பாடம் 4
சிக்கல் தீர்க்கும் நுட்பங்களும் சி-மொழி நிரலாக்கமும்

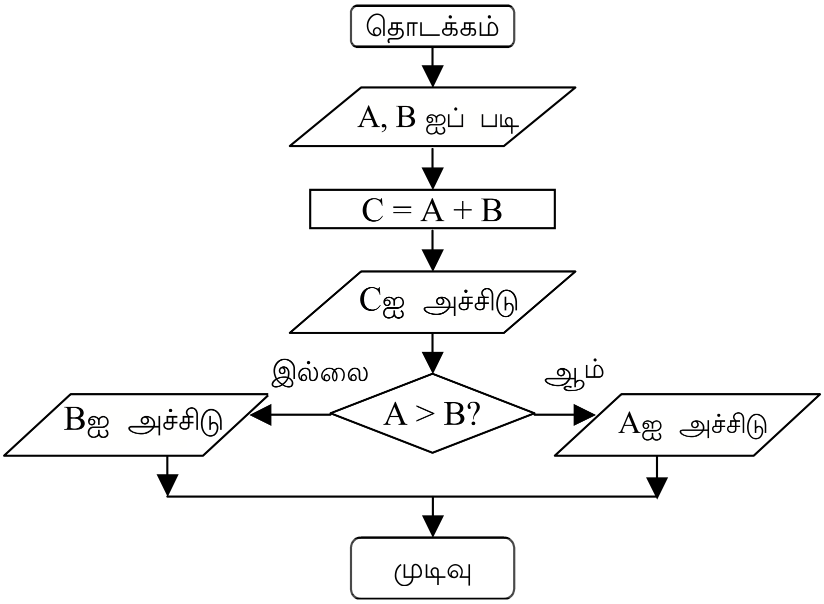
4.1 சிக்கல் தீர்க்கும் நுட்பங்கள் (Problem Solving Techniques)

நாம் செய்யும் கணிப்பணி, நாம் பயன்படுத்தும் கணிப்பொறி மொழியைச் சாராதது ஆகும். கணிப்பொறியையும் சாராதது ஆகும். கணிப்பொறியையும் மொழியையும் தேர்ந்தெடுப்பது பெரும்பாலும், குறிப்பிட்ட நிலைமையில் அவற்றின் பொருத்தமுடைமையைச் சார்ந்ததாகும். ஒரு நிரலை உருவாக்கும்போது, தமிழ் அல்லது ஆங்கிலம் போன்ற இயற்கை மொழியில்தான் சிந்திக்கிறோம். கடைசி நேரத்தில்தான், வடிவாக்கத்தை (design) உயர்நிலை மொழியில் ஒரு நிரலாக எழுதுகிறோம்.

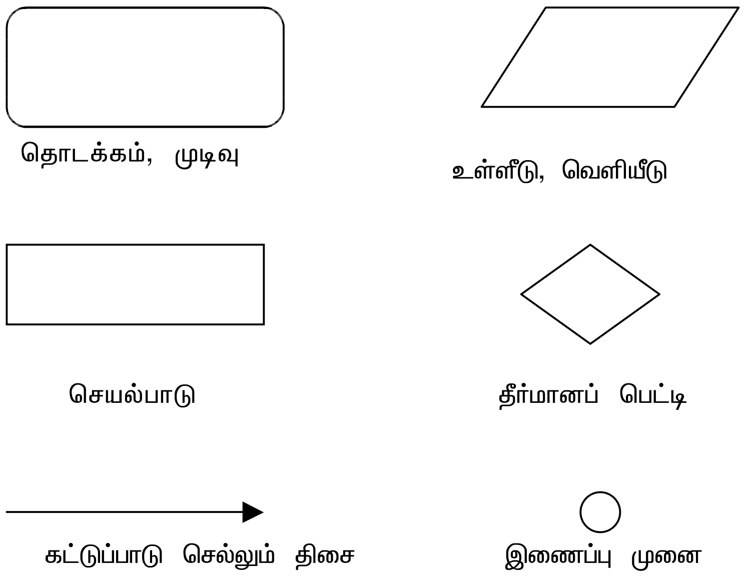
கணிப்பொறி மொழிகளில், ஒவ்வொரு கூற்றும், காற்புள்ளி, அரைப்புள்ளிகள் உட்படத் துல்லியமாக எழுதப்பட வேண்டும். கட்டளை வரிகளை எழுதும்போது ஒருவர் மிகுந்த கவனத்துடன் இருக்கவேண்டும். இயற்கை மொழிகளில், சொல்தொடர்கள் (sentences) மிகவும் நீளமாக இருக்கலாம். சில வேளைகளில் அவை தெளிவற்றதாய் இருக்கலாம். அனைத்து இயற்கை மொழிகளுக்குமே இவ்வியல்பு உண்டு. எனவே சிக்கல்களை எவ்விதக் குழப்பமுமின்றித் தெளிவாகப் புரிந்துகொள்ள, ஓர் இடைநிலை மொழியில் (Intermediary Language) எழுதுகிறோம். இதை எழுதுவதும் புரிந்து கொள்வதும் எளிது. எவ்விதக் குழப்பமும் எழாது. இத்தகைய இடைநிலை மொழி இயற்கை மொழிகளுக்கும் கணிப்பொறி மொழிகளுக்கும் இடைப்பட்ட மொழியாகும். அத்தகைய இடைநிலை மொழிகள் இரண்டினை நாம் படிக்க இருக்கிறோம். அவை, மிகப் பெருமளவில் பயன்படுத்தப்படுகின்ற பாய்வுப்படம் (Flow Chart) மற்றும் போலிக் குறிமுறை (Pseudo Code) ஆகியவை ஆகும்.

முதலில் நாம் பாய்வுப்படத்தை எடுத்துக் கொள்வோம். கணிப்பு நிலைப்பாதைகளின் பாய்வுகள் படமாக உருவகிக்கப்படுவதால் இது பாய்வுப்படம் எனப் பெயர்பெற்றது. ஓர் எடுத்துக்காட்டுடன் தொடங்குவோம். இரண்டு எண்களின் கூட்டுத் தொகையையும், இரண்டில் பெரிய எண்ணையும் கண்டுபிடிக்க வேண்டும். இப்பணியை நிறைவேற்றி முடிக்க, முதலில் இரண்டு எண்களை உள்ளீடாகப் பெற்று அவற்றை தனித்தனி

இடங்களில் தனித்தனிப் பெயர்களில் இருத்திவைக்க வேண்டும். பிறகு அவற்றின் கூட்டுத் தொகையைக் கணித்துத் திரையில் காட்ட வேண்டும். அடுத்து இரண்டு எண்களில் பெரிய எண் கண்டறியப்பட்டுத் திரையிடப்பட வேண்டும். இதற்கான பாய்வுப்படம் படம் 4.1-ல் தரப்பட்டுள்ளது. பாய்வுப் படத்தில் ஒவ்வொரு வடிவமும் ஒரு குறிப்பிட்ட பொருளைக் குறிக்கின்றன. அவை படம் 4.2-ல் தரப்பட்டுள்ளன.



பாய்வுப்படம் 4.1



பாய்வுப்படம் 4.2

மேலே குறிப்பிட்ட பாய்வுப்படத்தில்,

C = A + B

என்று எழுதப்பட்டுள்ளதற்குச் சிறப்பான பொருள் உண்டு. நாம் வழக்க மான நிகர்ப்பாட்டுக் (equal to) குறியைப் (=) பயன்படுத்தியுள்ள போதிலும், ஒரு நிகர்ப்பாட்டில் (equation) உள்ள பொருளில் அது பயன்படுத்தப் படவில்லை. உண்மையில் இந்தக் கூற்றின் (statement) பொருள், ''வலப் பக்கத்தில் இடம்பெற்றுள்ள A, B ஆகியவற்றின் இப்போதைய மதிப்பு களைக் கூட்டி, கூட்டுத் தொகையை இடப்பக்கமுள்ள C- ன் புதிய மதிப் பாக, இருத்திவை'' என்பதாகும். எடுத்துக்காட்டாக, இந்த விளக்கத்தின் படி,

A = A + 1

என்பதும் ஏற்கத் தகுந்த கூற்றுதான். A- ன் மதிப்பில் ஒன்றைக் கூட்டி, அப்புதிய மதிப்பு மீண்டும் A- யிலேயே இருத்திவைக்கப்படுகிறது. அதா வது, A- ன் மதிப்பு ஒன்று கூட்டப்படுகிறது.

A = A + B

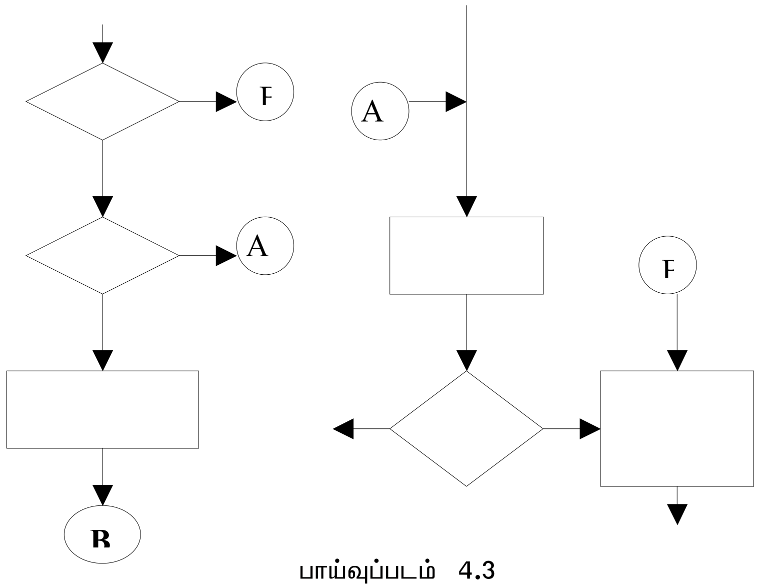
என்றும் எழுதலாம் என்பதை நினைவில் கொள்க. ஆனால்,

A + B = A

என்று எழுதமுடியாது. இடப்பக்கத்தில், வலப்பக்கத்தின் விடையை இருத்தி வைக்க ஒரேயொரு பெயர் மட்டுமே இருக்க வேண்டும்.

கணிப்பொறி மூலம் செய்ய முடிகிற அனைத்துக் கணிப்புகளையும் பாய்வுப்படமாக வரைந்து காட்ட முடியும். சிக்கல் சிறிதாக இருப்பின் பாய்வுப்படமும் சிறிதாக இருக்கும். பெரிய சிக்கல்களுக்கான பாய்வுப் படம் எப்படி இருக்கும் என எண்ணிப் பாருங்கள். நடைமுறை வாழ்க் கையின் சிக்கல்கள் எப்போதும் மிகப் பெரியவை. ஆனால் வகுப்பறைக் கணக்குகள் சிறியவையே. காரணம், வரம்புக்குட்பட்ட நேரத்துக்குள் சில குறிப்பிட்ட கருத்துருக்களை (Concepts) கற்றுத் தருவதற்காக அவை உருவாக்கப்படுகின்றன.

பெரிய சிக்கல்களுக்கு பாய்வுப் படம் பெரிதாக இருக்கும். ஆனால் நாம் பயன்படுத்தும் தாளின் அளவு சிறியது. எனவே, ஒரு பாய்வுப் படம் வரையப் பல தாள்கள் தேவைப்படலாம். ஆனால் ஒரு தாளிலி ருந்து அடுத்த தாளுக்குச் செல்வது எப்படி? 'இணைப்பிகள்' (Connec-tors) என்று அழைக்கப்படும் சிறிய வட்டங்கள் மூலம் இச்சிக்கலைத் தீர்க்கலாம். இந்த வட்டத்தில் ஓர் எழுத்தைக் குறிப்பிட வேண்டும். ஒரே எழுத்தைக் கொண்ட அனைத்து வட்டங்களும் அவை எங்கிருந்தாலும் ஒரே புள்ளியைக் குறிக்கின்றன. அவை ஒரே பக்கத்தில் இருக்கலாம் அல்லது வெவ்வேறான பக்கங்களில் இடம் பெற்றிருக்கலாம். பாய்வுப் படம் 4.3-ல் ஓர் எடுத்துக்காட்டு காண்க:



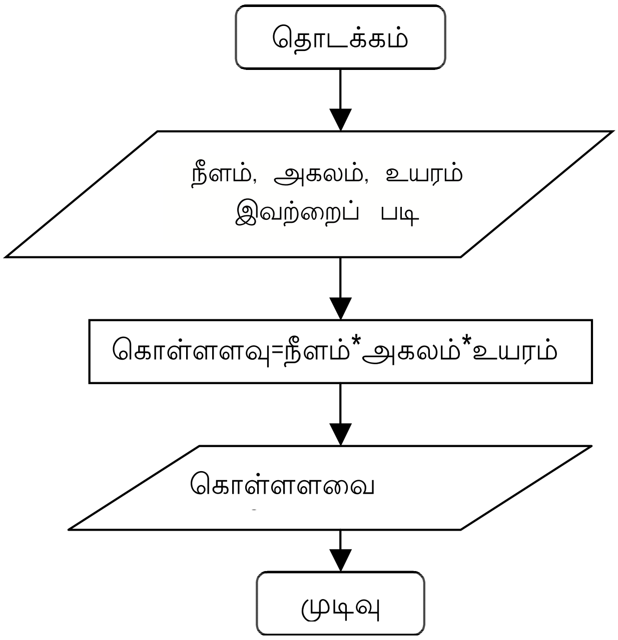
பாய்வுப்படம் 4.3

பாய்வுப்படங்களின் நன்மைகள்

பாய்வுப்படங்கள் துல்லியமானவை. நமது எண்ணங்களை மிகச் சரியாக உருவகப்படுத்துபவை. சிறிய பாய்வுப்படங்களை எளிதாகப் புரிந்து கொள்ள முடியும். நடைமுறை வாழ்வின் சிக்கல்களுக்கான பாய்வுப் படங்கள் பல பக்கங்கங்களை எடுத்துக் கொள்ளும், எனவே அவற்றைப் புரிந்துகொள்வது மிகவும் கடினம் என்பது ஒரு குறைபாடு ஆகும். எனவே, அத்தகைய நிலைமைகளில் எவரும் பாய்வுப்படங்களைப் பயன் படுத்த மாட்டார்கள்.

