உறுப்புகள்

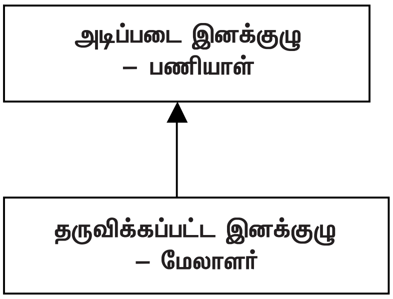
இயக்கப்படும் ஆக்கிகள்:--------------------, ----------------------.

9.5 மரபுரிமத்தின் வகைகள்

தருவிக்கப்பட்ட இனக்குழுக்களை அடிப்படையாகக் கொண்டு புதிய இனக் குழுக்களைத் தருவிக்க முடியும். மரபுரிமத்தில் பலவகை உள்ளன: ஒருவழி மரபுரிமம், பலவழி மரபுரிமம், பலநிலை மரபுரிமம், கலப்பு மரபுரிமம், படிமுறை மரபுரிமம்.

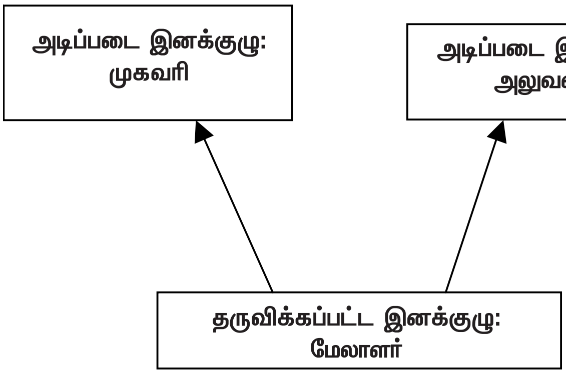
1) ஒருவழி மரபுரிமம் (Single Inheritance)

ஒரேயொரு இனக்குழுவை அடிப்படையாகக் கொண்டு தருவிக்கப் பட்ட இனக்குழுவை உருவாக்குவது ஒருவழி மரபுரிமம் ஆகும்.



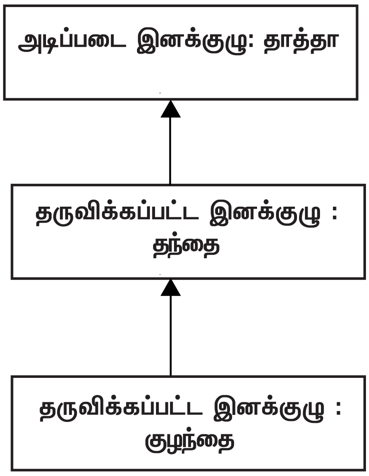
2) பலவழி மரபுரிமம் (Multiple Inheritance)

ஒன்றுக்கு மேற்பட்ட அடிப்படை இனக்குழுக்களிலிருந்து தருவிக்கப்பட்ட இனக்குழுவை உருவாக்குவது பலவழி மரபுரிமம் ஆகும்.



3) பலநிலை மரபுரிமம் (Multilevel Inheritance)

தருவிக்கப்பட்ட ஓர் இனக்குழுவை அடிப்படையாகக் கொண்டு இன்னொரு தருவிக்கப்பட்ட இனக்குழுவை உருவாக்குவது பலநிலை மரபுரிமம் ஆகும்.



Program - 9.2 நிரலின் வெளியீடு என்னவாக இருக்கும்?

```
// Program - 9.2
#include<iostream.h>
#include<conio.h>
class base
{
      public:
      base()
      {
            cout<<"\nConstructor of base class... ";
      }
      ~base()
      {
      cout<<"\nDestructor of base class.... ";
      }
};
class derived:public base
{
  public :
  derived()
  {
    cout << "\nConstructor of derived ...";
  }
  ~derived()
  {
      cout << "\nDestructor of derived ...";
  }
};

class derived2:public base
{
  public :
  derived()
  {
    cout << "\nConstructor of derived2 ...";
  }
  ~derived()
  {
      cout << "\nDestructor of derived2 ...";
  }
};

void main()
{
  derived2 x;
}
```

வெளியீடு:

Constructor of base class…

Constructor of derived ….

Constructor of derived2 …

Destructor of derived2…

Destructor of derived

Destructor of base class ..

> ✓ ***ஆக்கிகள்***, மரபுரிமம் பெற்ற இனக்குழுக்களின் வரிசையில் இயக்கப்படுகின்றன. அதாவது, அடிப்படை இனக்குழுவில் தொடங்கி இறுதியாகத் தருவிக்கப்பட்ட இனக்குழு வரையில். ***அழிப்பிகள்*** முன்பின் வரிசையில் இயக்கப்படும்.

பிற இனக்குழுக்களைத் தருவிப்பதற்காக மட்டுமே பயன்படும் இனக் குழுக்கள் அருவ இனக்குழுக்கள் அல்லது ***கருத்தியல் இனக்குழுக்கள்*** (Abstract Classes) எனப்படுகின்றன. அதாவது, கருத்தியல் இனக்குழுக்களில் பொருள்களை உருவாக்க முடியாது.

பயிற்சி வினாக்கள்

1) கீழ்க்காணும் வரையறுப்புகளை அடிப்படையாகக் கொண்டு, அட்டவணை -4,5,6 ஆகியவற்றை நிறைவு செய்க:

```
class node
{
        int x;
        void init();
        public:
        void read();
        protected:
        void get();
};
class type : public node
{
        int a;
        public:
        void func1();
        protected:
        int b;
        void getb();
}
```

```
class statement :private type
{
        int p;
        public:
        void func2();
        protected:
        void func3();
};
```

type **இனக்குழுவின் உறுப்புகள்**	**உறுப்புகளின் அணுகியல்பு**		
	private	protected	public
type இனக்குழுவில் மரபுவழிப் பெற்ற உறுப்புகள்			
type இனக்குழுவில் வரையறுக்கப் பட்டவை			

அட்டவணை 4 type **இனக்குழு**

statement **இனக்குழுவின் உறுப்புகள்**	**உறுப்புகளின் அணுகியல்பு**		
	private	protected	public
statement இனக் குழுவில் மரபுவழிப் பெற்ற உறுப்புகள்			
statement இனக் குழுவில் வரை யறுக்கப்பட்டவை			

அட்டவணை 5 statement **இனக்குழு**

பொருள்கள்	**அணுகமுடியும் உறுப்புகள்**	
	தரவு உறுப்புகள்	**உறுப்புச் செயற்கூறுகள்**
type இனக்குழு		
statement இனக்குழு		

அட்டவணை –6 பொருள்கள்

2) கீழ்க்காணும் நிரலில் உள்ள பிழைகளைச் சுட்டி, காரணம் கூறுக:

```
#include<iostream.h>
class A
{
        private :
                int a1;
        public:
                int a2;
        protected:
                int a3;
};
class B : public A
{
                public:
                void func()
                {
                        int b1, b2 , b3;
                        b1 = a1;
                        b2 = a2;
                        b3 = a3;
                }
};
void main()
{
                B der;
                der.a3 = 0;
                der.func();
}
```

3) கீழ்க்காணும் அறிவிப்புகளை நோக்குக. கீழுள்ள வினாக்களுக்கு விடை தருக:

```
class vehicle
{
        int wheels;
        public:
                void inputdata( int, int);
                void outputdata();
        protected :
                int passenger;
};
```

```
class heavy_vehicle : protected vehicle
{
        int diesel_petrol;
        protected:
                int load;
        public:
                void readdata( int, int);
                void writedata();
};
class bus: private _heavy_vehicle
{
        char marks[20];
        public:
                void fetchdata( char );
                void displaydata();
};
```

அ. heavy.vehicle என்னும் இனக்குழுவின் அடிப்படை இனக்குழுவையும் தருவிக்கப்பட்ட இனக்குழுவையும் குறிப்பிடுக.

ஆ. displaydata() என்னும் செயற்கூறு மூலம் அணுக முடிகிற தரவு உறுப்புகளைக் குறிப்பிடுக.

இ. bus இனக்குழுவின் பொருள், அணுக முடிகிற தரவு உறுப்புகளைக் குறிப்பிடுக.

ஈ. outputdata() என்னும் செயற்கூறினை, heavy_vehicle இனக்குழுவின் பொருள்களால் அணுக முடியுமா?

4. கீழ்க்காணும் நிரலின் வெளியீடு என்னவாக இருக்கும்?

```
#include<iostream.h>
#include<conio.h>
class student
{
        int m1, m2, total;
    public:
        student ( int a, int b)
        {
                m1 = a;
                m2 = b;
            cout<<"\n Non parameterized constructors..";
        };
}
```

பாடம் 10

சமுதாயத்தின் மீது கணிப்பொறியின் தாக்கம்

(Impact of Computers on Society)

10.1 முன்னுரை

“இந்தியா அதன் ஏழு லட்சம் கிராமங்களில் வாழ்கிறது”

-மோகன்தாஸ் கரம்சந்த் காந்தி

தகவல் தொழில்நுட்பத்தின் பலன்கள் சாதாரண மனிதர்களையும் சென்றடைய வேண்டுமெனில் குறைந்தபட்சம் மூன்று தொழில்நுட்பக் கூறுகள் நமக்குத் தேவை:

- இணைப்பாக்கம் (கணிப்பொறிப் பிணையங்கள் மற்றும் இணைய வசதி)
- விலை மலிவான கணிப்பொறிகள் அல்லது அதுபோன்ற சாதனங்கள்
- மென்பொருள்

நம் கவனத்தை ஈர்க்கும் தகவல் என்னவெனில், இன்றைய கணிப்பொறிப் பயன்பாடுகளில் 85% ***சொல் செயலாக்கம்*** (Wrod Processing) என்பதே. ஆனால், கணிப்பொறியானது சாதாரண மனிதர்க்கு இதைவிட அதிகமாகப் பயன்பட முடியும். தரம் மிக்க தகவல் தொழில்நுட்பக் கல்வி, சாதாரண மனிதர்களும் கணிப்பொறியைச் சிறந்த முறையில் பயன்படுத்த உதவும். சமுதாயத்தை மேம்படுத்தக் கணிப்பொறிகளைப் பயன்படுத்தும் வழிமுறைகளை இந்தப் பாடத்தில் காண்போம்.