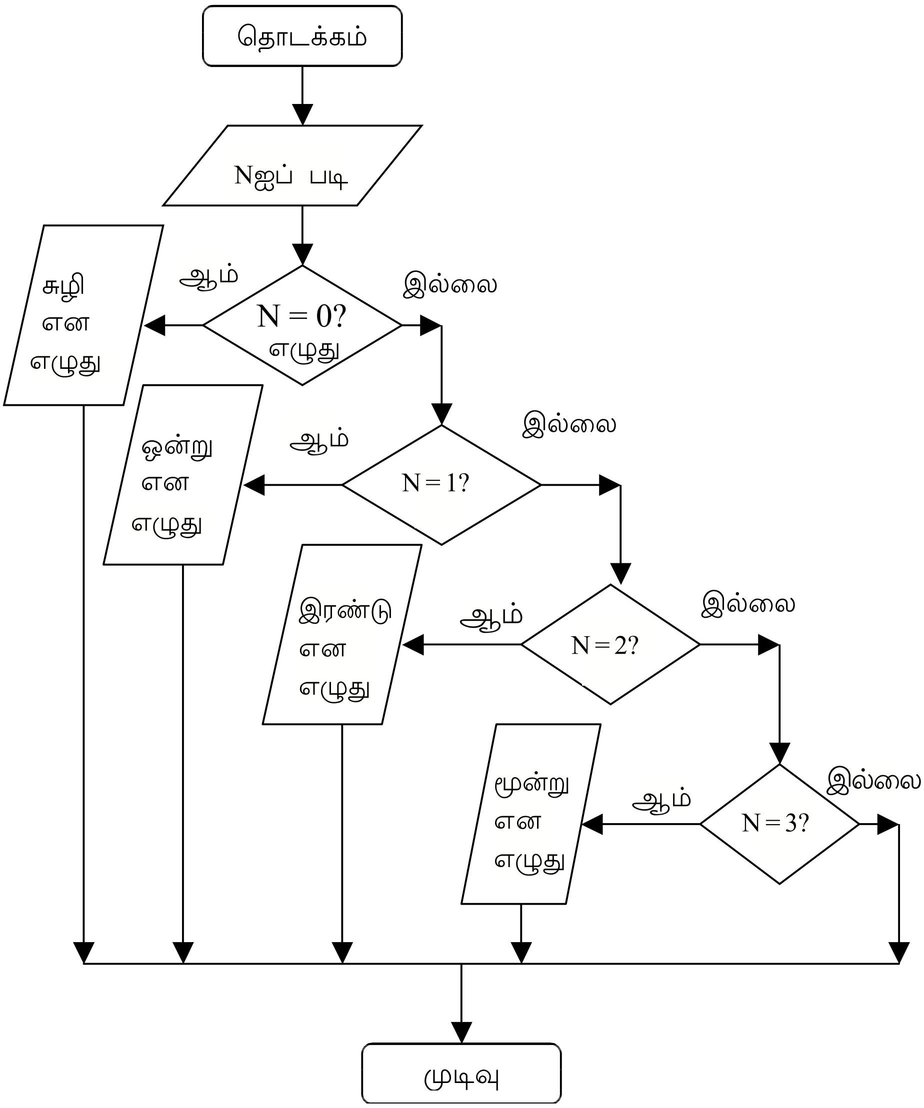
கீழே காணும் கணக்குகளுக்கான சிறிய பாய்வுப்படங்களை எடுத் துக் கொள்ளுங்கள். அவை, கணக்குகளுக்கான தீர்வை வழங்குகின்ற னவா எனப் பாருங்கள். அவற்றை மனப்பாடம் செய்யவேண்டாம். அவற் றைப் புரிந்துகொள்ள முயலுங்கள். ஒரு நிரலை எழுதுவதற்கு முன்பாக நாம் எவ்வளவு சிந்திக்க வேண்டியுள்ளது என்பதைக் கவனத்தில் கொள் ளுங்கள்.

ஒரு பெட்டியின் நீளம், அகலம், உயரம் ஆகியவற்றைக் கொண்டு அதன் கொள்ளளவு கணக்கிடப்படுவதை பாய்வுப்படம் 4.4-ல் காண்க.



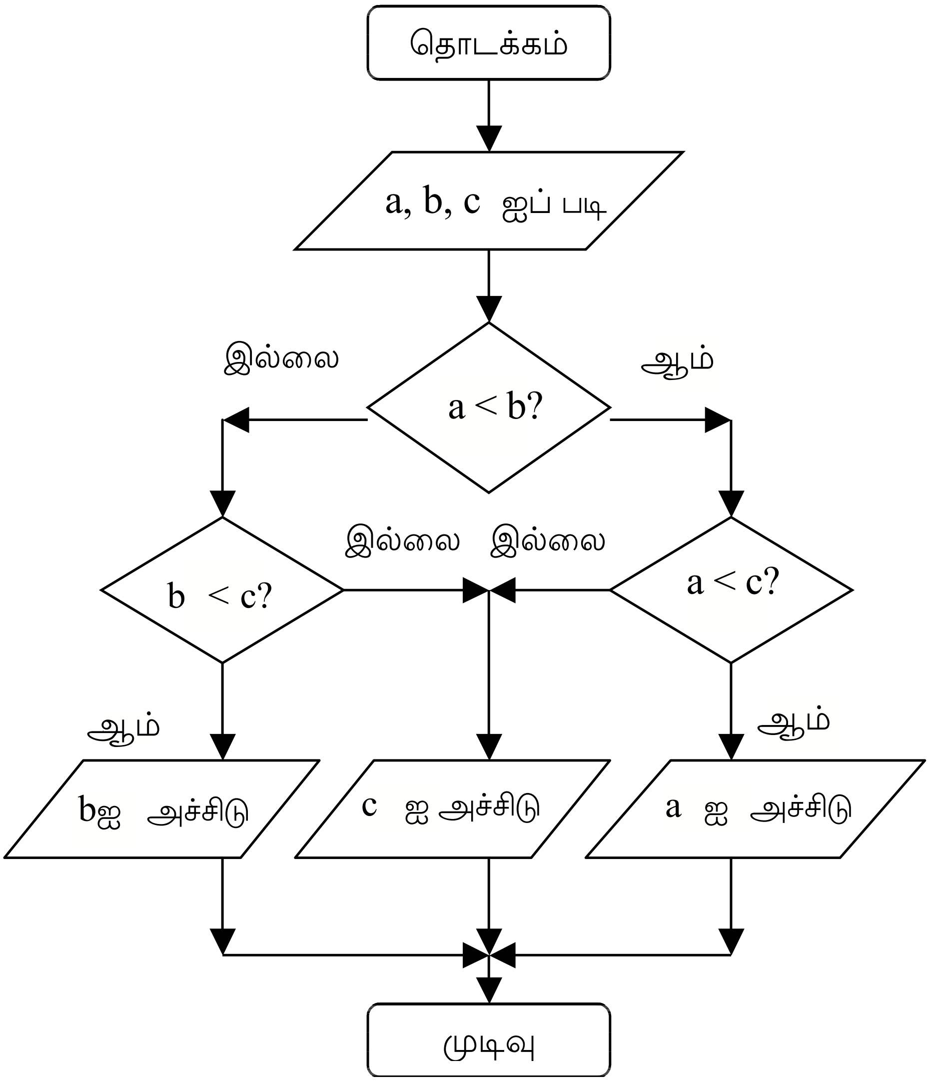
பாய்வுப்படம் 4.4

பாய்வுப்படம் 4.5, 0லிருந்து 3 வரையிலான எண்களைப் படித்து, அவற்றை எழுத்தில் எழுதிக் காட்டுகிறது.



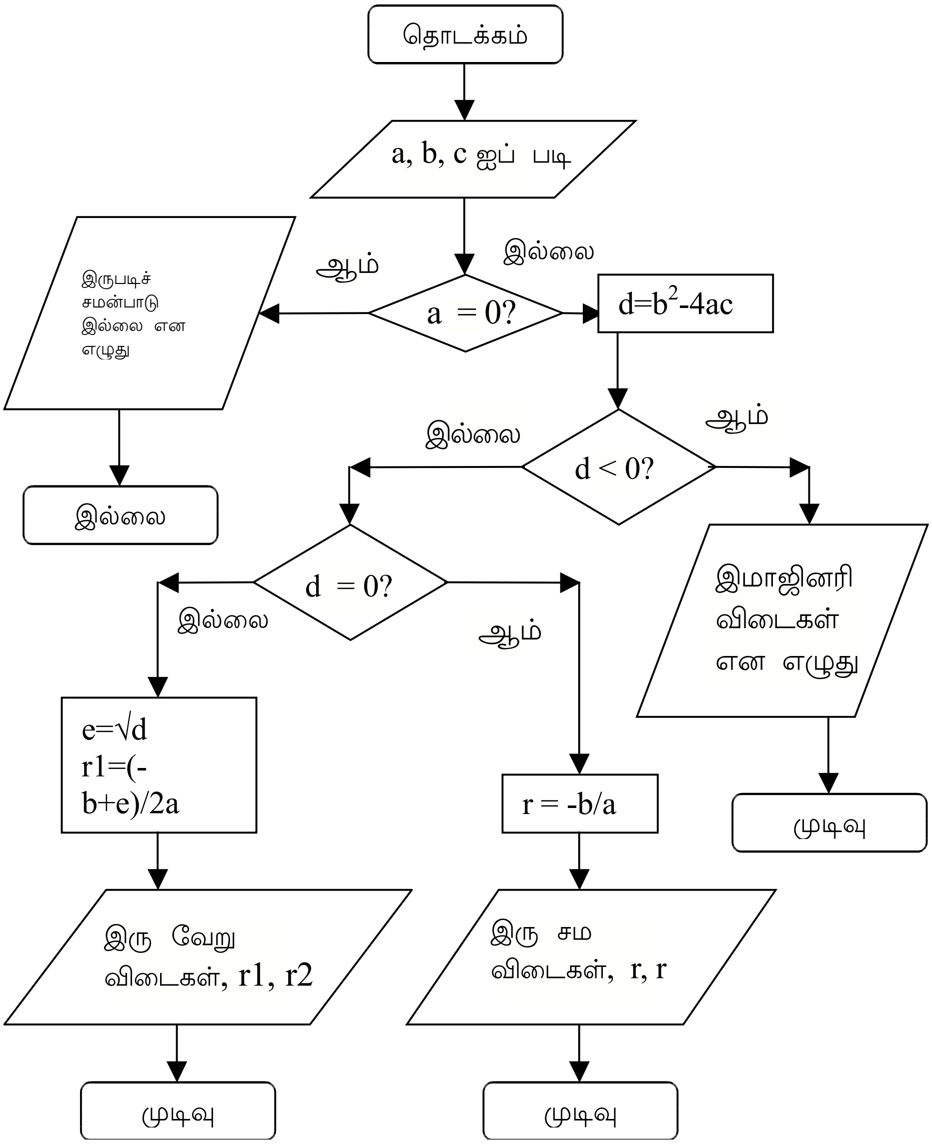
பாய்வுப்படம் 4.5

பாய்வுப்படம் 4.6, கொடுக்கப்பட்ட மூன்று எண்களில் சிறிய எண்ணைக் கண்டறிந்து சொல்கிறது.

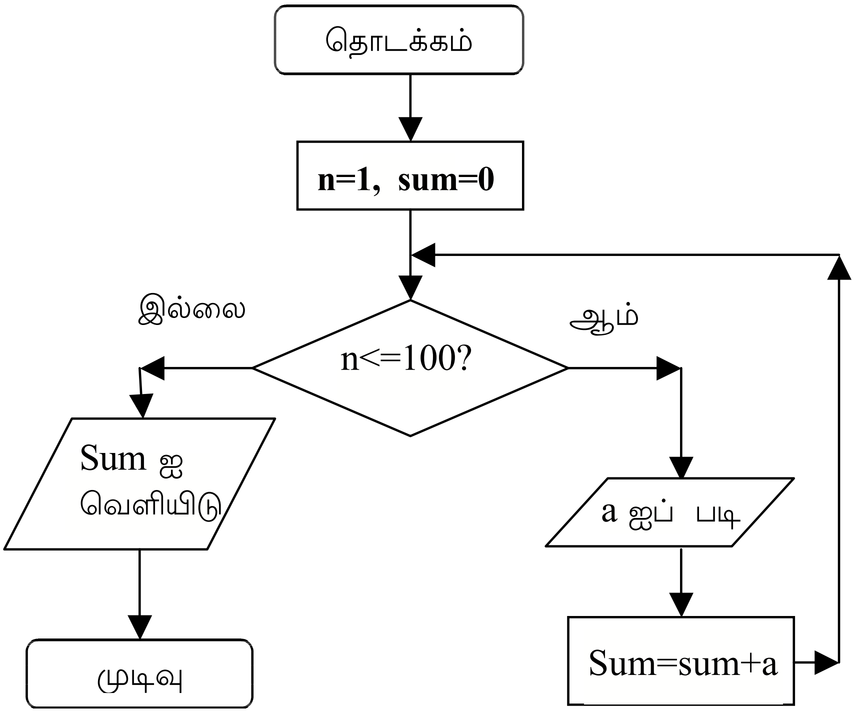


பாய்வுப்படம் 4.6

பாய்வுப்படம் 4.7, இருபடி நிகர்ப்பாட்டை (Quadratic equation) தீர்ப்ப தற்கான ஒரு வழிமுறையை வழங்குகிறது.

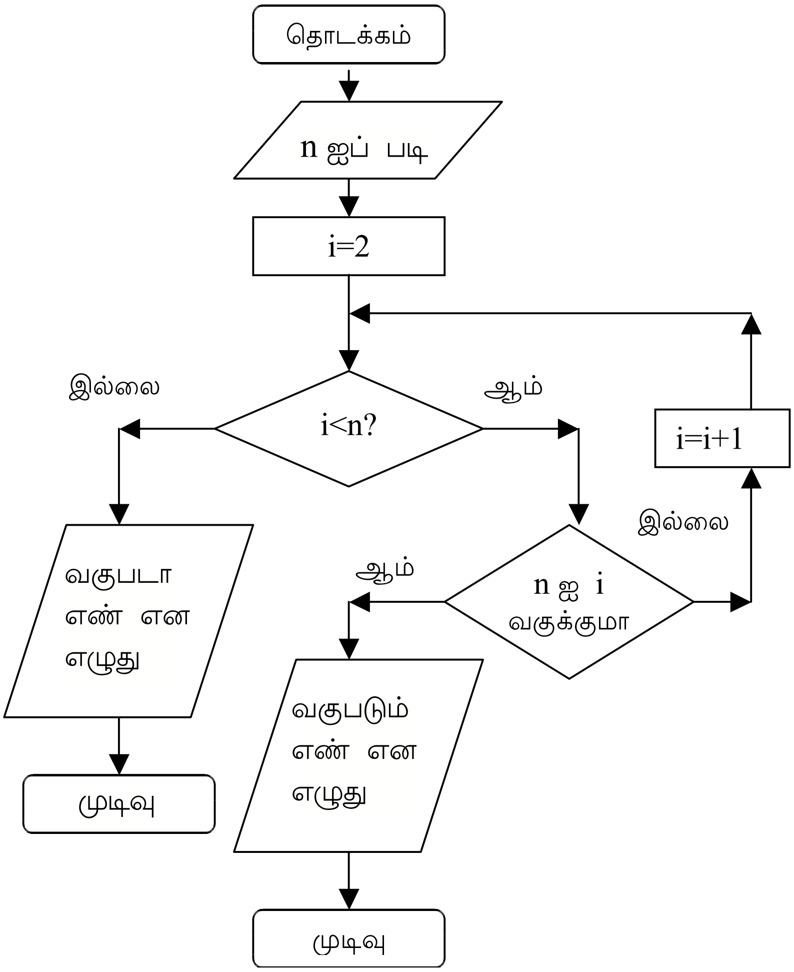


பாய்வுப் படம் 4.7



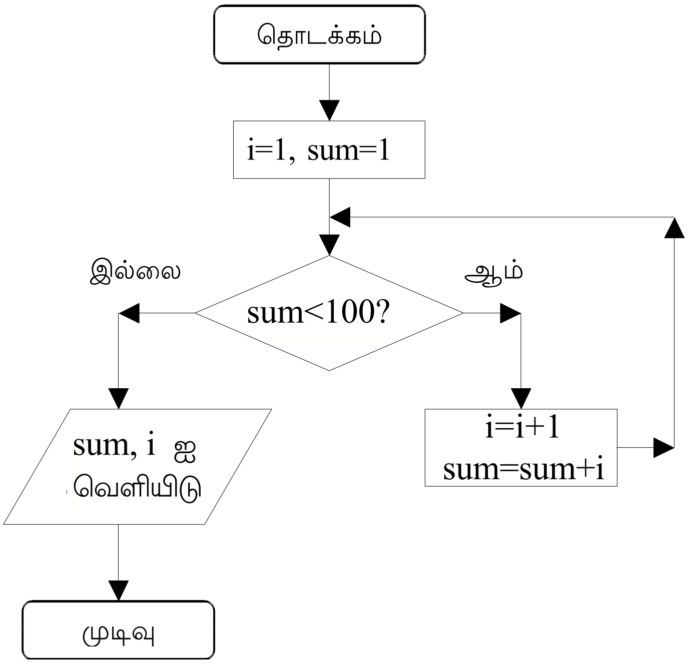
பாய்வுப்படம் 4.8

பாய்வுப்படம் 4.9, கொடுக்கப்பட்ட ஒரு முழுஎண் **(Integer),** பகு எண்ணா **(prime number)** அல்லது பகா எண்ணா என்பதைத் தீர்மானிக் கிறது.



பாய்வுப்படம் 4.9

பாய்வுப்படம் 4.10, 1+2+3+............+n என்பதன் கூட்டுத் தொகை சரியாக 1000 அல்லது அதற்குச் சற்றே கூடுதலாக வருமாறு, மீச்சிறு முழு எண்ணைக் (smallest integer) கண்டறிந்து சொல்கிறது.



பாய்வுப்படம் 4.10

4.1.1 அடிப்படையான நிபந்தனை மற்றும் கட்டுப்பாட்டுக் கட்டமைப்புகள்(Fundamental Conditional and Control Structures)

கணிப்பொறிப் போலிக் குறிமுறையில் (Computer pseudo code) ஒவ்வொரு படிநிலை (step) யையும் நாம் கணிப்பொறிக்கு அறிவுறுத்த வேண்டும். என்ன செய்யப் போகிறோம் என்பதைப்பற்றி நமக்குத் தெளிவான புரிதல் இருந்தால் மட்டுமே நம்மால் கணிப்பொறிப் போலிக் குறிமுறையை எழுத முடியும். நாம் சொல்வதை நாம் சொல்கிறபடியே கணிப்பொறி செய்யும். அது தானாகவே எதையும் செய்யாது. கணிப்பொறியைப் பரமார்த்த குருவின் ஐந்தாவது சிஷ்யன் என்றுகூடச் சொல்லலாம். ஏன் என வியப்படைகிறீர்களா?

பரமார்த்த குருவின் சிஷ்யர்களைப் பற்றிய பல கதைகளுள் ஒரு கதையைப் பார்ப்போம். பரமார்த்த குருவுக்கு நான்கு சிஷ்யர்கள் உண்டு. மட்டி, மடையன், மூடன், முட்டாள் என்பது அவர்களின் பெயர்கள். இந்த நான்கு பெயர்களும் 'அறிவிலி' என்னும் ஒரே பொருளையே உணர்த்துகின்றன. இவர்கள் நால்வரும் தங்கள் குருவுக்காக ஒரு வய தான குதிரையை வாங்கினர். குருவைக் குதிரையில் உட்காரவைத்துக் கொண்டு நால்வரும் நடந்து வந்தனர். குருவின் தலைப்பாகை ஒரு மரக்கிளையில் மோதிக் கீழே விழுந்துவிட்டது. சிறிது நேரம் கழித்து குரு தன் தலைப்பாகை எங்கே என்று கேட்டார். கீழே விழுந்துவிட்ட தாக சிஷ்யர்கள் கூறினர். ஏன் அதை எடுக்கவில்லை என்று கேட்ட போது, 'அவ்வாறு நீங்கள் சொல்லவில்லையே' என்று பதிலிறுத்தனர். உடனே குரு, இனிமேல் எது கீழே விழுந்தாலும் அதை எடுத்து உடன் கொண்டு வரவேண்டும் என்று கட்டளையிட்டார்.

ஒரு சிஷ்யன் பின்னோக்கி நடந்து சென்று தலைப்பாகையை எடுத்து வந்தான். தலைப்பாகை நிறைய குதிரையின் சாணம் இருப்பதைப் பார்த்துக் குருவுக்குக் கோபம் வந்தது. தலைப்பாகையோடு சாணத்தையும் எதற்காக எடுத்து வந்தீர்கள் என்று குரு கேட்க, "நீங்கள்தானே, கீழே எது விழுந் தாலும் எடுத்துவரச் சொன்னீர்கள். எதை எடுப்பது, எதை விடுப்பது என்பது எங்களுக்கு எப்படித் தெரியும்? எனவே, எவற்றையெல்லாம் எடுக்க வேண்டும் என ஒரு பட்டியல் கொடுத்துவிடுங்கள். எங்களுக்கு எவ்விதக் குழப்பமும் இருக்காது" என்று பதில் கூறினர். குரு ஒரு நீண்ட பட்டியலைக் கொடுத்தார். அவர்களது பயணம் தொடர்ந்தது.

சிறிது நேரம் கழித்து, வயதான அந்தக் குதிரை தடுமாறிக் கீழே விழுந்தது. குருவும் அவருடைய பொருட்களும் கீழே விழுந்து சிதறின. உடனே, ஒரு சிஷ்யன் பட்டியலிலிருந்த பொருட்களை வாசிக்கத் தொடங் கினான். தலைப்பாகை, வேட்டி, துண்டு என ஒவ்வொன்றாக எடுத்துச் சேகரித்தனர். குரு வெறும் இடுப்புத் துணியோடு கிடந்தார். தன்னைத் தூக்கிக் குதிரையின்மேல் கிடத்தும்படி சிஷ்யர்களிடம் கேட்டுக் கொண் டார். "பட்டியலில் நீங்கள் இடம்பெறவில்லை குருவே" என்று உடனடி யாகப் பதில் வந்தது. அவரது வேண்டுகோளை அவர்கள் காது கொடுத்துக் கேட்கத் தயாராயில்லை. பட்டியலைத் திருத்தி அமைக்கும்படியும் பட் டியலில் தன்னுடைய பெயரையும் சேர்த்துக் கொள்ளும்படியும் குரு கேட்டுக் கொண்டார். அதன்பிறகு, பட்டியலை மீண்டும் சரிபார்க்கும்படி ஆணையிட்டார். இப்போது, சிஷ்யர்கள் குருவைத் தூக்கிவிட்டனர்.

கணிப்பொறிப் போலிக் குறிமுறையும் இந்தப் பட்டியல் போன்றதுதான்.

கணிப்பொறி, பரமார்த்த குருவின் சிஷ்யர்களில் ஒருவரைப் போன்றது. இரண்டுக்கும் அதிக வேறுபாடில்லை. கணிப்பொறிக்குத் தெளிவாக, விளக்கமாக ஆணைகளை வழங்க வேண்டும். அவ்வாறு விளக்கமான முறையில் சிந்திக்க நாம் பயிற்சிபெற வேண்டும்.

மொத்தத்தில் பார்த்தால், மூன்றே மூன்று நுட்பங்களே மீண்டும் மீண்டும் இடம்பெறுகின்றன. இந்த நுட்பங்களைப் பயன்படுத்தித்தான் ஒட்டுமொத்தக் கணிப்பணியும் நிறைவேற்றப்படுகிறது. இவற்றைக் கற்றுக்கொள்ள வேண்டும் என்பது, மீனவர் நீச்சல் கற்றுக்கொள்ள வேண்டியது எவ்வளவு கட்டாயமோ அவ்வளவு கட்டாயமாகும்.