10.2 சொந்தப் பயன்பாட்டுக்குக் கணிப்பொறிகள்

சொந்தக் கணிப்பொறிகள் (Personal Computers) நமது வேலை முறை யையும், வாழ்க்கை முறையையும், ஏன் சிந்தனையையும்கூட முற்றிலும் மாற்றிவிட்டது. சொல் செயலாக்கம் (Word Processing), தரவுத் தளங்கள் (Databases), விரிதாள்கள் (Spread Sheets), பல்லூடக முன்வைப்புத் தொகுப்பு கள் (Multimedia Presentation Packages) - ஆகியவை பணிகளில் நமது செயல்திறனை அதிகரித்துள்ளன. இன்னும் எத்தனையோ மென்பொருள் தொகுப்புகள் பயன்பாட்டில் உள்ளன. கணிப்பொறிப் பதிப்பாக்கம் (Desktop Publishing) மற்றும் வரைகலைக்கான மனங்கவர் தொகுப்புகள் நாம் செய்யும் பணியின் மதிப்பைக் கூட்டியுள்ளன. ஓவியம் (Paint), விளையாட்டுகள் (Games) மற்றும் இவை போன்ற ஏராளமான மென்பொருள் தொகுப்புகள் சிறியவர் முதல் முதியவர் வரை அனைத்து வயதினரும் கணிப்பொறியைப் பயன்படுத்துவதற்கான வசதியை நல்குகின்றன. இணைய உலா (Browsing), மின்னஞ்சல் (e-mail), அரட்டை(chat) ஆகியவை நமது வாழ்க்கை முறையை மாற்றிவிட்டன.

இன்றைக்குக் கணிப்பொறிகள் பல்வேறு அளவுகளில், வடிவங்களில் கிடைக்கின்றன. அடிப்படைக் கணிப்பொறியை பயனரின் வீடுகளில் மிக்க பயனுள்ளதாக்கும் வகையில் தகவமைத்துக்கொள்ள முடியும்.

10.3 கணிப்பொறி மயமாக்கப்பட்ட வீடுகள்

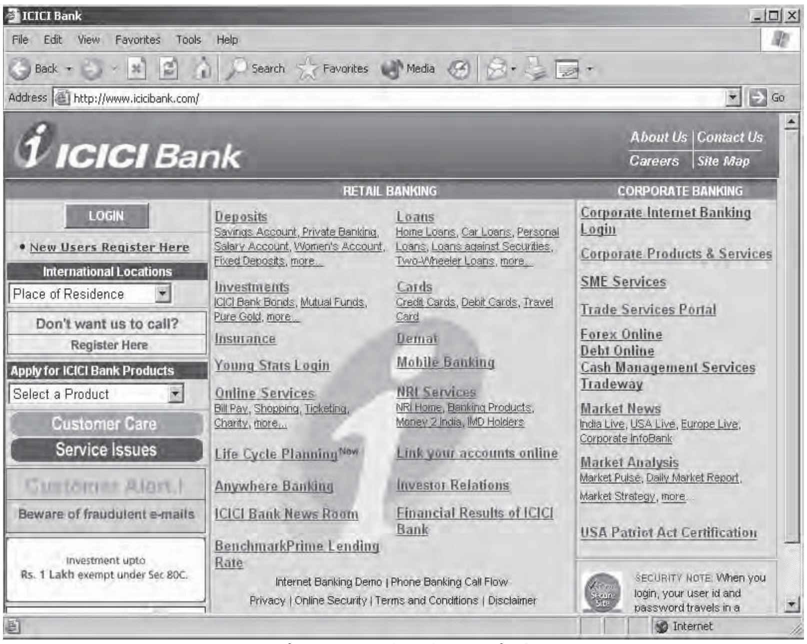
வீடு	தயாரிப்புப் பொருள்களும் சுருக்கமான விளக்கமும்
வரவேற்பறை	✹ எல்சிடி திரை, ஒளிப்படக்கருவி (கேமரா), ஒலி பெருக்கிகள்: சுவரில் பொருத்தப்பட்டிருக்கும். வெளிப்பாடு சிறப்பாய் இருக்கும். தரையிடம் மிச்சமாகும். ✹ தரவுக் காப்பகம் : தரவுகளைச் சேமித்து வைக்கவும் மேலாண்மை செய்யவும் உதவும். ✹ உணர்ச்சிக் கொள்கலன்கள்: சிறிய பெட்டிகள் - ஒரு திரையையும் ஒலிபெருக்கியையும் கொண்டிருக்கும். உணர்வுக்கு இதமூட்டும் நறுமணம் வீசும். மக்களைத் தீய பழக்கவழக்கத்துக்கு ஆட்படாமல் தடுக்கும். ✹ சொந்தத் தரவுக் காப்பகம் : குடும்ப ஒளிப்படங்கள் போன்ற சொந்தத் தகவல்கள் மற்றும் பிற சொந்த அரும்பொருட் செல்வங்கள் போன்றவற்றைச் சேமித்து வைக்கலாம். அதோடுகூட பிற மனிதர்களோடு இணைப்பையும் கொண்டிருக்கும். ✹ படப்பேசியும் கையேடும் : படங்களோடு கூடிய தொலைபேசி எண்களடங்கிய திரட்டு.

குழந்தைகள் அறை	✸ வீட்டிலுள்ள வானொலி, தொலைக்காட்சிப் பெட்டி யிலிருந்து வரும் பாடல்களைக் கேட்கும் வசதி கொண்ட சாதனங்கள். ✸ பின்னணி இசையோடு இணைந்து பாடக்கூடிய 'கரோக்கி' வசதியுள்ள சாதனங்கள் ✸ மின்-வளர்ப்பு விலங்குகளாகச் செயல்படும் ரோபோக்கள் ✸ இணையத்தில் விளையாட்டுகளை ஆட உதவும் சாதனங்கள்... இவை தவிர நடப்பு வாழ்வின் பாத்திரங்களைக் கணிப்பொறி உலகின் பாத்திரங்களாக மாற்றி அவற்றோடு குழந்தைகள் விளையாட முடியும்.
வீட்டு அலுவலகம்	✸ அசைவூட்டக் கதைகளை உருவாக்குவதற்கான மென் பொருள்கள் ✸ நினைவுச் சட்டகம்: தொடுதிரை, வருடி, நுண்பேசி வழியாகப் பிற மக்களோடு எளிதாக ஊடாட உதவும். ✸ புத்தக அலமாரி: மின்னணு நூல்களைப் படிக்கவும், பாதுகாக்கவும் பயன்படும் ✸ சொந்தப் படைப்பாக்கக் கருவி: படம், ஒலி, அசைவூட்டம், நிகழ்படம் போன்ற பல்லூடகத் தகவலுடன் பணியாற்றுதல். உயர்நிலை தரவு அணுகல் முறைமைகள்.
படுக்கை அறை	✸ பல்வேறு சாதனங்களுக்கான தொடு/குரல் கட்டுப்பாடுகள் ✸ காட்சித்திரைகள், தனிச்சிறப்பான தலைபேசிகள், நகர்பேசிகள் ✸ படம்பெருக்கி டிவி: தொலைக்காட்சிப் படங்களை மேற்கூரை அல்லது சுவர்களில் பெரிதாக்கிக் காட்டும். ✸ விழிப்பூட்டு கடிகாரம்
குளியலறை	✸ ஆடிகள், மருந்துப் பெட்டி, தனிச்சிறப்பான ஒலிபெருக்கிகள்
சமையலறை	✸ ஒலிபெருக்கிகள், அலமாரித் தொலைபேசி, நுண்ணறிவுள்ள காப்புத்துணி, கெட்டில், டோஸ்டர், உணவுப் பகுப்பாய்வி, உடல்நலக் கண்காணிப்பி, உணவுப் பாதுகாப்பு சாதனங்கள்
உணவுக்கூடம்	✸ உணவுப் பொருட்களைச் சூடாக வைத்திருக்கும் ஊடாடு மேசைத் துணி. ✸ பீங்கானில் செய்த கேட்பொலி இயக்கிகளும் ஒலிபெருக்கிகளும் ✸ உணவு மேசையைச் சுற்றிலும் தகவல் தொடர்பு வசதிகள் ✸ சமையல்காரர் மற்றும் சமையலறைப் பணியாளர்களிடம் உரையாட முடிகிற ஊடாடு திரைகள்.

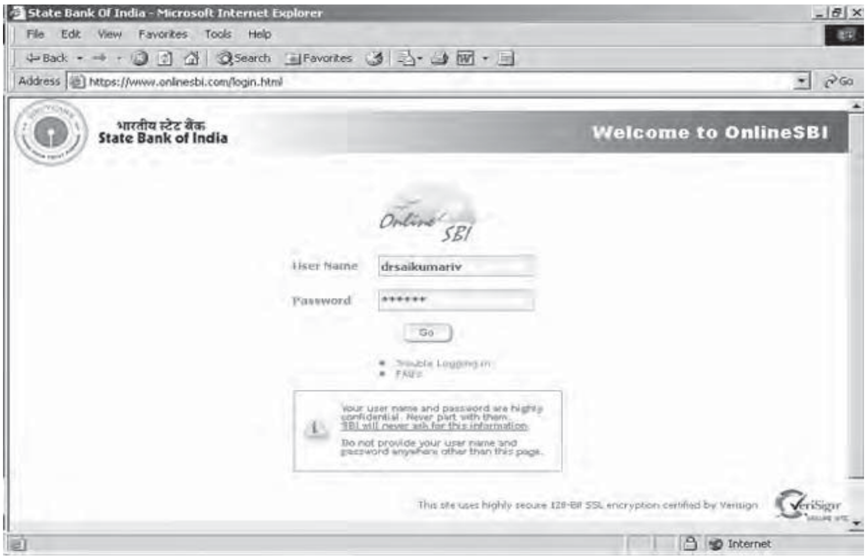
10.4 வீட்டிலிருந்தே வங்கிப் பணியும் கடைப்பொருள் வாங்கலும்

மரபு வழியான வங்கிப் பணியில் பயனாளர் பணப் பரிமாற்றம் போன்ற செயல்பாடுகளை மேற்கொள்ள வங்கிக்கு நேரடியாகச் செல்லவேண்டும். பணப் பரிமாற்றம் என்பது பணத்தைச் சேமிப்பில் போடுவது அல்லது சேமிப்பிலிருந்து பணத்தை எடுத்தல் அல்லது கடன் பெறுதல் போன்ற நடவடிக்கையாக இருக்கலாம். இப்போதெல்லாம் வங்கிகள் வேறுபல சேவைகளையும் வழங்குகின்றன. குறித்த கால வைப்புநிதிகள், வேளாண்மைக் கடன்கள், பாதுகாப்புப் பெட்டகங்கள் மற்றும் தொலைபேசி, மின்சாரம் போன்ற, பிற சேவைகளுக்குக் கட்டணம் செலுத்துதல் ஆகிய வசதிகள் உள்ளன. வங்கிச் செயல்பாடுகளில் சாதாரண மனிதனின் நம்பிக்கை மேம்பட்டுள்ளது. இதன் காரணமாய், தேசியப் பொருளாதாரத்தில் வங்கிகள் முக்கிய கூறாக மாறிவிட்டன.

பணி நேரங்களில் வங்கிகளில் மிக நீண்ட வரிசை நிற்பதைக் காணலாம். வங்கிகளில் தகவல் தொழில்நுட்பம் அறிமுகப்படுத்தப்பட்டதால், பயனாளருக்குச் சேவை வழங்குவதற்குத் தேவையான நேரம் வெகுவாகக் குறைந்துள்ளது. மிக நீண்ட வரிசைகளும் மிக வேகமாகக் கையாளப் படுகின்றன.



படம் 10.1 ஐசிஐசிஐ வங்கி



படம் 10.2 பாரத ஸ்டேட் வங்கி

ஒரு குறிப்பிட்ட வங்கியின் கணக்கிலிருந்து எந்த நேரத்திலும் எந்த இடத்திலும் பணம் எடுத்துக்கொள்ள ஏடீஎம் (ATM- Automatic Teller Machine) போன்ற உயர்தொழில்நுட்ப எந்திரங்கள் பயன்படுகின்றன. பயனாளருக்கு இரவிலோ, விடுமுறை நாட்களிலோ அவரசமாகப் பணம் தேவைப்படும் சூழ் நிலைகளில் இவை உதவுகின்றன. என்றாலும், பயனாளர் அருகிலுள்ள ஏடீஎம் மையத்துக்கு நேரில் செல்ல வேண்டும். மக்கள் பரவலாகப் பயன்படுத்தும் வங்கிகளின் ஒவ்வொரு கிளையும் ஏடீஎம் மையங்களை நிறுவும் நாள் வெகுதொலைவில் இல்லை.

மின்-வங்கிச் சேவை (e-Banking), மக்கள் வீட்டிலிருந்தபடியே இணையத் தின் வழியாக வங்கிச் சேவைகளை நுகர வாய்ப்பளித்துள்ளது. இது, வங்கிகளின் வீச்சையும் சேவைகளையும் மெய்யாகவே மேம்படுத்தியுள்ளது எனலாம்.

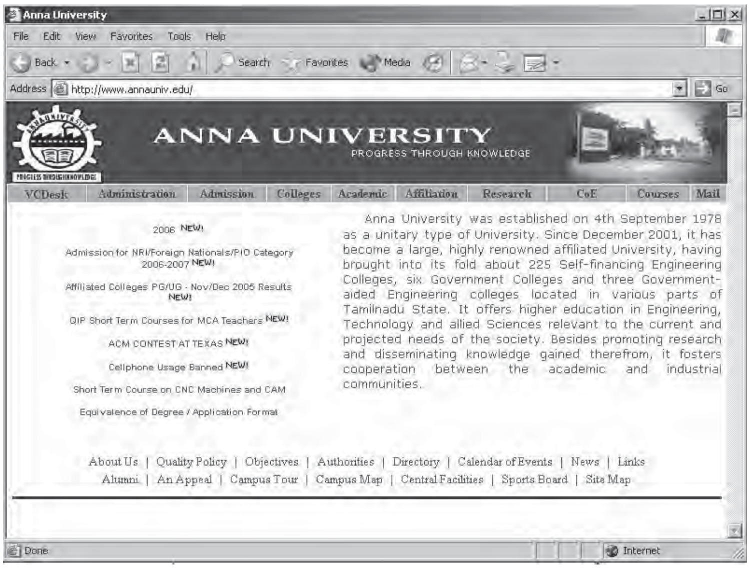
கணிப்பொறிகள் பல்வேறு பணிகளுக்கும் பயன்படுத்தப்படுகின்றன. கடையில் பொருள்கள் வாங்கவும்கூட பயன்படுத்த முடியும். மின்-கடைச்செலவு (e-shopping) முறையில் எந்தப்பொருளையும், எந்த நிறுவனத்தின் பொருளை யும், எவ்வளவு வேண்டுமானாலும், எந்த இடத்திலிருந்தும் வாங்கிக்கொள்ள முடியும். நீங்கள் கடைக்கு நேரில் செல்ல வேண்டியதில்லை. கடையின் வலையகத்தில் (website) நீங்கள் வாங்க விரும்பும் பொருட்களின் படம் மற்றும் பிற விவரங்களைக் காணலாம். அவற்றில் தேவையானவற்றைத் தேர்ந் தெடுத்து, அனுப்பிவைக்குமாறு கோரிக்கை அனுப்பலாம். பொருள்களுக்கான விலையை முன்கூட்டியே செலுத்துவதற்குப் பல்வேறு வழிமுறைகள் உள்ளன.

கடன் அட்டைகள் (Credit cards), கடைக்காரரிடம் முன்கூட்டியே பதிவு செய்தல் போன்ற வழிமுறைகள் பரவலாகப் பயன்பாட்டில் உள்ளன. வாங்கப்பட்ட பொருட்கள் வீட்டிற்கு வந்து வினியோகிக்கப்படும்.

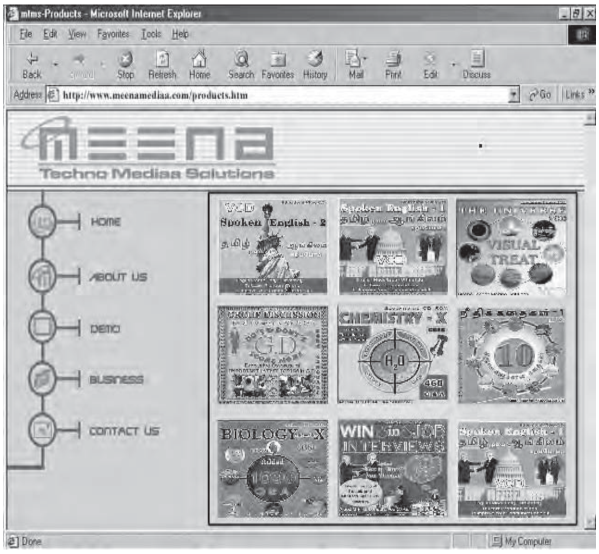
10.5 கல்வியில் கணிப்பொறிகள்

கல்வியில் பல பிரிவுகளிலும் கணிப்பொறிகள் பயன்படுத்தப்படுகின்றன. அவற்றுள் சில:

- உள்நாட்டு/வெளிநாட்டு நூலாசிரியர்களின் கல்விக் குறுவட்டு களையும், புத்தகங்களின் புதிய பதிப்புகளையும் இணையத்தின் வழியே தேடி அறிந்து அவற்றை வாங்குதல்.
- கணிப்பொறி வழியான பாடப் பயிற்சிகள். (Computer Based Tutorials - CBT).
- பள்ளிகள், கல்லூரிகள், பல்கலைக் கழகங்கள், அவை வழங்கும் படிப்புகள், அவற்றில் சேர்வதற்கான விதிமுறைகள், விடுதி வசதிகள், கல்வி உதவித் தொகைகள், கல்விக் கடன்கள், வேலைவாய்ப்புக்கான வழிகாட்டுதல்- இவை பற்றிய தகவல்களைப் பரப்புதல்.
- மின்-கற்றல் (e-Learning) முறையில் இணையத்தின் வழியே கல்வி கற்றுப் பட்டங்களும் சான்றிதழ்களும் பெற முடியும்.



படம் 10.3 அண்ணா பல்கலைக் கழகம்



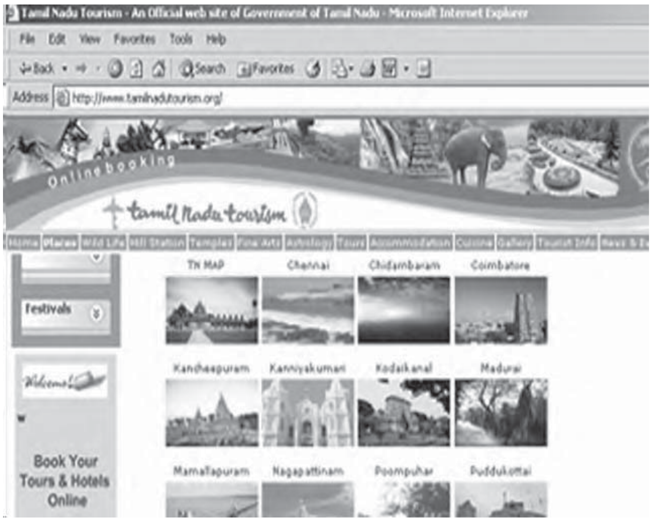
படம் 10.4 கணிப்பொறிவழி பாடப் பயிற்சிகள்

படம் 10.5 உலகளாவிய கல்வி பற்றிய தகவல்

10.6 பொழுதுபோக்கில் கணிப்பொறிகள்

பொழுதுபோக்கிலும் கணிப்பொறிகளின் பங்களிப்பு உள்ளது. கணிப்பொறி மற்றும் இணையம் மூலமாக ஓவியம், இசை, நடனம், யோகா, விளையாட்டு கள், அறிவியல், இயற்கை, சூடான செய்திகள் மற்றும் நிகழ்வுகள் பற்றிய உங்கள் அறிவைப் புதுப்பித்துக்கொள்ள முடியும். பல்வேறு வழிபாட்டுத் தலங்கள் பற்றியும், சுற்றுலாத் தலங்கள் பற்றியும் நிறையவே அறிந்துகொள்ள முடியும்.