வரிசைமுறைப்படுத்தல் (Sequencing)

பொதுவாகக் கணக்கீடுகள் ஒன்றன்பின் ஒன்றாக வரிசை முறையில் செய்யப்படுகின்றன. அடிப்படையான கட்டுப்பாட்டுக் கட்டமைப்புகளுள் வரிசைமுறையும் ஒன்றாகும்.

4.1.1.1 கிளைபிரித்தல் (Branching)

இருவழிக் கிளைபிரித்தல் (Two-way branching)

ஒரு வினாவைக் கேளுங்கள். 'ஆம்' அல்லது 'இல்லை' என்னும் விடையைப் பெறுங்கள். விடைக்கேற்ப, இரண்டு பாதைகளுள் ஒன்றில் கிளைபிரித்து அனுப்புங்கள். இது ஒரு சாய்சதுரப் பெட்டி மூலம் விளக்கிக் காட்டப்படுகிறது. பல பாய்வுப்படங்களில் இத்தகைய பெட்டியைப் பார்த்திருப்பீர்கள். ஒரே பாய்வுப்படத்தில் பல இடங்களில் பார்த்திருப்பீர்கள். கிளைபிரித்தல் என்பதும் அடிப்படையான கட்டுப்பாட்டுக் கட்டமைப்புகளுள் ஒன்றாகும்.

எடுத்துக்காட்டு:

If A > B Then Print A, Otherwise Print B

என்ற கட்டளை அமைப்பு "If......Then.......Else" கட்டமைப்பு ஆகும். ஒரு பாதையில் செயல் எதுவும் இல்லை எனில், " If..... Then" கட்ட மைப்பைப் பயன்படுத்தலாம். இவ்வமைப்பில் பதில் 'இல்லை' எனில், எதுவும் செய்யாமலேயே, செயல்பாடு அடுத்த கூற்றுக்குச் செல்ல வேண்டும்.

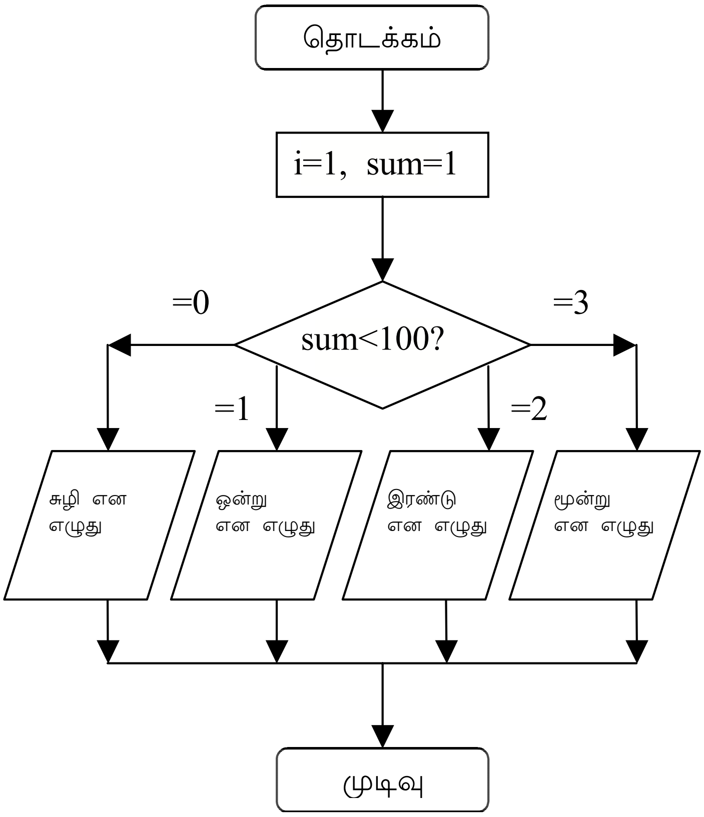
எடுத்துக்காட்டு:

If you find anyone there, then say hello

பலவழிக் கிளைபிரித்தல் (Multi-way Branching)

சில வினாக்களுக்கு, ஆம் அல்லது இல்லை என்கிற பதில் மட்டும்

இருக்க முடியாது. எடுத்துக்காட்டாக, “இந்தப் பையனின் வயது என்ன?” என்ற கேள்விக்கு எத்தனையோ எண்களில் ஒரு எண் பதிலாக இருக்கும். பதிலின் அடிப்படையில், வெவ்வேறு பாதைகளில் சென்று வெவ்வேறு கணிப்பீடுகளைச் செய்ய வேண்டியிருக்கலாம். இது பலவழிக் கிளை பிரித்தல் எனப்படுகிறது. இதனைக் கீழேயுள்ள பாய்வுப்படத்தில் உள்ள வாறு விளக்கிக் காட்டலாம்.



பாய்வுப்படம் 4.11

எடுத்துக்காட்டு:

If n is 0 then print ‘Zero’

1 then print ‘One’

2 then print ‘Two’

3 then print ‘Three’

4.1.2 பன்முறைச் செயல் (Iteration)

நிச்சயித்த பன்முறைச் செயல் (Definite Iteration)

மூன்றாவது அடிப்படையான நுட்பம் ‘பன்முறைச் செயல்’ ஆகும். அதாவது, குறிப்பிட்ட செயல்பாடுகளை மீண்டும் மீண்டும் செய்தல். ஆனாலும், அதே தரவுகள் மீது அதே செயல்பாடுகளைத் திரும்பச் செய்யப் போவதில்லை. அவ்வாறு செய்வது நேரத்தை வீணடிப்பதாகும். எடுத்துக்காட்டாக, நூறு எண்களைப் பெறுதல், ஆயிரம் வாடிக்கையா ளர்களுக்குத் தரவேண்டிய வட்டியைக் கண்டறிதல். இந்த இரண்டு எடுத் துக்காட்டுகளிலும், ஒரே பணியை எத்தனை முறை திரும்பச் செய்ய வேண்டும் என்பதை அறிந்திருக்கிறோம். எனவேதான் இதனை ‘நிச்ச யித்த பன்முறைச் செயல்’ என்கிறோம். இந்த வழிமுறையில், குறிப் பிட்ட செயல்பாடு எத்தனை முறை செய்யப்படுகிறது என்பதை எண்ணிக் கொண்டே வரவேண்டும். இவ்வாறு எண்ணுவதற்குப் பயன்படும் மாறி சுட்டு மாறி (Index veriable) அல்லது கட்டுப்பாட்டு மாறி (Control Variable) என்று அழைக்கப்படுகிறது.

சுட்டு மாறியைப் பயன்படுத்துவதில் நான்கு அடிப்படைப் படி நிலைகள் உள்ளன:

1) சுட்டு மாறியில் தொடக்கத்தில் ஒரு தொடக்க மதிப்பை இருத்த வேண்டும்.

2) சுட்டு மாறியின் இப்போதைய மதிப்பு (v) அதன் இறுதி மதிப்பு டன் ஒப்பிடப்பட்டு, இனிமேலும் திரும்பச் செய்யவேண்டுமா என்பதைத் தீர்மானிக்க வேண்டும்.

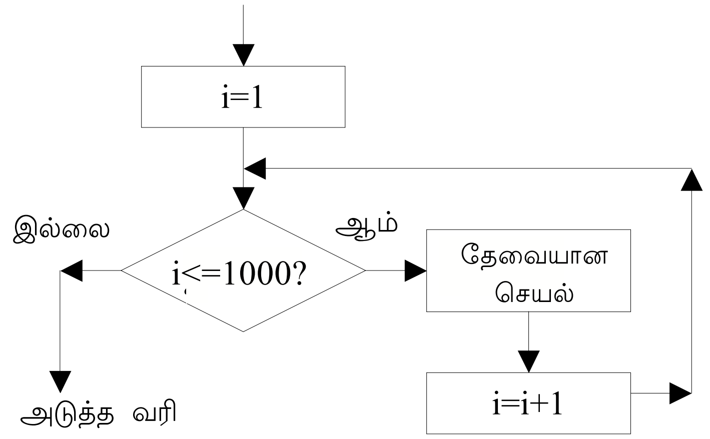
3) விடை ‘ஆம்’ எனில்,

- தேவையான செயல்பாடுகளை ஒருமுறை நிறைவேற்றவும்.
- சுட்டு மாறியின் மதிப்பை ஒன்று கூட்டவும்.
- இரண்டாவது படிநிலைக்குச் சென்று, சரிபார்த்தலை மீண்டும் செய்க.

4) விடை 'இல்லை'எனில்,

- திரும்பச் செய்தல் முடிந்து போனது.
- வரிசையமைப்பில் அடுத்த கட்டளைக்குச் செல்லவும்.

பாய்வுப்படம் 4.12 நிச்சயித்த பன்முறைச் செயலை விளக்குகிறது. படத்தில், திசைவிக்கப்பட்ட கோடுகளால் ஆன ஒரு மடக்கின் (Loop) மூலம் பன்முறைச் செயல் காட்டப்பட்டுள்ளது.



பாய்வுப்படம் 4.12

சில சூழ்நிலைகளில், எத்தனைமுறை திரும்பச் செய்யவேண்டும் என்பதை தொடக்கத்தில் கணிக்க முடியாமல் போகலாம். எடுத்துக்காட்டாக, 1+2+3+..... +n என்பதன் கூட்டுத் தொகை நூறை எட்டும் வகையில், மீச்சிறு n- ன் மதிப்பைக் காண வேண்டும். ஒவ்வோர் எண்ணாகக் கூட்ட வேண்டும். அவ்வாறு கூட்டும்போது நூறை எட்டிவிட்டதா என்பதைச் சோதிக்கவேண்டும். நூறை எட்டியதும் நிறுத்திவிட வேண்டும். எண்களைக் கூட்டும் செயலை எத்தனைமுறை செய்வோம் என்பதை முன்கூட்டியே ஓர் எண்ணிக்கையாகச் சொல்ல முடியாது. இத்தனை முறை என எண்ணிக் கொண்டிருப்பது பயன்தராது. 'நூறை எட்டிவிட்டதா' என்கிற நிபந்தனையைத்தான் பரிசோதிக்க வேண்டும். இத்தகைய பன்முறைச் செயல், 'நிச்சயத்திடா பன்முறைச் செயல்' (Indefenite Iteration) எனப்படுகிறது.

வரிசைமுறைப்படுத்தல் (Sequencing), கிளைபிரித்தல்(Branching), பன்முறைச் செயல் (Interation) ஆகியவற்றைப் பயன்படுத்தி அனைத்துக் கணிப்பீடுகளையும் செய்து முடிக்க முடியும்.

4.1.3 போலிக் குறிமுறை (Pseudo Code)

ஒரு குறிப்பிட்ட பணியை நிறைவேற்றுவதற்கான செயல்முறையை பாய்வுப்படம் மூலமாக வரைந்து காட்டுவதற்குப் பதிலாக, போலிக் குறி முறை மூலமாகவும் எழுதிக் காட்டலாம். 'போலிக் குறிமுறை' என்பது, ஆங்கில மொழித் தொடருக்கும் உயிர்நிலைக் கணிப்பொறி மொழிக் கட்டளைகளுக்கும் இடைப்பட்டது. ஆங்கிலத்தில் தொடர்கள் நீளமாக இருக்கும்; துல்லியமாக இருக்காது. கணிப்பொறி மொழிகளில் கட்டளைத் தொடரமைப்பு கறாராகப் பின்பற்றப்பட வேண்டும். போலிக் குறிமுறை யில் இந்த இரண்டு உறுத்தல்களும் கிடையாது.

மிகச் சில தொடர்வகைகளே இதில் உள்ளன. அவையும் மிகச் சிறியவை. ஆனாலும், எந்தவொரு செயல்முறையையும் வரையறுக்கும் திறன் கொண்டவை. நமக்கு மற்றுமொரு நயமான தொடரமைப்பு வச தியும் உள்ளது. ஒரே தொடரில் பல கூறுகளை ஒன்று சேர்க்க, வழக்க மான அடைப்புக்குறிகளை வழக்கமான பொருளில் பயன்படுத்த முடி யும். வரிகளை உள்தள்ளல் (Indentation) மூலமாகவும், ஒன்றுக்கு மேற் பட்ட கூற்றுகளை ஒன்றுசேர்க்க முடியும். போலிக் குறிமுறையில் எழு தப்படும் விவரங்களை எளிதாகப் புரிந்துகொள்ள முடியும்.