படம் 10.6 பொழுதுபோக்கில் கணிப்பொறிகள்



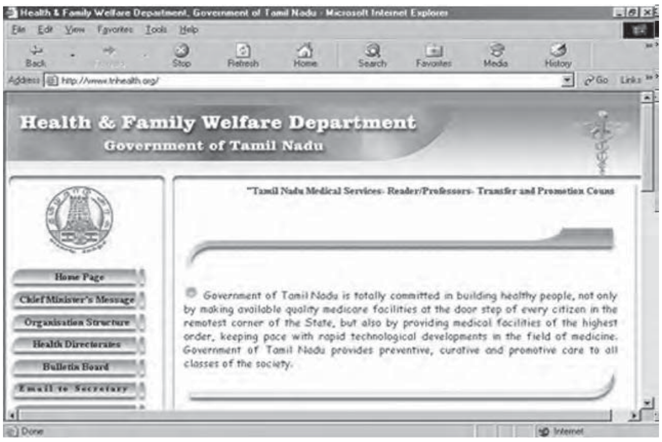
படம் 10.7 சுற்றுலாத் துறையில் கணிப்பொறிகள் (தமிழ்நாடு)

படம் 10.8 சுற்றுலாத் துறையில் கணிப்பொறிகள் (இந்தியா)

10.7 நலவாழ்வுக்குக் கணிப்பொறிகள்

நலவாழ்வுத் துறையில் ஏராளமான தரவுகள் குவிந்து கிடக்கின்றன. அவற்றைப் பராமரிக்கக் குறைந்த அளவு நிதி வளமும் மனித வளமுமே உள்ளன. அவற்றைக் கட்டிக் காப்பதற்கான தேவை அதிகரித்துள்ளது.

மருத்துவர்களும் மருத்துவத் துறை நிர்வாகிகளும் கணிப்பொறிகளைப் பயன்படுத்தத் தொடங்கிய பிறகு நலவாழ்வுச் சேவைகள் மேம்பட்டுள்ளன.



படம் 10.9 நலவாழ்வு மற்றும் குடும்பநலத் துறையில் கணிப்பொறிகள் (இந்தியா)

நலவாழ்வுத் துறையின் பல பிரிவுகளிலும் கணிப்பொறிகள் பயன் படுத்தப்படுகின்றன. அவற்றுள் சில:

- மருத்துவமனை மேலாண்மை முறைமை
- நோயாளிகள் தொடர்சேவை முறைமை
- நோயாய்வு ஆவணங்களை நலவாழ்வு நிலையங்களுக்கிடையே பரிமாறிக் கொள்ளல்

* தொற்று நோய் பற்றி முழுவிவரம் அறிதலும் கண்காணித்தலும்
* உயர்நிலைக் கணிப்பணி நுட்பங்களுடன் கூடிய தீர்மானிப்பு உதவி முறைமை

இன்றைக்குப் பெரும்பாலான மருத்துவர்கள் தமது தேவைகளுக்கு ஏற்ற வகையில் நவீன வழிமுறைகளைப் பின்பற்றுகின்றனர். நோயாளிகளுக்கு இது ஒரு நல்ல அறிகுறியாகும். தொலைமருத்துவம் (Tele-medicine) என்பது, பெரும்பாலும் மேலே குறிப்பிடப்பட்டுள்ள அடிப்படை முறைமைகளின் மீதே கட்டமைக்கப்பட்டுள்ளது. சேய்மையிலிருந்து கொண்டே நோயாய்வு செய்ய இணையம் வழி வகுக்கிறது. வல்லுநர் ஆலோசனை கிடைக்கமுடியாத இடங்களிலும் அத்தகைய ஆலோசனை கிடைப்பதை இணையம் உறுதி செய்கிறது.

10.8 வேளாண்மைத் துறையில் கணிப்பொறிகள்

வேளாண்மைத் துறை, தொழில்நுட்பம் சாராத துறைபோலத் தோன்றலாம். சாதாரணமாக இத்துறையை நோக்குபவர்கள் எண்ணுவதை விடவும் கணிப்பொறிமயமாக்கத்தால் இத்துறை மிகவும் பலனடைந்துள்ளது. செய்தொழிலாகவும், விருப்பார்வமாகவும் வேளாண்மையில் ஈடுபட்டுள்ள உழவர்கள், விதைகள் பற்றிய கணிப்புகள் மற்றும் பூச்சிக் கொல்லிகள் பற்றிய தகவல்கள் அடங்கிய நிகழ்நிலைத் தகவல் வளங்களான இணைய தளங்கள் மூலம் பெரிதும் பயனடைந்துள்ளனர். தட்பவெப்பநிலை, மண்வளம், தற்போதைய சந்தை விலைகள் ஆகியவற்றின் அடிப்படையில் எந்த விளைபொருள் அதிக அளவு லாபத்தைத் தரும் என்பதைக் கணித்துச் சொல்லும் வருமானக் கணிப்புத் தகவல்கள், உழவர்களுக்குப் பெரிதும் பயன்படுகின்றன.

வேளாண்மைத் துறையில் மென்பொருள் உருவாக்கப்பட்டுள்ள பிரிவுகள்:

* வேளாண்மை நிதியங்கள் மற்றும் கணக்கியல்
* மாற்று விவசாய நுட்பங்கள்
* கால்நடை வளர்ச்சி
* கட்டடங்கள் மற்றும் நீர்ப்பாசனம்

படம் 10.10 வேளாண்மையில் கணிப்பொறிகள்

- பிற வேளாண்மைத் துறையினர் மற்றும் விஞ்ஞானிகளுடன் கலந்துரையாடல்
- பண்ணை நில மதிப்பீடுகள்
- உரப் பகுப்பாய்வு
- இணையத்தில் பண்ணை வளங்கள், அரட்டைக் குழுக்கள், வகையின விளம்பரங்கள், பிற பண்ணை தொடர்பான தகவல்கள் - இவற்றுக்கான தொடுப்புகளைக் கண்டறிதல்
- தோட்டக் கலை
- பசு இனத்தை மேம்படுத்தி வருமானத்தைப் பெருக்குதல்
- நில மேலாண்மை
- பால் உற்பத்தி
- செயற்கைக்கோள் படங்களைப் பயன்படுத்தி விளைபயிர் பற்றிய முடிவுகளை மேற்கொள்ளுதல்

10.9 நிகழ்நேரப் பயன்பாடுகளில் இணையம்

மேற்சொல்லப்பட்ட அனைத்துப் பயன்பாடுகளும் நிகழ்நேரத்தில் இணையத்தில் கிடைப்பவை. ரயில், விமானப் பயணச் சீட்டுகளை வீட்டில் அமர்ந்தபடியே நிதானமாக இணையம் மூலம் முன்பதிவு செய்துகொள்ள முடியும்.

படம் 10.11 நிகழ்நேரப் பயன்பாடுகள்

பயிற்சிகள்

இந்தப்பாடத்தில் விளக்கப்பட்ட பயன்பாட்டு மென்பொருள்கள் பற்றி மேலும் புரிந்துகொள்ள உதவும், பல்லூடக விளக்கப் படங்கள் தனியாக ஒரு குறுவட்டில் உங்கள் பள்ளிக்குத் தரப்பட்டுள்ளது. அதன் உள்ளடக்கத்தை நீங்கள் கட்டாயம் பார்க்கவேண்டும். இக்குறுவட்டைப் பெற உங்கள்ஆசிரியரை அணுகுக. பாடத்தில் குறிப்பிடப்பட்டுள்ள இணைய தளங்களையும் பார்வையிடுங்கள்.

பாடம் 11

தகவல் தொழில்நுட்பம் சார்ந்த சேவைகள்
(IT Enabled Services)

11.1 முன்னுரை

பல்வேறு துறைகளில் சேவைகளின் தரத்தை மேம்படுத்திக் கொள்ளப் பயனாளர்களுக்குத் தகவல் தொழில்நுட்பம் உதவுகிறது. இத்தகைய சேவைகளைத் ***தகவல் தொழில்நுட்பம் சார்ந்த சேவைகள்*** (IT Enabled Services - ITES) என்கிறோம். இத்தகு சேவைகள் பெரும்பாலும் அதிகமான மனிதர்களை ஈடுபடுத்துகின்ற சேவைகளாகும். தொலைத்தொடர்புப் பிணையங்கள் அல்லது இணையம் வழியாக இச்சேவைகள் பலதரப்பட்ட வணிகப் பிரிவுகளுக்கும் வழங்கப்படுகின்றன. தகவல் தொழில்நுட்பம் சார்ந்த சேவைகள் பெருமளவு வேலைவாய்ப்புகளை அதிகரித்துள்ளன.

கணிப்பொறியைப் பயன்படுத்தி ஒரு கடிதத்தைத் தட்டச்சு செய்வது ஐடீஇஎஸ் சேவையில் அடங்குமா? இல்லை என்பதே பதிலாகும். எனினும், பயனாளர் ***டிக்டாஃபோன்*** (Dictaphone) என்னும் சிறப்புச் சாதனத்தில் பேசி, அப்பேச்சினை அப்படியே உரைவடிவில் கடிதமாக மாற்றும் பணி ஐடீஇஎஸ்-இல் அடங்கும்.

சொல்செயலிகள் (Wordprocessors), விரிதாள்கள் (Spreadsheets), தரவுத் தளங்கள் (Databases) ஆகியவை, பல்வேறு மரபுவழிப்பட்ட சேவைகள் தகவல் தொழில்நுட்பம் சார்ந்தவையாக மாற வழி வகுத்துள்ளன. எனினும், பயனாளர், இத்தகைய தகவல் தொழில்நுட்பக் கருவிகளைப் பயன்படுத்திப் பலன்களைப் பெறவேண்டுமாயின் அவற்றின் பல்வேறு கூறுகளையும் கற்றுக்கொள்ள வேண்டும். ஐடீஇஎஸ், பயனாளர்கள் இவற்றைக் கற்க வேண்டிய தேவையைக் குறைக்கின்றது. அதன்மூலம் அவற்றின் மதிப்பைக் கூட்டுகின்றது. ஆக, தகவல் தொழில்நுட்பத்தை அதிகம் கற்றறியாத பயனாளர்களும் தகவல் தொழில்நுட்பத்தின் சக்தி முழுவதையும் கைவரப்பெற ஐடீஇஎஸ் உதவும் வல்லமையைப் பெற்றுள்ளது.