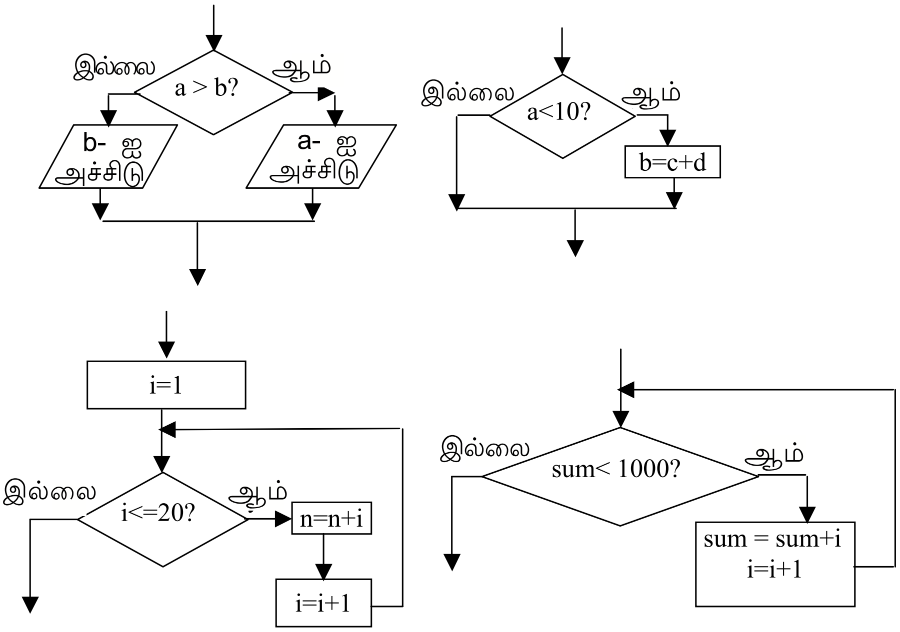
பாய்வுப்படத்தில் நாம் பார்த்த கிளைபிரித்தல், திரும்பச் செய்தல் போன்ற கட்டுப்பாட்டுக் கட்டமைப்புகளை போலிக் குறிமுறையில் இவ் வாறு எழுதலாம்:

* **If.... then............else............**
* **If....... then.............**
* **For............. to........do.....**
* **While.............do.........**

இவற்றுக்கான சில எடுத்துக்காட்டுக் குறிமுறைகள்:

- **If a > b then print a else print b**
- **If** a < 10 **then** b = c + d
- **For i = 1 to 20 do**
  n = n + i
- **While sum < 100 do**
  sum = sum + i
  i = i + 1

While......do எடுத்துக்காட்டின், உள்தள்ளப்பட்ட இரண்டு வரிகளும் ஒன்றாகச் சேர்க்கப்பட்டு, ஒரே செயல்பாட்டுக் கட்டளையாகக் கருதப் படவேண்டும் என்பதைக் கவனத்தில் கொள்க. மேற்கண்ட எடுத்துக் காட்டுகளுக்கான பாய்வுப்படங்களைக் கீழே காண்க:



பாய்வுப்படம் 4.13

அனைத்துக் கட்டுப்பாட்டுக் கட்டமைப்புகளும், ஒரேயொரு நுழைவு முனை, ஒரேயொரு வெளியேறு முனையையே கொண்டுள்ளன என்பதை நோக்குக. அதாவது இக்கட்டளை அமைப்புகளில் குறிப்பிடப்பட்டுள்ள செயல்பாடுகளை நிறைவேற்றி முடித்தபின், கட்டுப்பாடு அடுத்துள்ள கூற்றுக்கு மாற்றப்பட்டு விடும். அதாவது, கணிப்பொறி வரிசைமுறையில் அடுத்து இடம்பெற்றுள்ள கட்டளையை நிறைவேற்றத் தொடங்கும். இது தான் இக்கட்டளை அமைப்புகளின் ஒரு சிறப்புக் கூறாகும். புரிதலையும், பிழைகளைக் கண்டறிந்து நீக்குதலையும் இது எளிதாக்கி விடுகிறது.

முந்தைய பகுதியில் நாம் தீர்த்துவைத்த சிக்கல்களுக்கான போலிக் குறிமுறைகள் கீழே கொடுக்கப்பட்டுள்ளன. பாய்வுப்படங்களையும், போலிக் குறிமுறைகளையும் ஒப்பிட்டு, அவையிரண்டும் ஒத்திருப்பதை உறுதி செய்து கொள்க.

◆ கொள்ளளவு கண்டறிதல்

```
start
read length, breadth and height.
volume = length x breadth x height
print volume
end
```

மேற்கண்ட எடுத்துக்காட்டில், வரிசைமுறைக் கூற்றுகள் மட்டுமே பயன்படுத்தப்பட்டுள்ளன.

◆ எண்களை எழுத்தால் எழுதுதல்

```
start
read n
if n is
        0 then write 'zero'
        1 then write 'one'
        2 then write 'two'
        3 then write 'three'
end
```

◆ மூன்று எண்களில் சிறிய எண்ணைக் கண்டறிதல்

```
start
read a, b, c
if a < b then
        (if a < c then
                print a
        else
                print c
        )
else
        (if b < c then
                print b
        else
                print c )
end
```

◆ இருபடி நிகர்ப்பாட்டைத் தீர்த்தல்

```
start
 read a, b, c
if a = 0 then
        (
         write 'this is not a quadratic equation'
          exit
        )
else
        find d = b2 - 4ac
        if d < 0 then
                write 'imaginary roots'
      else
                if d = 0 then
                        r = -b/a
                        write ' equal roots'
                          write r, r
                else
                        r1 = (-b + d)/ 2a
                        r2 = (-b - d)/2a
                          write 'unequal roots'
                        write r1,  r2
end
```

◆ 100 வரையிலான எண்களின் கூட்டுத் தொகை காணல்

```
start
sum = 0
n = 1
while n <= 100 then do
                read  a
                 sum = sum + a
                n =  n + 1
print sum
end
```

◆ பகா எண்களைக் காணல்

```
start
read n
    for i = 1 to n-1 do
     if i divides n then
                (write 'not a prime'
                 exit program
                )
 write 'prime number'
end
```

◆ கூட்டுத் தொகை ஆயிரத்தை எட்டும் வரை

```
start
 i = 1
 sum = 1
 while sum < 1000 do
           i = i + 1
          sum = sum  + i
 print i and sum
end
```

உங்கள் கவனத்துக்குச் சில குறிப்புகள்

◆ ஒரு If... then....else கூற்றுக்குள் இன்னொரு If.... then..... else கூற்று தரப்பட்டுள்ளது. இதனைத் தெளிவாக உணர்த்தும் பொருட்டு உள்தள்ளி எழுதப்பட்டுள்ளது.

◆ ஒரு கூற்றுக்குள்ளேயே இடம்பெறும் இன்னொரு கூற்றுத்தான் மேலும் உள்தள்ளி எழுதப்பட்டுள்ளது. மற்றபடி வரிசைமுறை அமைப்பிலுள்ள கூற்றுகள் ஒரே அளவாக உள்தள்ளப்பட்டு எழுதப்பட்டுள்ளன.

◆ கணிதத்தில் அடைப்புக் குறிகளைப் பயன்படுத்துவது போன்றே, இங்கேயும் ஒன்று சேர்க்க அடைப்புக் குறிகளைப் பயன்படுத்தி உள்ளோம்.

◆ போலிக் குறிமுறையில் எழுதப்பட்டுள்ள செயல்முறைகள், ஒரு கணிப்பொறி நிரலைப் பெரிதும் ஒத்திருப்பதால், இவற்றை உயர்நிலை மொழிக் கணிப்பொறி நிரல்களாக மாற்றி அமைப்பது மிகவும் எளிதாகும்.

4.1.4 சரிபார்ப்பு (Walkthrough)

ஒரு நிரல் எழுதுவதில் முதல்படி, தீர்வுக்கான வழிமுறையை விவரிக்கும் ஒரு பாய்வுப்படம் அல்லது போலிக் குறிமுறையை உருவாக்குதல் ஆகும். சிக்கலுக்கான ஒரு தீர்வினை வடிவமைப்பதில் இது தொடக்க நிலை ஆகும். பாய்வுப்படம் அல்லது போலிக் குறிமுறையை அடிப் படையாக வைத்து, ஒரு குறிப்பிட்ட மொழியின் கட்டளை அமைப்பு விதிகளுக்கு உட்பட்டு நிரல் எழுதப்படுகிறது. ஒரு நிரலை எழுதுவதற்கு முன்பாக, சிக்கல் தீர்வுக்கான வழிமுறை சரியானதா என்பதைப் பரிசோதிக்க வேண்டும். எனவே, நாம் உருவாக்கிய பாய்வுப்படமும், போலிக் குறிமுறையும் சரியானபடி அமைந்துள்ளனவா என்பதைப் பரிசோதிக்க வேண்டும். பரிசோதனைக்கு முன்பாக, தீர்வுநெறி (Algorithm) என்றால்

என்ன என்று பார்ப்போம். அதன்பிறகு, பாய்வுப்படம் அல்லது போலிக் குறிமுறையைப் பரிசோதிக்க ஒரு வழிமுறையைக் காண்போம்.

'தீர்வுநெறி' என்பது, கீழ்க்காணும் பண்புகளைக் கொண்ட ஒரு செயல் முறை ஆகும்:

- வரம்புறு (finite) எண்ணிக்கையிலான படிநிலைகள் (steps) இருக்க வேண்டும்.

- ஒவ்வொரு படிநிலையும், எவ்விதக் குழப்பமுமின்றி செயல்ப டுத்தும் நிலையில் இருக்க வேண்டும்.

- ஒவ்வொரு படிநிலையும், வரம்புக்குட்பட்ட நினைவகப் பகுதியைப் பயன்படுத்தி, வரம்புக்குட்பட்ட நேரத்துக்குள் செயல்படுத்தும் நிலையில் இருக்க வேண்டும்.

- முழுமையான நிரலும், வரம்புக்குட்பட்ட நேரத்துக்குள் இயங் கும்படி இருக்க வேண்டும்.

ஒரு தீர்வுநெறிக்கான பாய்வுப்படம் அல்லது போலிக் குறிமுறை கொடுக்கப்பட்டுள்ளதாக வைத்துக் கொள்வோம். இதனை அடிப்படை யாகக் கொண்டு ஒரு கணிப்பொறி எவ்வாறு செயல்படும் என்பதைப் பரிசோதிக்கும் வழிமுறையை, ஒத்திகை அல்லது சரிபார்ப்பு எனலாம். ஆங்கிலத்தில் walkthrough என்றழைக்கப்படுகிறது. எடுத்துக்காட்டாக, ஒரு பாய்வுப்படத்தை எடுத்துக் கொள்வோம். உங்களை ஒரு கணிப்பொறி யாக உருவகித்துக் கொள்ளுங்கள். தொடங்கு (Start) எனக் குறிப்பிட் டுள்ள பெட்டியிலிருந்து தொடங்க வேண்டும். இந்த இடத்திலிருந்து, அம்புக்குறிகளால் குறிக்கப்பட்ட பாதையில் நீங்கள் பயணிக்க வேண் டும். ஒரு செவ்வகப் பெட்டிக்குள் நுழைந்தால், தேவையான கணிப்பீடு களைச் செய்யவேண்டும்.

அது ஓர் உள்ளீட்டுப் பெட்டி எனில், குறிப்பிட்ட மாறிகளுக்குரிய மதிப்புகளை எடுத்துக் கொள்ளவேண்டும். ஒவ்வொரு முறையும் வெவ் வேறு தொகுதி மதிப்புகளை எடுத்துக் கொள்ளவேண்டும். அப்பொழுது தான் வெவ்வேறு பாதை வழியே பயணித்துப் பரிசோதிக்க முடியும். சிறிய நிரல்களுக்கு, தொடக்கத்திலிருந்து முடிவுவரை இருக்கின்ற அனைத்துப் பாதைகளிலும் பயணிப்பது கடினமான பணியன்று. ஒரு வழிமுறை சரி யானதா என்பதை உறுதிப்படுத்துவதில் இது ஒரு படிநிலையாகும்.

அது ஒரு வெளியீட்டுப் பெட்டி எனில், மாறிகளின் தற்போதைய மதிப்புகளை வெளியிட வேண்டும்.

4.1.5 ஒரு நிரலை உருவாக்குதல்

ஒரு சிறிய நிரலை எழுதுவது எளிது. எனவே, ஒரு சிறிய நிரலுக்கான போலிக் குறிமுறை அல்லது பாய்வுப்படத்தை முதலில் வரைவோம். அதன் அடிப்படையில் நிரலை எழுதுவது எளிது. ஆனாலும், நடைமுறை வாழ்க்கையில் நாம் காணும் மிகப்பெரிய சிக்கல்களுக்கு இந்த வழிமுறை உதவாது. அதற்கு, முறைப்படியான ஓர் அணுகுமுறை மிகமிகத் தேவையாகும்.