ஐடீஇஎஸ் சேவையின் தரத்தை நேரடியாகவோ, மறைமுகமாகவோ மேம்படுத்துகிறது. வாடிக்கையாளர்களின், கூடுதலான திருப்தி, சிறந்த தோற்றமும் உணர்வும், மேம்பட்ட தரவுத்தளம் ஆகியவை நேரடியான பலன்களில் சிலவாகும். மறைமுகப் பலன்கள் சிறிதுகாலம் கழித்துக் கிடைக்கப்பெறும். ஒரு பயனுக்கென சேகரிக்கப்பட்ட தரவுகள் வேறுபல தேவைகளுக்கும் சிறிது காலத்துக்குப் பின் பயன்படும்

இந்தப் பாடத்தில் தரப்பட்டுள்ள தகவல் தொழில்நுட்பம் சார்ந்த சேவைகளுள் சில:

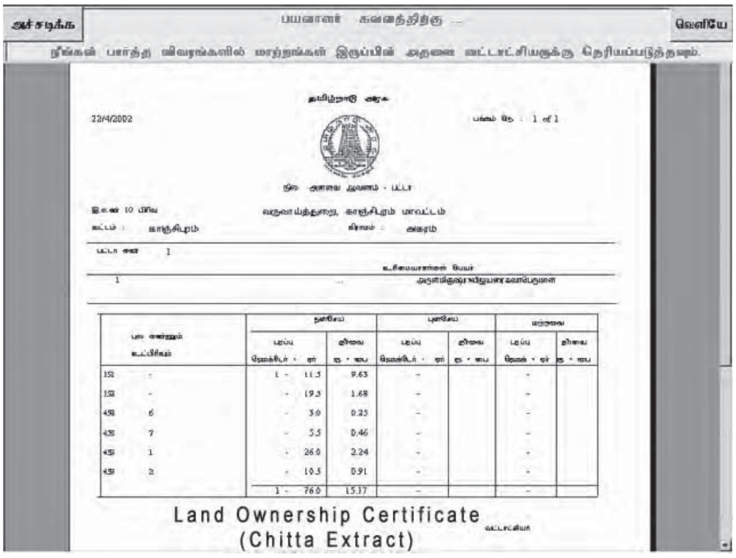
- மின்-அரசாண்மை (e-Governance)
- அழைப்புதவி மையங்கள் (Call Centres)
- தரவு மேலாண்மை (Data Management)
- தொலை மருத்துவம் (Telemedicine)
- மருத்துவ ஆவணப் பெயர்ப்பு (Medical Traanscription)
- கணிப்பொறித் தரவாக்கம் (Data Digitization)
- வலையகச் சேவைகள் (Website Services)

வணிகச் செயலாக்கப் புறத்திறனீட்டம் (Business Process Outsourcing - BPO), துடிமத் தகவல் உருவாக்கம் (Digital Content Development), அசைவூட்டம் (Animation), மனிதவளச் சேவைகள் (Human Resources Services), சேய்மைப் பராமரிப்பு (Remote Maintenance) ஆகியவை ஐடீஇஎஸ் ஊடுருவியுள்ள பிற சேவைகளாகும்.

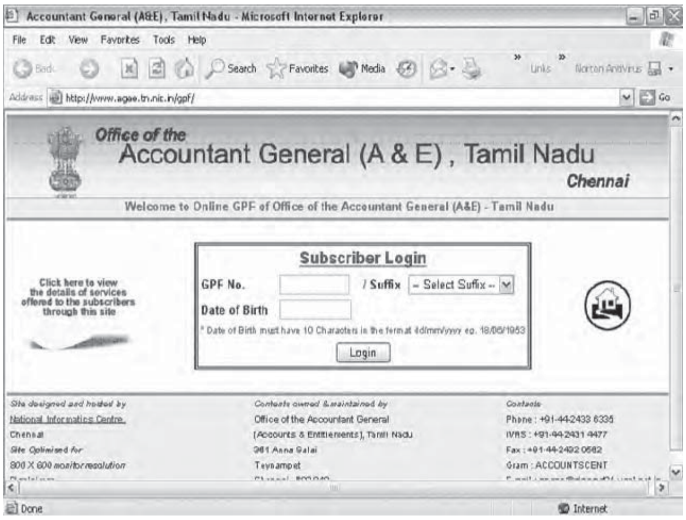
ஐடீஇஎஸ் பணிகளுக்கு, தகவல் தொழில்நுட்பத்தில் குறிப்பாக தரவுத் தளங்களிலும், இணையத்திலும் செய்முறை அறிவுத்திறன் தேவையாகும். தவிரவும், ஆங்கிலத்தில் சிறந்த தகவல் தொடர்புத்திறன் (Communication skill) இருக்க வேண்டிய கட்டாயத் தேவையாகும். ஐடீஇஎஸ் சேவைகளை திறம்பட நடைமுறைப்படுத்துவதற்கு, தொழிலகப் பயன்பாடு, செய்தொழில் நுட்பம் (Professionalism), நற்பழக்கம் (etiquette) ஆகியவற்றின் அடிப்படைகளை நன்கு புரிந்துக் கொள்ளக்கூடிய அகத்திறனில் (Softskill) முறையான பயிற்சி அளிக்கப்படவேண்டும்.

11.2 மின்–அரசாண்மை (e-Governance)

நீங்கள் அரசு வலையகங்களைப் பார்வையிடவும் அவை வழங்கும் சேவைகளைப் பெறவும் கணிப்பொறிகள் உங்களுக்கு உதவுகின்றன. அரசு அமைத்துள்ள பல்வேறு வலையகங்கள் பல பயனுள்ள தகவல்களைக் கொண்டுள்ளன. அரசுத் துறைகள் பற்றி விவரங்கள், குறிப்பிட்ட பணிகள், சிறப்பு நலத்திட்டங்கள், ஆவணங்கள், தொடர்புகள், தொடுப்புகள், ஐஏஎஸ் அக இணையம், வலையக வரைபடம், தேடல், புதியது என்ன, பத்திரிகைக் குறிப்புகள், கருத்தாய்வு - ஆகியவற்றைக் கொண்டுள்ளன. இந்த வலையகங்கள் ஆங்கிலம், தமிழ் ஆகிய இரு மொழிகளிலும் உள்ளன.



படம் 11.1 இணையத்தின் மூலமாக நிலச் சான்றிதழ் பெறுதல்



படம் 11.2 தலைமைக் கணக்காயர் (ஏ&இ) அலுவலகம், தமிழ் நாடு அரசு

11.3 அழைப்புதவி மையங்கள் (Call-Centres)

தகவல் தொழில்நுட்பத்தின் தாக்கம் உலகம் முழுவதும் காணப்படுகிறது. தகவல் தொழில்நுட்பச் சேவைகளின் பயனாளர்கள் அல்லது வாடிக்கையாளர்கள் உலகம் முழுவதிலும் நிறைந்துள்ளனர். ஆண்டில் 365 நாட்களும், நாளில் 24 மணிநேரமும் தகவல் தொடர்பு சேவையை நிறுவனங்கள் வழங்கவேண்டும் என வாடிக்கையாளர்கள் எதிர்பார்க்கின்றனர்.

அழைப்புதவி மையம் என்பது, சில வேளைகளில், குறிப்பிட்ட வாடிக்கையாளர் சேவைக்காக தொலைபேசி அடிப்படையில் அமைந்த சேவைப் பகிர்வு மையம் என வரையறுக்கப்படுகிறது. காரணம் இம் மையங்கள், சந்தைப் படுத்தல் (Marketing), விற்பனை (Selling), தகவல் பரிமாற்றம், வாடிக்கையாளர்-தொடர்பான பணிகளுக்குப் பயன்படுத்திக் கொள்ளப்படுகின்றன. ஒவ்வோர் அழைப்புதவி மையமும் போதுமான அளவு தொலைதொடர்பு வசதிகளையும், பயிற்சிபெற்ற ஆலோசகர்களையும் கொண்டுள்ளன. மிகப்பரந்த தரவுத் தளங்கள், இணையம் மற்றும் பிற நிகழ்நேரத் தகவல் உதவி மையங்களையும், அணுகும் வசதிகளைப் பெற்றுள்ளன. வாடிக்கையாளர்களுக்குத் தகவலையும், உதவிச் சேவைகளையும் அழைப்புதவி மையங்கள் வழங்குகின்றன. இவை ஆண்டில் அனைத்து நாட்களிலும், நாளில் அனைத்து நேரங்களிலும் (அதாவது 24x365 சேவை) செயல்படுகின்றன.

11.4 தரவு மேலாண்மை (Data Management)

தரவு மேலாண்மை என்பது, ஐடீஇஎஸ் சேவையில் ஒரு வகை. பல்வேறு மூலங்களிலிருந்து பெறப்படும் தரவுகளைத் திரட்டுவது அவற்றைக் கணிப்பொறியில் சேமிப்பது, பின் செயலாக்குவது - ஆகிய பணிகளை உள்ளடக்கியதாகும். மரபுவழி, தரவுச் செயலாக்கச் சேவை என்பது, கையெழுத்தில் நிரப்பப்பட்ட படிவங்களில் உள்ள தரவுகளை கணிப்பொறியில் பதிவது, படங்களையும், அச்சிட்ட வெளியீடுகளையும் கணிப்பொறியில் ஏற்றி அவை அனைத்தையும் ஒன்றாக்கித் தரவுத் தளங்களை உருவாக்குவது - ஆகிய பணிகளை உள்ளடக்கியதாகும். ஆனால், பல்லூடகத் தொழில்நுட்பம் மற்றும் இணையத்தின் வருகையால், தரவுத் தளங்களில் அச்சிட்ட ஆவணங்களை மட்டுமின்றி படங்கள், ஒலி மற்றும் நிகழ்படத் தகவல்களையும் சேமிக்க முடியும். அதே போல, தகவல் வெளியீட்டு ஊடகங்களிலும் பெருமளவு மாற்றங்கள் ஏற்பட்டுள்ளன. வழங்கிக் (Server) கணிப்பொறிகளில் சேமித்து வைக்கப்பட்டுள்ள தரவுத்தளங்களைப் பலரும் பகிர்ந்து கொள்ள முடிகிறது. அச்சிட்ட பதிப்புகளிலும் மற்றும் குறுவட்டுகளிலும் தகவல்கள் வெளியிடப்படுகின்றன. இணையம் வழி பெறப்படும் வினவல்களின் அடிப்படையில் இந்த வெளியீடுகள் அமைகின்றன.

நிறுவனங்களில் செயல்திறனுடன் லாபம் தரத்தக்க வகையில் தகவல் தொழில்நுட்பத்தைப் பயன்படுத்திக் கொள்வதில் தரவு மேலாண்மை முக்கிய

இடம் வகிக்கிறது. இந்த வகைப்பாட்டில் வழங்கப்படும் ஐடீஇஎஸ் சேவைகள் வருமாறு:

- ஆஸ்க்கி (ASCII) உரைவடிவில் தகவல்களைத் தரவுத்தளத்தில் பதிவேற்றல்
- அச்சிட்ட உரை ஆவணங்களைக் கணிப்பொறி ஆவணமாக மாற்றிச் செயலாக்குதல்
- தேவைகேற்ற வகையில் அறிக்கை தயாரித்தல்
- தரவுகள் உள்ளீடு
- தரவுகளைத் திருத்தியமைத்தல்
- ஆவண உருவாக்கம்
- படிவங்களை படிமங்களாகக் குறுவட்டுகளில் பதிதல்
- கையெழுத்து, அச்செழுத்து, காந்த எழுத்து, பட்டைக்குறிமுறை- இதுபோன்ற எந்த வடிவத்தில் இருக்கும் தகவல்களையும் கணிப்பொறித் தகவலாக மாற்றிச் சேமித்தல்
- படங்களை வருடிச் சேமித்தல்
- படங்களை உள்ளீடு செய்தல்
- படங்களைக் கணிப்பொறியில் சேமித்து வைத்தல்; தேவையானபோது தேடி எடுத்தல்
- வெளியீடுகளை ஆய்வு செய்தல்
- புள்ளிவிவரப் பகுப்பாய்வு

தரவு மேலாண்மை அடிப்படையிலான ஐடீஇஎஸ் சேவைகள் மூலமாகப் பெரிதும் பயனடையக்கூடிய நிறுவனங்கள்:

- கணக்குவைப்பு, நிதியியல் சேவைகள் போன்ற பின்னணி அலுவலகச் செயல்பாடுகள்
- வங்கித்துறை
- அரசுத்துறைகள்
- மருத்துவமனைகள்
- காப்பீட்டுத்துறை
- சட்டத்துறை
- உற்பத்தி நிறுவனங்கள்
- நகராட்சிகள்
- காவல்துறை
- பொதுமக்கள் நுகர்சேவைகள்
- பதிப்புத்துறை
- தொலைதொடர்பு
- போக்குவரத்து

மேலே குறிப்பிட்ட ஒவ்வொரு நிறுவனமும் ஐடீஇஎஸ் சேவைகளில் தரவு மேலாண்மையின் முக்கியமான பிரிவுகளில் பரந்த வாய்ப்புகளை உருவாக்கி யுள்ளன. வங்கித்துறை (Banking), நிதியியல் சேவைகள்(Financial Services), காப்பீடு (Insurance) ஆகியவை இணைந்து சுருக்கமாக பிஎஃப்எஸ்ஐ (BFSI) என்று அழைக்கப்படுகின்றன. பிஎஃப்எஸ்ஐ மற்றும் ஓய்வூதியச் சேவைகள் ஐடீஇஎஸ்ஸில் பெருமளவில் வளர்ச்சிபெறும் பிரிவுகளாகும்.

இப்பிரிவுகளில், தரவுப் பாதுகாப்பு, வாடிக்கையாளர் தனிமறை (Privacy) ஆகிய இரண்டும் முக்கியமான கூறுகளாகும். ஐடீஇஎஸ் சேவையாளர் இவற்றுக்கு உத்திரவாதம் அளிக்கவேண்டும். ஐடீஇஎஸ் சேவையாளர் ஒருவரே பல நிறுவனங்களுக்கும் சேவை வழங்க முடியும். எனவே ஒவ்வொரு நிறு வனத்தின் தனிமறைக் கூறுகளுக்கும் ஐடீஇஎஸ் சேவையாளர் உத்திரவாதம் தரவேண்டும். கணிப்பொறி நன்னெறி ஐடீஇஎஸ்ஸின் வெற்றிக்கு இன்றியமை யாததாகும்.

11.5 மருத்துவப் பெயர்ப்பாவணமும் தொலைமருத்துவமும்
(Medical Transcription and Tele-medicine)

மருத்துவப் பெயர்ப்பாவணம் என்பது, ஒரு மருத்துவ நோய் ஆய்வின் முடிவுகளை விளக்குகின்ற ஒரு நிலையான, சட்டபூர்வமான ஆவணம் ஆகும். இதனை இணையம் வழி உடனுக்குடன் எங்கும் அனுப்பிவைக்க முடியும். காப்பீட்டுத் தொகையைப்பெற இது உதவும். மருத்துவப் பெயர்ப்பாவணம் மூன்று முதன்மையான படிநிலைகளைக் கொண்டது. அவை:

படிநிலை–1: ஐடீஇஎஸ்ஸின் இச்சேவையைப் பயன்படுத்திக்கொள்ள விரும்பும் மருத்துவ மனைகள் ஒரு சேவையாளரிடம் ஒப்பந்தம் செய்து கொள்கின்றன. மருத்துவர் ஒரு சிறப்புச் சாதனம் மூலம் அல்லது தொலைபேசி வழியாகத் தன்னுடைய ஆலோசனைகளை வழங்குகிறார். குரல் தகவல் மறுமுனையிலுள்ள வழங்கிக் கணிப்பொறியில் (server) பதிவு செய்யப்படுகிறது.

படிநிலை–2: ஒலித்தகவல் கணிப்பொறித் தகவலாய் மாற்றப்பட்டு ஐடீஇஎஸ் சேவையாளருக்கு அனுப்பப்படுகிறது. பெரும்பாலும் இந்தச் சேவை நிறுவனம் வேறொரு நாட்டில் இருக்கும். அமெரிக்கா போன்ற நாடுகளில் பெயர்ப்பாவணச் சேவையை வழங்குவது, நோயாளிக்கும் மருத்துவமனைக்கும் மிகுந்த செலவுபிடிக்கக் கூடியதாகும். எனவே, செலவு கட்டுப்படியாகும் ஒரு நாட்டில் இப்பணியைச் செய்து முடிப்பதன் மூலம், ஐடீஇஎஸ்ஸின் இப்பிரிவு செலவைக் குறைக்கிறது. கணிப்பொறித் தகவல் மீண்டும் ஒலித்தகவலாக மாற்றப்படுகிறது. பயிற்சிபெற்ற பெயர்ப்பாவண எழுத்தாளர்கள் ஒலித்தகவலைக் காதில் கேட்டு, கணிப்பொறியில் உரைத்தகவலாய் மாற்றித் தருவர். இந்த ஆவணம் மருத்துவர் செய்த நோயாய்வு பற்றிய முறையான ஆவணமாகப் பாதுகாக்கப்படும்.

படிநிலை–3: இவ்வாறு பெயர்த்தெழுதப்பட்ட கோப்புகள் தரக்கட்டுப்பாட்டு வல்லுநர்களுக்கு அனுப்பப்படும். மருத்துவரின் குரல் தகவலைக் கேட்டு, பெயர்ப்பாவணத்துடன் சரிபார்ப்பர். தேவையெனில் திருத்தங்கள் செய்வர். அதன்பின் பெயர்த்தெழுதப்பட்ட அறிக்கைகள் உரை ஆவணங்களாக மருத் துவமனைக்கே திரும்பவும் அனுப்பிவைக்கப்படும். இவை சட்டப்படி செல்லு படியாகும் ஆவணங்களாகும். காப்பீட்டுத் தொகை பெறப் பயன்படுத்திக் கொள்ள முடியும்.

11.6 கணிப்பொறித் தரவாக்கம் (Data Digitization)

கணிப்பொறித் தரவாக்கம் என்பது பிற வடிவங்களில் உள்ள தகவல் களைக் கணிப்பொறியில் கையாள்வதற்கேற்ற தகவலாய் மாற்றியமைப்பதாகும். வரைபடங்கள், கையெழுத்து, நிகழ்படங்கள், ஒலி ஆகிய பல்வேறு வடிங்களில் உள்ள தகவல்களைக் கணிப்பொறித் தகவலாக்க முடியும்.

கணிப்பொறியில் தகவல்களைச் சேமித்து வைப்பதால் பயனாளர் பல்வேறு பலன்களைப் பெற முடியும். தகவலை எளிதாக அணுக முடியும். தேவையான தகவலை எளிதாகத் தேடிக் கண்டறிந்து எடுத்தாள முடியம். தகவல்களைத் திருத்திப் புதுப்பிக்க முடியும். எனினும், கணிப்பொறியில் தகவலைச் சேமித்து வைப்பதை, நீண்டகாலத் தகவல்காப்பு நுட்பமாகப் பயன்படுத்துவது ஆபத்தில் முடியவும் வாய்ப்புண்டு. கணிப்பொறித் தகவலாக்கத் தொழில்நுட்பங்கள் வெகுவிரைவாக மாறி வருகின்றன. தகவல் சேமக்காப்பு (Preservation) என்பது நீண்டகாலச் செயல்நுட்பம். ஆனால் இன்றைய பல தொழில்நுட்பங்கள் மிகக் குறுகிய காலத்தில் பயனொழிந்து போகின்றன. தொழில்நுட்பத்தின் இத்தகைய நிலையற்ற தன்மை, கணிப்பொறித் தகவல்களின் இழப்பில் கொண்டுபோய் விட்டுவிடலாம். தகவலைப் பலகாலம் சேமித்துக் காக்கவேண்டிய நோக்கமே தோற்கடிக்கப்பட்டுவிடும். கணிப்பொறித் தரவாக்கத் தொழில்நுட்பம் பயன் படும் சில பிரிவுகளைக் காண்க:

- ஆண்டறிக்கைகளும் விலைப்பட்டிகளும்
- புத்தகங்கள்
- தரவுத்தள ஆவணக் காப்பகம்
- மின்னஞ்சல் விவரப்பட்டியல்களும் துண்டு வெளியீடுகளும்
- பொறியியல் மற்றும் வடிவாக்கம்
- நிலவியல் தகவல் முறைமை
- திரைப்படங்கள், இசைப்பாடல்கள், மிகுதெளிவு படச் சேமிப்பு
- செய்பொருள்/சேவைக்கான பயிற்சிக் குறிப்பேடுகள்
- ஆய்விதழ்கள் மற்றும் கருத்தரங்குக் கட்டுரைகள்