ஒரு நிரலில் ஏராளமான விவரங்களைத் தர வேண்டியிருக்கும். ஆனால், மிகப் பெரிய முயற்சியில் இறங்கி, மிக நுணுக்கமான விவரங்களைப் பற்றியெல்லாம் சிந்திக்கத் தொடங்கினால், உண்மையான பணியில் கவனம் செலுத்த முடியாமல் போய்விடும். இப்படிப்பட்ட சூழ்நிலையில் ஒருவர், மிக முக்கியமான பணிகளை மட்டும் அவரே பார்த்துக் கொள்ளவேண்டும். சிறுசிறு பகுதிகளை அவருக்குக் கீழ் பணி புரிபவர்களிடம் விட்டுவிட வேண்டும்.

ஒரு பெரிய பணியைச் செய்து முடிப்பதற்கு இதுதான் சரியான முறையாகும். ஒற்றுமையின் வலிமையை நாமறிவோம். இதை விளக்கும் பல கதைகளைப் படித்திருக்கிறோம். எடுத்துக்காட்டாக, சிறிய குச்சிகளை ஒன்றாகக் கட்டிவைத்தால் உடைக்க முடியாது. ஆனால், அவற்றைத் தனித்தனியே பிரித்தெடுத்தால் மிக எளிதாக முறித்துவிட முடியும். இவ்வாறு பிரித்துவைத்து வெற்றிகொள்ளும் அணுகுமுறை பல இடங்களில் பயன்படுத்தப்படுகிறது. ஒரு பெரிய நிரலை எழுதுவதற்கு இதே வழிமுறைதான் மிகவும் பயன்தரக் தக்கதாகும்.

ஒரு நிரலை உருவாக்குவதற்கு முன்பாக, அந்தச் சிக்கலைச் சிறிய சிக்கல்களாகப் பிரித்துக் கொள்ளவேண்டும். அத்தகைய சிறிய சிக்கல்களுக்குக் கண்டறியப்படும் விடைகளை ஒன்றுசேர்த்து, பெரிய சிக்கலுக்கு விடைகாணும் வழிமுறையை நாம் அறிந்து வைத்திருக்க வேண்டும். ஒரு நிரலை உருவாக்குவதில் இது ஒரு படிநிலை ஆகும். ஒரு பாய்வுப்படம் அல்லது போலிக் குறிமுறை வரைய முடிகிற அளவுக்குச் சிறிதாகும் வரை, பெரிய சிக்கலை மீண்டும் மீண்டும் உடைத்துப் பிரித்துக் கொண்டே வரவேண்டும்.

நமக்கு ஏற்கெனவே பழக்கமான போலிக் குறிமுறையைப் பயன்படுத்தியே, சார்பு நிரல்களை ஒருங்கிணைக்கும் தருக்க முறையையும் எழுதிவிட முடியும். பாய்வுப்படங்களின் மூலமாகவும் இதனைச் சாதிக்க முடியும். எனவே, பல போலிக் குறிமுறைகளையும் பாய்வுப் படங்களையும் பயன்படுத்தி, முழு நிரலின் வழிமுறை நுட்பத்தையும் எழுதி

விட முடியும். அதன்பிறகு, இவற்றைக் கணிப்பொறி நிரலாக மாற்றுவது மிகவும் எளிய பணியாகிவிடும்.

ஒரு நிரலை உருவாக்குவதில் இதுபோன்ற அணுகுமுறை 'மேலிருந்து கீழ் அணுகுமுறை' (Top Down Approach) எனப்படுகிறது. ஒவ்வொரு படிநிலையிலும், ஒரேயொரு சிறுபணியில் மட்டுமே கவனம் செலுத்துகிறோம். சிறிய சிக்கல்களின் விடைகளை ஒருங்கிணைத்துத் தீர்வினை அடைவது எப்படி என்பதைக் கண்டறிகிறோம். கணிப்பொறி நிரல்களைப் பொறுத்தவரை, இந்த வழிமுறையை முதலில் பயன்படுத்தும்போது, செயல்முறைகளுக்கே (procedures) முக்கியத்துவம் தருகிறோம், அதாவது, எப்படிச் செயல்படுத்துவது என்கிற வழிமுறைகளுக்கே முக்கியத்துவம் தருகிறோம், தரவுகளுக்கு (data) அல்ல. இதுவே 'கட்டமைப்பு நிரலாக்கம்' எனப்படுகிறது. ஒரு சிக்கலைச் சிறிய சிக்கல்களாக உடைக்கும்போது, செயல்முறையோடு தரவுகளும் கணக்கில் எடுத்துக் கொள்ளப்படுமாயின், அதனை 'பொருள்நோக்கு அணுகு முறை' (Object Oriented Approach) என்கிறோம். நிரல்களை உருவாக்குவதற்கு இதுவே மிகவும் ஏற்புடைய அணுகுமுறை எனக் கண்டறியப்பட்டுள்ளது. பொருள்நோக்கு அணுகுமுறையின் அடிப்படையில் உருவாக்கப்படும் நிரல்களை எழுதுவதற்கு, சி++ மற்றும் ஜாவா போன்ற கணிப்பொறி மொழிகள் உதவுகின்றன. இந்த அணுகுமுறைபற்றி நாம் விரிவாகப் படிக்கப் போவது இல்லை.

மேலிருந்து கீழ் அணுகுமுறையின் பயன்பாட்டை ஓர் எடுத்துக் காட்டின் மூலம் விளக்க இருக்கிறோம். எந்தவொரு நிரலை எழுதுவதற்கும் இதே வழிமுறையைப் பின்பற்றுங்கள்.

சிக்கல்: நூறு எண்களை ஏறுமுக வரிசையில் ஒருங்கமைக்க ஒரு நிரல் எழுதுக.

வழிமுறை:

- நூறு எண்களைப் பெற்று a1, a2,....... ஆகியவற்றில் இருத்துக.
- அவற்றுள் மீச்சிறு எண்ணைக் கண்டறிக.
- இந்த எண்ணை முதல் எண்ணுடன் இடமாற்றம் செய்க.
- இதேபோல, இரண்டாம் எண், மூன்றாம் எண்ணிலிருந்து தொடங்கி 98 முறை (ஏன் என்று சொல்ல முடியுமா?) மேற்கண்ட இரண்டு படிநிலைகளை திரும்பத் திரும்பச் செய்யவும்.
- நூறு எண்களையும் திரையில் காட்டவும்

மேற்கண்ட படிநிலைகளை முறைப்படியான போலிக் குறிமுறைக் கட்டளைகளாக எழுதிப் பார்ப்போம்.

```
Read 100 numbers and put them in an array, as a(1), a(2) etc.

Do for i = 1 to 99

  Find the smallest of a(i) to a(100)
  Let the smallest number be at the jth position
  Interchange a(i) and a(j)

Output a(1), a(2) ... a(100)
```

கீழேயுள்ள சிறிய பணிகளை நம்மால் செய்து முடிக்க முடியுமெனில், ஒட்டுமொத்தச் சிக்கலுக்கும் எளிதாகத் தீர்வு கண்டுவிட முடியும்.

- நூறு எண்களை உள்ளீடாகப் பெறுதலும், வெளியிடுதலும்
- கொடுக்கப்பட்ட எண் தொகுதியில் மீச்சிறு எண்ணைக் கண்டறிதல்
- இரண்டு எண்களை இடம் மாற்றுதல்

இந்தப் பணிகளை எவ்வாறு செய்துமுடிக்கப் போகிறோம் என்கிற விவரங்களைப் பற்றி இப்போது நாம் கவலைப்பட போவதில்லை. அடுத்த கட்டச் செயலாக்கத்தில் இதைக் கவனிப்போம். இப்போது இரண்டாம் படிநிலையில், மேற்கண்ட சிறுபணிகள் ஒவ்வொன்றாக எடுத்துக் கொண்டு தீர்வுகாண முயல்வோம்.

நூறு எண்களைப் படித்தல்:

```
For Count = 1 to 100 do
        Read a(count)
```

இதே வழிமுறையை நூறு எண்களை வெளியிடவும் பயன்படுத்தலாம்.

a(1) முதல் a(100) வரையிலான **நூறு எண்களில் மீச்சிறு எண்ணைக் கண்டறிதல்:**

```
Let  i = 1
Let position = 1
Let min = a(i)

For n = i+1 to 100 do
        If a(n) < min then
                min = a(n)
                position = n

Let j=position
```

a(i) மற்றும் a(i) இரண்டின் மதிப்புகளையும் இடம் மாற்றுதல்:

Let temp = a(i)
a(i) = a(j)
a(j) = temp

இங்கே, இவ்வாறு எழுதினால் பிழையாகிப் போகும்.

a(i) = a(j)

a(j) = a(i)

காரணம், முதல் கட்டளை நிறைவேற்றப்பட்டவுடனே a(i)- ன் பழைய மதிப்பு இழக்கப்படுகிறது.

4.2 சி–மொழி நிரலாக்கம் – ஓர் அறிமுகம்

நிரலாக்கத்துக்கு மிகவும் பரவலாகப் பயன்படுத்தப்படும் செல்வாக்குப் பெற்ற நிரலாக்க மொழி சி-மொழியாகும். முப்பது ஆண்டுகளுக்கும் முன்னால் ஏடீ&டீ பெல் ஆய்வுக் கூடத்தில் டென்னிஸ் எம்.ரிட்சி, சி-மொழியை உருவாக்கினார். தொடக்கத்தில், யூனிக்ஸ் இயக்க முறைமையோடு இணைந்து பயன்படுத்துவதற்கென வடிவமைக்கப்பட்ட போதிலும், சி-மொழி ஒரு பொதுப்பயன் மொழியாகத் திகழ்கிறது. சி-மொழி, செயல்திறன் மிக்கது; நெகிழ்வுத் தன்மை கொண்டது; பெயர்வுத்திறன் (Portability) பெற்றது. ஒருவகைக் கணிப்பொறியில் இயக்குவதற்காக எழுதப் பட்ட மென்பொருளை, எவ்வித மாற்றமும் இன்றியோ அல்லது சிறிதளவு மாற்றத்துடனோ வேறுவகைக் கணிப்பொறியில் நிறுவி இயக்க முடியும். இத்தகு பண்பையே 'பெயர்வுத் திறன்' என்கிறோம். சி-மொழி பல்வேறு பட்ட செயற்குறிகளையும் (Operators) கட்டளைகளையும் (Commands) கொண்டுள்ளது. இயக்க முறைமைகள் (Operating Systems), நிரல் பெயர்ப்பிகள் (Compilers), உரைச் செயலிகள் (Text Processors), தரவுத்தள மேலாண் முறைமைகள் (Database Management Systems) போன்ற பல விதமான முறைமை மென்பொருள்களை சி-மொழியில் உருவாக்க முடியும். அது மட்டுமின்றி பயன்பாட்டு மென்பொருள்களை (Application Software) உருவாக்குவதற்கும் சி-மொழி மிகவும் ஏற்றது.

சி-மொழியானது கீழ்க்காணும் அடிப்படை வகை உறுப்புகளால் ஆனது:

- மாறிலிகள் (Constants)
- குறிப்பெயர்கள் (Identifiers)
- நிறுத்தற்குறிகள் (Punctuation)
- சிறப்புச் சொற்கள் (Keywords)

இந்த உறுப்புகள் அனைத்தையும் சேர்த்து மொத்தமாக வில்லைகள் (tokens) என்கிறோம். நிரல்பெயர்ப்பியானது மேலும் சிறிய உறுப்பு களாகக் கூறாக்க முடியாத, மூல நிரலின் உரைப்பகுதியே வில்லை எனப்படுகிறது.

number = number + 1;

என்னும் கூற்றை எடுத்துக் கொள்ளுங்கள். இதிலுள்ள வில்லைகள் வரு மாறு:

number	-	குறிப்பெயர் (மாறி)
=	-	செயற்குறி
+	-	செயற்குறி
1	-	மாறிலி
;	-	நிறுத்தற்குறி

ஆக, மேற்கண்ட கூற்று, குறிப்பெயர், மாறிலி, செயற்குறி போன்ற வில்லைகளின் தொகுப்பாகும். சி-மொழி நிரலாக்கத்தில் if, while, for போன்ற சொற்களைச் சிறப்புச் சொற்கள் என்கிறோம். அவற்றைக் குறிப்பிட்ட பயன்பாடுகளுக்கு மட்டுமே பயன்படுத்திக் கொள்ளமுடியும். இந்தப் பாடத் தில் பின்னால் வரும் பகுதியில் சிறப்புச் சொற்களைப் பற்றிப் படிப் போம்.