கணிப்பொறித் தரவாக்கத்தின் படிநிலைகள் வருமாறு:

- வாடிக்கையாளரின் தேவைகளை அறிந்து கொள்ளல்
- அவையே கணிப்பொறித் தரவாக்கத்தின் நோக்கங்களை வரையறுப்பதற்கான அடிப்படைகளாகக் கொள்ளப்படுகின்றன.
- ஒரு முன்னோட்டப் பயன்பாடு உருவாக்கப்படுகிறது.
- முன்னோட்டப் பயன்பாடு ஏற்றுக்கொள்ளப்படுமாயின், வாடிக்கையாளர் வேண்டிக்கொள்கிற முழுமையான கணிப்பொறி தரவாக்கப்பணி எடுத்துக்கொள்ளப்படும்.
- வெவ்வேறு வகைப்பட்ட தரவுகளைக் கணிப்பொறித் தரவாக்க வெவ்வேறு நுட்பங்கள் பயன்படுத்தப்படுகின்றன. மூல ஆவணங்களுடைய கணிப்பொறித் தகவல் வடிவத்தின் தரத்தை மேம்படுத்த பற்பல உயர்நுட்ப மென்பொருள் தொகுப்புகள் கிடைக்கின்றன.
- கணிப்பொறித் தரவாக்கப்பட்ட தகவல்கள் வரிசைப்படுத்தப்பட்டு, எளிதாக அணுகும் வகையில், உள்ளடக்க அட்டவணைகள் தயாரிக்கப்படுகின்றன. கணிப்பொறித் தகவலைச் சேமித்து வைக்க மீவுயர் தொழில்நுட்ப மற்றும் நம்பகத்தன்மையுள்ள சேமிப்பு வசதிகள் பயன்படுத்தப்படுகின்றன.

கணிப்பொறித் தகவலாக்கத்தினால் பல்வேறு பலன்கள் கிடைக்கின்றன. அவற்றுள் முக்கியமான சில பலன்கள்:

- ஆவணங்களை நீண்ட காலம் பாதுகாக்க முடியும்.
- முக்கியமான ஆவணங்களை ஒரே இடத்தில் சேமித்து வைக்கமுடியும்.
- தகவல்களை மிக எளிதாக அணுகி, எளிதாகப் பயன்படுத்த முடியும்.
- தேவையான தகவல்களை, குறிப்பாகப் படங்கள் மற்றும் உரைப்பகுதிகளை விரைவாகவும், மிகச் சிறப்பாகவும் தேடிக் கண்டறிய முடியும்.
- படங்கள் மற்றும் உரைத்தகவல்களை மிக எளிதாகப் பரிமாறிக்கொள்ள முடியும்.
- குறுவட்டுகள், இணையம் மற்றும் பிற மின்னணு ஊடகங்கள் மூலமாக மிக எளிதாகத் தகவல் பரிமாற்றம் செய்ய முடியும்.

11.7 வலையகச் சேவைகள்

கீழ்க்காணும் வலையகச் சேவைகளை அணுகிப் பயன்பெறவும் கணிப்பொறிகள் உதவுகின்றன:

- வேளாண்மைச் சந்தைப்படுத்தல் பிணையம்
- வேலைவாய்ப்பு ஆலோசனை
- இணையம் வழி வேலைவாய்ப்பு
- பொதுச் சேமநிதி
- பல்வேறு தேர்வுகளின் முடிவுகள்

உலகின் ஏதோ ஒரு மூலையிலிருந்தும் பயன்படுத்திக் கொள்ளக்கூடிய இன்னும் பல ஐடீஇஎஸ் சேவைகளை வருங்காலத்தில் மிக விரைவிலேயே காண இருக்கிறோம்.

பயிற்சிகள்

இந்தப்பாடத்தில் விளக்கப்பட்ட பயன்பாட்டு மென்பொருள்கள் பற்றி மேலும் புரிந்துகொள்ள உதவும்,பல்லூடக விளக்கப் படங்கள் தனியாக ஒரு குறுவட்டில் உங்கள் பள்ளிக்குத் தரப்பட்டுள்ளது. அதன் உள்ளடக்கத்தை நீங்கள் கட்டாயம் பார்க்கவேண்டும். இக்குறுவட்டைப் பெற உங்கள்ஆசிரியரை அணுகுக. பாடத்தில் குறிப்பிடப்பட்டுள்ள இணைய தளங்களையும் பார்வை யிடுங்கள்.

பாடம் 12

கணிப்பொறி நன்னெறி
(Computer Ethics)

இரண்டாம் உலகப்போரின்போது ***நார்பெர்ட் வெய்னர்*** (Norbert Weiener) எழுதிய நூலில் முதன்முதலாகக் கணிப்பொறி நன்னெறிக் கோட்பாடுகள் பற்றிக் குறிப்பிட்டுள்ளார். வெய்னரின் நூலில் குறிப்பிடப்பட்டுள்ளவை:

(1) மனித வாழ்வின் நோக்கம் பற்றிய ஒரு விளக்கம்

(2) நீதிநெறியின் நான்கு கோட்பாடுகள்

(3) நன்னெறிகளை நடைமுறைப் படுத்துவதற்கான ஒரு சக்தி வாய்ந்த வழிமுறை

(4) கணிப்பொறி நன்னெறி தொடர்பான அடிப்படைக் கேள்விகள் பற்றிய கருத்துரைகள்.

(5) முக்கியமான கணிப்பொறி நன்னெறிக் கோட்பாடுகளுக்குரிய எடுத்துக்காட்டுகள்

1960-களின் இடைப்பகுதியில் கலிஃபோர்னியாவிலுள்ள மென்லோ பார்க்கில் உள்ள ***எஸ்ஆர்ஐ இன்டர்நேஷனலைச்*** சேர்ந்த ***டான்பார்க்கர்*** (Donn Parker), கணிப்பொறிப் பணியாளர்கள் கணிப்பொறியைத் தீங்கு எண்ணத்துடன் சட்டப்புறம்பாகப் பயன்படுத்தியதை ஆய்வுசெய்யத் தொடங்கினார்.1980-களில் அமெரிக்காவிலும் ஐரோப்பாவிலும் தகவல் தொழில்நுட்பத்தின் எண்ணற்ற சமூக, நன்னெறி தொடர்பான தீங்குகள் மக்கள் பிரச்சினையாக மாறின. கணிப்பொறி மூலமான குற்றங்கள், கணிப்பொறிக் கோளாறுகளினால் ஏற்பட்ட பேரிழப்புகள், கணிப்பொறித் தரவுத்தளங்கள் வழியாக ஒருவரின் தனிமறையில் (Privacy) குறுக்கீடுகள், மென்பொருள் சொத்துரிமை தொடர்பான சட்ட வழக்குகள் அவற்றுள் சில.

1990-களில் அனேக பல்கலைக் கழகங்கள், கணிப்பொறி நன்னெறி பற்றிய முறைப்படியான படிப்புகளை அறிமுகப்படுத்தின. பற்பல பாடப் புத்தகங்கள் மற்றும் பிற படிப்பு விளக்கவுரைகள் உருவாக்கப்பட்டன. இவை, ஆராய்ச்சிக்கான புதிய களங்கள் உருவாகக் காரணமாயின. ஆய்விதழ் வெளிவருவதற்கு அடித்தளமிட்டன.

பொதுவாகச் சொல்வதெனில் ***நன்னெறி*** என்பது, ஒழுக்கத்தின் தரப்பாடுகளைத் தீர்மானிக்கிற விதிமுறைகள் அல்லது சமூகம் ஏற்றுக்கொள்ளக் கூடிய நடத்தைகளைக் குறிப்பதாகக் கொள்ளலாம். இன்றைக்குப் பல கணிப்பொறிப் பயனாளர்கள் தகவல் தொழில்நுட்பம் தொடர்பான நடவடிக்கைகளைப் பொருத்தமட்டில் நன்னெறி என்பது எது, எது நன்னெறி

இல்லை என்கிற கேள்விகளை எழுப்புகின்றனர். நன்னெறி தொடர்பான சில பொதுவான வழிகாட்டுதல்கள் பயனுள்ளவையாக அமையும் என்பதில் ஐயமில்லை. தகவல் தொழில்நுட்பத்தில் அவற்றைப் பயன்படுத்த ஏதுவாக இருக்கும்.

கணிப்பொறி நன்னெறி பற்றிய பொதுவான வழிகாட்டுதல்கள் இவற்றுக் கெல்லாம் தேவைப்படுகின்றன:

- சொந்தத் தரவுகளைப் பாதுகாத்தல்
- கணிப்பொறிக் குற்றம்
- பாதுகாப்பு அரண்களை உடைத்தல்

12.1 தரவுப் பாதுகாப்பு (Data Security)

கீழ்க்காணும் வழிமுறைகள் சிலவற்றைப் பயன்படுத்தி ஒருவரின் சொந்தத் தரவுகளைப் பாதுகாத்துக் கொள்ளமுடியும்:

நேரடிப் பாதுகாப்பு (Physical Security)

நேரடிப் பாதுகாப்பு என்பது, கணிப்பொறி மென்பொருள்களைக் குறிப்பாக தரவுகள் சேமிக்கப்பட்டுள்ள மின்காந்த வட்டுகளையும் மற்றும் சட்டப்புறம்பாக அணுகவும் களவாடவும் பழுதாக்கவும் அழிக்கவும் ஏதுவான பிற பொருள் களையும் பாதுகாத்துக் கொள்வதைக் குறிக்கிறது. இத்தகு கணிப்பொறி வளங்களை அணுகுவோர்க்குக் கட்டுப்பாடு விதிப்பதன் மூலம் இப்பாதுகாப்பை வழங்கிட முடியும்.

சொந்தப் பாதுகாப்பு (Personal Security)

சொந்தப் பாதுகாப்பு என்பது, அனுமதி பெற்றவர்கள் மட்டுமே கணிப்பொறி முறைமைக்குள் நுழைய முடியுமாறு மென்பொருளைத் தகவமைப்பதைக் குறிக்கிறது. பயனர் பெயர் (User ID) மற்றும் கடவுச் சொல்(Password) ஆகியவை இத்தகைய பாதுகாப்புக்கான கருவிகளாகும். கணிப்பொறியில் தகவலறியத் தேவை உள்ளவர்கள் மட்டுமே பயனர் பெயரும் கடவுச்சொல்லும் பெற்றிருக்க வேண்டும்.