4.2.1 மாறிலிகள் (Constants)

மாறிலி என்பது எண்வகையாக இருக்கலாம். அல்லது எண்வகை அல்லாததாகவும் இருக்கமுடியும். ஓர் எண்ணாகவோ, ஓர் எழுத்தா கவோ, ஓர் எழுத்துச் சரமாகவோ இருக்கலாம். மாறிலியை ஒரு நிரலில் தரவு மதிப்பாகப் பயன்படுத்திக்கொள்ள முடியும். மாறிலி என்கிற பெய ருக்கேற்ப, ஒரு முறை வரையறுக்கப்பட்ட மதிப்பை மாற்றியமைக்க முடியாது. மாறிலி என்பது மாற்ற முடியாதது. எண்வகைத் தரவு பெரும் பாலும் எண்களால் ஆனது. பதின்மப் புள்ளி (decimal point) இடம்பெற லாம். எண்வகை அல்லாத தரவு என்பது எண்கள், எழுத்துகள், இட வெளிகள், முறைமை ஏற்கின்ற வேறெந்த சிறப்புக் குறிகளையும் கொண் டிருக்கலாம். வேறு வகையில் சொல்வதெனில் எழுத்தெண் (alphanu-meric) குறியீடுகளைக் கொண்டிருக்கும். இவ்வகைத் தரவை மதிப்புரு (literal) எனவும் கூறலாம். மாறிலிகள் ஒரு தரவினத்தையும், ஒரு மதிப்பை யும் உணர்த்தி நிற்கின்றன. எண்வகை மாறிலி மூவகைப்படும்:

- முழுஎண் மாறிலி (Integer Constant)
- மிதவைப் புள்ளி மாறிலி (Floating - point constant)
- குறியுரு மாறிலி (Character constant)

சி-மொழியின் அடிப்படைத் தரவினங்கள் பற்றி இப்பாடத்தின் பிற் பகுதியில் படிப்போம்.

4.2.1.1 முழுஎண் மாறிலி (Interger Constant)

முழுஎண் மாறிலி ஒரு பதின்ம எண் (அடியெண் 10) ஆகும். தொகை மதிப்பைக் குறிக்கும் முழு எண் ஆகும். 0 முதல் 9 வரையி லான இலக்கங்களால் ஆனது. ஒரு முழுஎண் மாறிலி 0x அல்லது 0X எனத் தொடங்கினால் அது ஒரு பதினறும (அடியெண் 16) மாறிலி ஆகும். 0 எனத் தொடங்கினால் எண்ம (அடியெண் 8) மாறிலி ஆகும். இவ்வாறெல்லாம் இல்லையெனில் பதின்ம எண் என்றே கொள்ளப் படும்.

23, 36, 48 ஆகியவை பதின்ம (decimal) மாறிலிகள்

0 x1C, 0XAB, 0x23 ஆகியவை பதினறும (hexadecimal) மாறிலிகள்

071,023, 035 ஆகியவை எண்ம (octal) மாறிலிகள்

முழுஎண் மாறிலிகளைப் பொறுத்தவரை கழித்தல் குறியுடன் இல்லை யெனில் நேர்மஎண் (Positive) ஆகவே கொள்ளப்படும். -18 என்பதும் ஏற்கத்தகு முழுஎண் மாறிலியே. 2,345 என்னும் மாறிலி, ஏற்புடையது அன்று. காரணம் இந்த எண்ணில் சிறப்புக் குறி (காற்புள்ளி) இடம் பெற் றுள்ளது.

4.2.1.2 மிதவைப் புள்ளி மாறிலி (Floating Point Constant)

மிதவைப் புள்ளி மாறிலி என்பது ஒரு குறியிட்ட மெய் எண் (signed real number) ஆகும். இவ்வகை எண், முழுஎண் பகுதி, பதின்மப் புள்ளி கள், பின்னப் பகுதி, அடுக்கெண் பகுதி ஆகியவற்றைக் கொண்டிருக் கும். ஒரு மிதவைப் புள்ளி மாறிலியைக் குறிப்பிடும்போது, பதின்மப் புள்ளிக்கு முந்தைய இலக்கங்கள் (முழுஎண் பகுதி) அல்லது பதின்மப் புள்ளிக்கு அடுத்திருக்கும் இலக்கங்கள் (பின்னப்பகுதி) இல்லாமல் இருக் கலாம். ஆனால் இரண்டுமே இல்லாமல் இருக்க முடியாது. அடுக்கெண் பகுதி சேர்க்கப்படுமாயின் பதின்மப் புள்ளியை விட்டுவிடலாம். அடுக் கெண் பகுதி, பதின்ம எண் அமைப்பில் 10-இன் அடுக்குகளைக் கொண் டிருக்கும்.

எடுத்துக்காட்டு:

58.64 என்பது ஒரு மிதவைப் புள்ளி மெய்யெண் மாறிலி ஆகும். இதனை அடுக்கெண் முறையில் இவ்வாறெல்லாம் எழுதலாம்:

5.864E1 => 5.864 X 10^1 => 58.64
5864E-2 => 5864 X 10^{-2} => 58.64
0.5864e2 => 0.5864 X 10^2 => 58.64

E அல்லது e என்னும் எழுத்துகள், மிதவைப் புள்ளி மாறிலி அடுக்கெண் (exponent) வடிவில் உள்ளதை உணர்த்துகிறது.

4.2.1.3 குறியுரு மாறிலி (Character Constant)

'குறியுரு' (Character) என்பது அகரவரிசை எழுத்துகள், எண்கள், கணிப்பொறி முறைமை கையாளும் பிற சிறப்புக் குறியீடுகள் ஆகியவற்றை உள்ளடக்கியதாகும். மேற்கண்ட குறியுருக்களின் தொகுதியை கணிப்பொறி முறைமையின் குறியுருத் தொகுதி (Character Set) என்கிறோம். இத்தொகுதியிலுள்ள ஒரு குறியுருவை ஒற்றை மேற்கோள்குறிகளுக்குள் அமைத்தால் அது ஒரு குறியுரு மாறிலியைக் குறிக்கும். சி-மொழியில் பயன்படுத்தப்படும் குறியுருக்களை மூன்று பிரிவுகளில் வகைப்படுத்தலாம்.

- எழுத்து வகைக் குறியுருக்கள்: a,b,c,.........., z, A,B,C,...........,Z
- எண்வகைக் குறியுருக்கள்: 0 முதல் 9 வரை
- சிறப்புக் குறியுருக்கள்: +,-, * / % # = , .' " ( ) [ ] ; : ?

எடுத்துக்காட்டு:

'1', 'a', '+', '-' ஆகியவை ஏற்புடைய குறியுரு மாறிலிகள் ஆகும்.

ஒரு குறியுரு மாறிலியைக் குறிப்பிட இரண்டு ஒற்றை மேற்கோள் குறிகளைப் பயன்படுத்துகிறோம். அப்படியெனில், ஒற்றை மேற்கோள் குறியை எப்படி ஒரு குறியுரு மாறிலியாகக் குறிப்பிடுவது? ஒற்றை மேற்கோள் குறியை இரண்டு ஒற்றை மேற்கோள் குறிகளுக்குள் அமைத்து ''' என எழுதுவது ஏற்றுக் கொள்ளப்படாது.

ஒற்றை மேற்கோள் குறியை ஒரு குறியுரு மாறிலியாகக் குறிப்பிட விடுபடு வரிசையைப் (escape sequence) பயன்படுத்தலாம். பின்சாய்வு கோட்டை அடுத்து ஓர் எழுத்து சேர்ந்த குறியுருச் சேர்மானங்களை 'விடுபடு வரிசைகள்' என்கிறோம்.

' \'' என்பது ஏற்புடைய ஒற்றை மேற்கோள் குறியுரு மாறிலி.

அதேபோல, விடுபடு வரிசையைப் பயன்படுத்தி, சில அச்சிடவிய லாத குறியுருக்களைக் குறிப்பிடலாம்.

எடுத்துக்காட்டுகள்:

'\a' மணி **(Beep)**

'\b' பின்னிடவெளி **(Back Space)**

'\f' படிவச் செலுத்துகை **(Form Feed)**

'\r' செலுத்தி திருப்பல் **(Carriage Return)**

'\n' புதிய வரி **(New Line)**

'\0' இன்மக் குறியுரு **(null character)**

பின்சாய்வுக் கோட்டையே ஒரு குறியுரு மாறிலியாகக் குறிப்பிட, இரண்டு பின்சாய்வுக் கோடுகளைப் ('\\') பயன்படுத்த வேண்டும்.

4.2.1.4 சர நிலையுரு (String literal)

சர நிலையுரு அல்லது சர மாறிலி என்பது, முறைமையின் குறியுருத் தொகுதியிலுள்ள குறியுருக்களின் சரத்தை இரட்டை மேற்கோள் குறிகளுக்குள் தருவதாகும். முன்னியல்பாக (by default), இன்மக் குறியுரு ('\0') சர நிலையுருவின் இறுதிக் குறியுருவாகச் சேர்த்துக் கொள்ளப்படும். ஒரு சர மாறிலியின் பகுதியாக இரட்டை மேற்கோள் குறியும் இடம்பெற வேண்டுமெனில் விடுபடு விசையைப் ('\"') பயன்படுத்த வேண்டும்.

"hello" என்பது ஏற்புடைய சர நிலையுரு. இந்தச் சர நிலையுருவில் உள்ள எழுத்துகளின் சரியான எண்ணிக்கை, இறுதியில் உள்ள இன்மக் குறியுருவையும் சேர்த்து 6 ஆகும். இன்மக் குறியுரு கண்ணில் புலப் படாது. நினைவகத்தில் இந்தச் சரத்தை இருத்திவைக்க ஆறு பைட்டுகள் தேவை. என்றாலும் இச்சரத்தின் நீளம் (length) என்பது 5 எழுத்துகளே ஆகும்.

4.2.2. குறிப்பெயர்கள் (Identifiers)

ஒரு நிரலில் உள்ள மாறிகள் (Variables), செயல்கூறுகள் (functions), தரவினங்கள் (data types), சிட்டைகள் (labels) ஆகியவற்றைக் குறிக்கும் பெயர்கள் குறிப்பெயர்கள் எனப்படுகின்றன. ஒரு மாறியின் பெயர் எழுத்து கள், எண்களைக் கொண்டிருக்கலாம். பிற குறியுருக்கள் இடம்பெறக் கூடாது. மாறியின் பெயரில் அடிக்கீறு (underscore) மட்டும் இடம் பெறலாம். குறைந்த அளவு ஓரெழுத்து. அதிக அளவாக 32 எழுத்துகள் இடம்பெற லாம். முதல் எழுத்து, அகர வரிசை எழுத்தாகத்தான் (alphabet) இருக்க வேண்டும். எண்கள் இடம்பெறக் கூடாது. ஏற்புடைய மாறிப் பெயர்கள்:

x

length

x_value

y_value

a123

சி-மொழியின் சிறப்புச் சொற்களை (கட்டளைச் சொற்கள்) குறிப் பெயர்களாகப் பயன்படுத்தக் கூடாது. கீழ்க்காணும் பெயர்கள் மாறியின் பெயர்களாக ஏற்க இயலாதவை. அதற்கான காரணம் தரப்பட்டுள்ளது.

123	**-**	முதல் குறியுரு எண்ணாகும்.
1abc	**-**	முதல் குறியுரு எண்ணாகும்.
x value	**-**	பெயரில் இடவெளி **(space)** உள்ளது.
x &y	**-**	ஏற்கவியலாத சிறப்புக் குறியுரு (&) இடம் பெற்றுள்ளது.
for	**-**	சி-மொழியின் சிறப்புச் சொல்

4.2.3 அடிப்படைத் தரவினங்கள் (Fundamental Data Types)

நிரலாக்கத்தில் அதிகமாகப் பேசப்படும் சொற்களுள் 'தரவு' என்பதும் ஒன்று. கணிப்பொறிக்கு உள்ளீடாகத் தரப்படும் செயலாக்கம் பெறாத தகவலே, 'தரவு' என வரையறுக்கப்படுகிறது. சி-மொழியில் 'தரவு' என்பது பல்வேறு இனமாக வகைப்படுத்தப்பட்டுள்ளது. எண்வகைத் தரவுகள் int, float, char என மூன்று இனங்களாகப் பிரிக்கப்பட்டுள்ளன. இவை அடிப்படைத் தரவினங்கள் அல்லது முதனிலைத் தரவினங்கள் என்று அழைக்கப்படுகின்றன. சி-மொழியில் இவை முன் வரையறுக்கப்பட்டவை (pre-defined). இவற்றை சி-நிரல்பெயர்ப்பி (C-Compiler) அடையாளம் கண்டுகொள்ளும்; இவற்றின் மதிப்புகளைச் செயல்பாடுகளில் பயன்படுத்திக் கொள்ளும். எனவே இத்தரவினங்களை மூலத் தரவினங்கள் (primitive data types) என்றும் அழைப்பதுண்டு.

மேற்கண்ட தரவினங்களில் மாறிகளை அறிவிக்கும் முறை:

```
int x;
float f;
char ch;
```

இவற்றுள் int என்பது முழுஎண்ணையும் (Interger), float என்பது மெய் (real) எண்ணையும், char என்பது ஒற்றைக் குறியுருவையும் குறிக்கின்றன.