பணியாளர் பாதுகாப்பு (Personal Security)

பணியாளர் பாதுகாப்பு என்பது, கணிப்பொறி முறைமையையும், தரவு களையும் நேர்மையற்ற அல்லது கவனக்குறைவான பணியாளர்களிடமிருந்து பாதுகாப்பதைக் குறிக்கிறது.

12.2 கணிப்பொறிக் குற்றம் (Computer Crime)

கணிப்பொறிக் குற்றம் என்பது, கணிப்பொறி மென்பொருள், தரவுகள், இவற்றை அணுகுதல் - இவை குற்றத்தின் காரணமாய், நோக்கமாய் அல்லது கருவியாய் அமையக் கூடிய எந்தவொரு சட்டப் புறம்பான நடவடிக்கையையும் குறிக்கிறது.

கணிப்பொறிக் குற்றங்கள் சில:

- இணையத்தின் வழியாகப் பணப் பரிமாற்றம் செய்வது தொடர்பான குற்றங்கள்.
- கணிப்பொறியைப் பயன்படுத்திச் சட்டப்புறம்பாக வெளிநாட்டுத் தொலைபேசி அழைப்புகளைச் செய்தல்
- ரகசியமான கோப்புகளை அத்துமீறி அணுகுதல்
- வன்பொருள்களைக் களவாடல்
- சொந்தத் தரவுகளை முறைகேடாகப் பயன்படுத்தல்/விற்றல்
- வன்பொருள் அல்லது மென்பொருள்களின் உரிமையிலா நகலாக்கம்
- நச்சுநிரல் (Virus)
- பாதுகாப்பு அரண் உடைத்தல் (Cracking)
- கணிப்பொறி நேரத்தைக் களவாடல்

ஒட்டுமொத்தக் கணிப்பொறிக் குற்றங்களில் 80% குற்றங்கள் அந்த நிறுவனத்துக்குள்ளேயே நிகழ்பவை என்பதை அறிக. 60%-க்கு மேற்பட்ட குற்றங்கள் அறிவிக்கப்படாமலேயே போய்விடுகின்றன.

போலியான வன்பொருள், மென்பொருள்களை உருவாக்குவதும் பயன்படுத்துவதும் ***உரிமையிலா நகலாக்கம்*** (Piracy) எனப்படுகிறது.

உரிமையிலா நகலாக்கம் செய்ய நாம் முனைவதற்குக் காரணம்:

- நாம் இலவசப் பொருள்களை விரும்புகிறோம்.
- நம்மால் இலவசமாகப் பெறமுடியும் என்கிறபோது, எதற்காக விலைகொடுத்து வாங்கவேண்டும் என்கிற எண்ணம்.
- நமது எண்ணமும் செயல்பாடுகளும் சுயநலத்தை அடிப்படையாய்க் கொண்டவை
- குறைந்த ஆபத்து விளைகிற, பண ரீதியாகப் பயன்தருகிற ஒன்றைப் பெறும் வாய்ப்புக் கிடைக்கும் எனில் அதைத் தயங்காமல் செய்து முடிப்போம். நாம் வாழும் முதலாளித்துவ சமூகம் அவ்வாறு நம்மைப் பழக்கியிருக்கிறது.

நச்சுநிரல் தன்னைத்தானே நகலெடுத்துப் பெருக்கிக்கொள்ளும். உங்கள் கணிப்பொறியில் சேமித்து வைத்துள்ள தரவுகளுக்கும் கோப்புகளுக்கும் பாதிப்பு ஏற்படுத்தக் கூடியது. சவாலைச் சாதித்துக் காட்ட வேண்டும், பிறர் உடைமைகளுக்கு அழிவை ஏற்படுத்த வேண்டும் என்கிற தூண்டுதல் உள்ள, திறன்மிக்க மாபெரும் நிரலாக்கத் திறன்கொண்ட நிரலர்கள் எழுதுகின்ற நிரல்களே இவை. அறியப்பட்ட நச்சுநிரல்கள் 57000-க்கும் அதிகமானவை. ஒவ்வொரு நாளும் 6 புதிய நச்சுநிரல்கள் கண்டறியப்படுகின்றன.

நிறுவனங்களில் பயன்படுத்தப்படும் கணிப்பொறிகளில்பெரும்பாலானவை எல்லா நேரமும் பயன்பாட்டில் இருப்பதில்லை. பல மணி நேரம் அவை வாளா இருக்கின்றன. இப்படி இருக்கும் நேரங்களில் கணிப்பொறிகளைப் பலனுள்ள முறையில் பயன்படுத்தப் பல்வேறு வழிமுறைகள் ஆய்வு செய்யப்பட்டு வருகின்றன. எனினும், இவ்வாறு ஒரு நிறுவனத்தில் பயன் படுத்தப்படாமல் இருக்கின்ற கணிப்பொறி நேரங்கள் சட்டப்புறம்பான வழிகளில் களவாடப்படுகின்றன. நிறுவனத்துக்குத் தெரியாமலேயே வாளா இருக்கும் கணிப்பொறிகளில் வேறுபிற மென்பொருள்கள் இயங்கிக் கொண்டிருக்கின்றன. இதனையே 'கணிப்பொறி நேரம் களவாடப்படல்' என்கிறோம்.

கணிப்பொறி நன்னெறி தொடர்பாகப் பரவலாக எடுத்துக்காட்டப்படும் குறிப்பு, ***கணிப்பொறி நன்னெறிக் கழகம்*** வரையறுத்துள்ள ***கணிப்பொறி நன்னெறிக்கான*** பத்துக் கட்டளைகள் ஆகும். அவை கீழே தரப்பட்டுள்ளன:

- பிறருக்குத் தீங்கு விளைவிக்குமாறு ஒரு கணிப்பொறியைப் பயன்படுத்தாதீர்கள்.
- பிறரின் கணிப்பொறிப் பணியில் தலையிடாதீர்கள்.
- பிறரின் கணிப்பொறிக் கோப்புகளை நோட்டம் விடாதீர்கள்.
- கணிப்பொறியைப் பயன்படுத்திக் களவாடாதீர்கள்.
- பொய் சாட்சி உருவாக்க கணிப்பொறியைப் பயன்படுத்தாதீர்கள்.
- நீங்கள் விலை கொடுத்து வாங்காத பதிப்புரிமை உள்ள மென்பொருள்களை நகலெடுக்காதீர்கள் அல்லது பயன்படுத்தாதீர்கள்.
- பிறருடைய கணிப்பொறி வளங்களை அனுமதியின்றியோ, உரிய கட்டணமின்றியோ பயன்படுத்தாதீர்கள்.
- பிறரின் அறிவுசார் படைப்புகளை முறைகேடாக அபகரித்துக் கொள்ளாதீர்கள்.
- நீங்கள் உருவாக்கும் மென்பொருள் அல்லது நீங்கள் வடிவமைக்கும் முறைமையின் சமூக விளைவுகளைச் சற்றே எண்ணிப் பாருங்கள்.
- உடன்வாழும் மனிதர்களின் உணர்வுகளைக் கணக்கில் கொண்டு உரிய மரியாதை வழங்கும் வகையிலேயே எப்போதும் கணிப்பொறியைப் பயன்படுத்துங்கள்.

கணிப்பொறிக் குற்றங்களுக்கென அரசாங்கம் சிறப்பான சட்டங்களை நிறைவேற்ற வேண்டும். இன்றைக்குப் பல நாடுகள் பல்வேறு வகையான சட்டங்களை நிறைவேற்றி, கணிப்பொறிக் குற்றம் புரிவோர்க்குத் தண்டனை களை வழங்கி வருகின்றன. இந்தியாவும் கணிப்பொறிக் குற்றங்களைத் தடுக்க மின்வெளிச் சட்டங்களை (Cyber Laws) இயற்றியுள்ளது.

12.3 அரண் உடைத்தல் (Cracking)

சட்டப்புறம்பான முறையில் ஒரு கணிப்பொறி முறைமையில் அல்லது பிணையத்தில் நுழைவதையே ***அரண் உடைத்தல்*** என்கிறோம். கணிப்பொறி யில் இருக்கும் சிறப்பான வளங்களைச் சட்டப்புறம்பாகப் பயன்படுத்துவதே அரண் உடைத்தலின் அடிப்படை நோக்கம் ஆகும். வளங்கள் என்பவை வன்பொருள், மென்பொருள், கோப்புகள், முறைமைத் தகவல்கள்- இவற்றுள் எதுவாகவும் இருக்கலாம். பழிவாங்கல், வணிகப் போட்டி, சாகசம் ஆகியவை இத்தகைய குற்றத்துக்கான பிற காரணங்கள் ஆகும்.

12.4 வேலை, குடும்பம், பொழுதுபோக்கு

கையகக் கணிப்பொறிகளும் தொலைக் கணிப்பணியும் (Tele Computing), மனிதர்கள் தமது பணிகளை எந்த இடத்துக்கும் எந்த நேரத்திலும் எடுத்துச் சென்று செய்து முடிக்கும் நிலையை உருவாக்கியுள்ளன. இதன் விளைவாக, அவர்களது அலுவலகப் பணிகள் குடும்பத்தினருடன் இருக்கும் நேரத்தையும், விடுமுறைகளையும், ஓய்வு நேரத்தையும் விழுங்கி விடுகின்றன. குடும்பம், நண்பர்கள் போன்ற மரபுவழிப்பட்ட அமைப்புகளையே பலவீனப் படுத்துகின்றன. தனிவாழ்க்கை, பொதுவாழ்க்கை என்ற இரண்டுக்கும் இடையே யான கோடு அழிந்து வருகிறது. கணிப்பொறி நன்னெறியில் இது மிகவும் முக்கியமான பிரச்சினையாக உருவெடுத்து வருகிறது.

பயிற்சி வினாக்கள்:

1. கணிப்பொறி அமைப்பில் நுழைவதற்குக் கடவுச்சொல் (Password) இருக்க வேண்டிய தேவை என்ன?
2. கணிப்பொறி இயக்க முறைமை (Operating System) பாதுகாப்பினை எவ்வாறு மேம்படுத்துகிறது?