எடுத்துக்காட்டு:

1 முழு எண் **(int)**

1.0 மெய்யெண் **(float)**

'1' குறியுரு மாறிலி **(char constant)**

"1" சர நிலையுரு **(string literal)**

சி-மொழியில் 'சரம்' (string) எனத் தனியான தரவினம் இல்லை. எனினும் சர நிலையுருவுடன் தொடர்புடைய தரவினம் char * (குறியுருச் சுட்டு- character pointer) ஆகும். இதைப் பற்றிய விளக்கம், இப்பாடத்தில் 'அணிகள்' (Arrays) பற்றிய பகுதியில் தரப்பட்டுள்ளது.

int மதிப்பை நினைவகத்தில் இருத்திவைக்க 2 பைட்டுகள் தேவை. float மதிப்புக்கு 4 பைட்டுகளும், char மதிப்புக்கு 1 பைட்டும் தேவை.

4.2.3.1 தருவிக்கப்பட்ட தரவினங்கள் (Derived Data Types)

long, double, unsigned மற்றும் அணிகள்(Arrays), சுட்டுகள்(Pointers) ஆகியவை மூலத் தரவினங்களிலிருந்து தருவிக்கப்பட்ட தரவினங்கள் ஆகும். long என்பது int லிருந்தும், double என்பது float- லிருந்தும் தருவிக்கப்பட்டவை. long int மாறியை அறிவிக்கும் முறை:

long int i ;

அல்லது

long i;

long இன மதிப்பை நினைவகத்தில் இருத்த, 4 பைட்டுகள் தேவை. மெய்யெண்களை (real) மிகவும் துல்லியமாகக் கையாள வேண்டுமெனில் double இனத்தைப் பயன்படுத்த வேண்டும். இது நினைவகத்தில் 8 பைட்டுகளை எடுத்துக்கொள்ளும்.

unsigned int என்பது int இனத்தைப் போலவே 2 பைட்டுகளையே எடுத்துக் கொள்ளும். ஆனால் அதிலுள்ள அனைத்து பிட்டுகளும் மதிப்பினை இருத்தப் பயன்படுத்திக் கொள்ளப்படும். ஆனால் சாதாரண int மதிப்பில், உயர்மதிப்பு பிட் (இடது ஓர பிட்) நேர்ம (positive) அல்லது எதிர்ம (negative) எண் என்பதைக் குறிக்கும் குறி (sign) பிட் ஆகும். int- ன் மதிப்பு - 32768 முதல் +32767 வரை. unsigned int- ன் மதிப்பு 0 முதல் 65535 வரை. இதுபோலவே unsigned long, unsigned char ஆகிய தரவினங்களும் உள்ளன.

4.2.3.2. சுட்டு மாறிகள் (Pointer Variables)

சி-மொழியின் மாறிகளை சாதாரண மாறிகள் (ordinary variables), சுட்டு மாறிகள் (pointer variables) என வகைப்படுத்தலாம். சாதாரண மாறி அது தொடர்புடைய இனத்தின் மதிப்புகளை எடுத்துக்கொள்ளும்.

எடுத்துக்காட்டு:

int x;

இங்கே, x என்பது, int இனத்தில் ஒரு சாதாரண மாறி; முழு எண்ணை அதன் மதிப்பாக ஏற்றுக்கொள்ளும்.

x = 10;

சுட்டு மாறி அறிவிக்கப்படும் முறை:

int *y ;

மேற்கண்ட அறிவிப்பில் y என்பது சுட்டு மாறி. இதன் இனம் - முழுஎண் சுட்டு (int *).

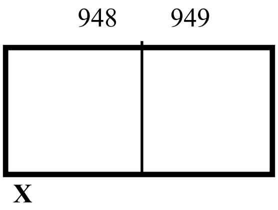
சுட்டு மாறி எப்போதும் ஒரு முகவரியையே மதிப்பாக ஏற்கும். பொதுவாக, ஒவ்வொரு மாறியும், அது சார்ந்த இனத்துக்கு ஏற்ப, நினைவகத்தில் இடம்பிடித்துக் கொள்கிறது. முதன்மை நினைவகத்திலுள்ள ஒவ்வோர் இருப்பிடத்தையும் அதன் முகவரி மூலம் அணுக முடியும்.

மேற்கண்ட எடுத்துக்காட்டில் x என்பது சாதாரண மாறி, y என்பது சுட்டு மாறி. x என்பது ஒரு சாதாரண முழுஎண், y என்பது ஒரு முழு எண்ணுக்கான சுட்டு. எனவே, x இன் முகவரியை y -ல் இருத்த முடியும்.

y = &x;

சுட்டுகளைப் பொறுத்தவரை இரண்டே இரண்டு செயற்குறிகளே உள்ளன. ஒன்று முகவரி சுட்டு (address of -&) செயற்குறி. மற்றொன்று உள்நோக்கு (indirection-*) செயற்குறி. இரண்டுமே ஒருமச் செயற்குறிகள் (unary operators). ஒரேயொரு மாறியின் மீது மட்டுமே செயல்படுபவை. ஒருமச் செயற்குறிகள் பற்றி அடுத்த பகுதியில் படிப்போம்.

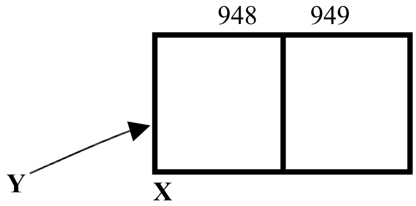
x ஒரு முழுஎண் என்பதால், நினைவகத்தில் இரண்டு பைட்டுகளை எடுத்துக் கொள்ளும். அதன் முகவரிகளை கீழே உள்ளவாறு படமாக உருவகிக்கலாம்:



x- ன் முகவரி 948 என்பதைப் படத்தில் காண்க. முதன்மை நினைவகத்தில் ஒவ்வொரு பைட்டையும் அதன் முகவரி மூலம் அணுக முடியும். நினைவகத்தில் ஒரு மாறி இருக்கும் இடத்தின் முகவரியை அறிய முகவரி சுட்டு (address of - &) செயற்குறியைப் பயன்படுத்த வேண்டும்.

y = &x;

என்ற கூற்றில், x- ன் முகவரி y என்னும் சுட்டு மாறியில் பதிவு செய்யப்பட்டுள்ளது. y சுட்டு மாறி என்பதால், அது முகவரி மதிப்பை மட்டுமே ஏற்கும் என்பதறிவோம். "y, x- ஐச் சுட்டுகிறது" எனச் சொல்லலாம். இதனை இவ்வாறு உருவகிக்கலாம்:



x- ன் மதிப்பு 10 என்க. x- ன் மதிப்பை y என்னும் சுட்டு மாறியின் மூலமாகப் (ஏனெனில், y, x – ஐச் சுட்டுகிறது அல்லவா?) பெற முடியும். அதற்கு, "உள்நோக்கு" (indirection - *) செயற்குறியைப் பயன்படுத்த வேண்டும். அதாவது, *y என்பது, y- ஆல் சுட்டப்படும் முகவரியில் இருத்தி வைக்கப்பட்டுள்ள மதிப்பைப் பெற்றுத்தரும். எனவே, x- ன் மதிப்பு 10, சுட்டுமாறி y, x -ஐச் சுட்டுகிறது எனில், * y - ன் மதிப்பு 10 ஆகும்.

மேற்கண்ட எடுத்துக்காட்டில் நீங்கள் நினைவில் கொள்ள வேண்டியவை:

y, x என்ற மாறியின் முகவரியைக் கொண்டுள்ளது (&x)

*y, x என்ற மாறியின் மதிப்பைக் கொண்டுள்ளது (x)

முகவரி சுட்டு (address of - &), உள்நோக்கு (indirection -*) செயற்குறிகள் இரண்டுமே ஒருமச் செயற்குறிகள் என்பதை நினைவில் கொள்க.

4.2.4 செயற்குறிகள் (Operators)

சி-மொழி வளமான செயற்குறிகளைக் கொண்டுள்ளது. செயற்குறிகள் செயல்களை நிகழ்த்துகின்றன. "நிறைவேற்றப்பட வேண்டிய ஒரு செயல்பாட்டை (operaton) குறிக்கின்ற ஒரு குறியீடு (symbol)" என செயற்குறியை (operator) வரையறுக்கலாம். என்னென்ன பணிகளைச் செய்ய வேண்டும், அவற்றை எந்த வரிசையில் செய்யவேண்டும் என்பதைச்

செயற்குறிகள் கணிப்பொறிக்கு உணர்த்துகின்றன. செயல்பாடுகள் நிறை வேற்றப்படுகின்ற வரிசைமுறை 'முன்னுரிமை வரிசை' (order of precedence) என்று அழைக்கப்படுகின்றன. படிமரபு (hierarchy) என்றும் அழைக்கப்படும். செயல்பாடுகள் நிறைவேற்றப்படும் திசைமுகம் (இடமிருந்து வலம் அல்லது வலமிருந்து இடம்) தொடர்புறுத்தம் (associatirity), எனப் படுகிறது. சி-மொழியில் மூவகைச் செயற்குறிகள் உள்ளன:

- ஒருமச் செயற்குறிகள் (Unary operators)
- இருமச் செயற்குறிகள் (Binary operators)
- மும்மச் செயற்குறி (Ternary operators)

4.2.4.1 ஒருமச் செயற்குறிகள் (Unary operators)

ஒருமச் செயற்குறிகள் முன்னுரிமை வரிசையில் முதல்நிலையில் உள்ளன. இவை ஒரேயொரு செயலேற்பியைக் (operand) கொண்டுள்ளன. கணிப்பிடும் திசைமுகம் (associativity) வலமிருந்து இடமாகும். ஒருமச் செயற்குறிகளின் பட்டியலையும் அவற்றின் பணிகளையும் அட்டவணை 4.1-ல் காண்க:

அட்டவணை 4.1 ஒருமச் செயற்குறிகள்

குறியீடு	செயல்பாட்டின் வகை	திசைமுகம்
++	மிகுப்பு **(increment)**	
--	குறைப்பு **(decrement)**	வலமிருந்து இடம்
*	உள்நோக்கல்**(indirection)**	
&	முகவரிசுட்டல் **(address of)**	
!	எதிர்மறை **(logical NOT)**	

மிகுப்பு, குறைப்பு செயற்குறிகள் முறையே ஒரு மாறியின் தற்போதைய மதிப்பில் ஒன்று கூட்டவும், ஒன்று குறைக்கவும் பயன்படுத்தப் படுகின்றன. ஒருமச் செயற்குறிகள் பொதுவாக மாறிக்கு முன்பாக இடம் பெறுகின்றன. எடுத்துக்காட்டாக, x என்னும் int மாறியின் முகவரியை அறிய, &x எனப் பயன்படுத்துகிறோம். ஆனால், மிகுப்பு, குறைப்பு செயற் குறிகளைப் பொறுத்தவரை செயலேற்பியின் (மாறியின்) முன்னால் அல் லது பின்னால் இடம் பெறமுடியும்.

எனவே, மிகுப்பு, குறைப்பில் இரண்டு வடிவங்கள் உள்ளன:

- பின்னொட்டு (postfix) மிகுப்பு அல்லது குறைப்பு
- முன்னொட்டு (prefix) மிகுப்பு அல்லது குறைப்பு

++, -- ஆகிய ஒருமச் செயற்குறிகள் செயலேற்பியின் (மாறியின்) முன்னால் இடம்பெறுமாயின், முறையே 'முன்னொட்டு மிகுப்பு' அல்லது 'முன்னொட்டுக் குறைப்பு' என்று அழைக்கப்படுகின்றன. அவை செய லேற்பியை அடுத்து இடம்பெறுமாயின், முறையே 'பின்னொட்டு மிகுப்பு' அல்லது 'பின்னொட்டுக் குறைப்பு' என்று அழைக்கப்படுகின்றன. இவற் றின் பயன்பாடுகள்பற்றி இப்பாடத்தின் பிற்பகுதியில் படிப்போம்.

4.2.4.2 இருமச் செயற்குறிகள் (Binary Operators)

கணக்கீட்டுச் செயற்குறிகள் (Arithmatic operators)

+, -, *, /, % ஆகிய கணக்கீட்டுச் செயற்குறிகள் இரண்டு செய லேற் பிகள் மீது செயல்படும் என்பதால் இவை இருமச் செயற்குறிகள் என்று அழைக்கப்படுகின்றன. கணக்கீட்டுச் செயற்குறிகளின் பட்டியலை அட்டவணை 4.2-ல் காண்க. அனைத்து கணக்கீட்டுச் செயற்குறிகளின் திசைமுகம் இடமிருந்து வலம் ஆகும்.

அட்டவணை 4.2 கணக்கீட்டுச் செயற்குறிகள்

குறியீடு	செயல்பாட்டு வகை	திசைமுகம்
+	கூட்டல் (Addition)	
-	கழித்தல் (Subtraction)	
*	பெருக்கல் (Multiplication)	இடமிருந்து வலம்
/	வகுத்தல் (Division)	
%	வகுமீதி (Modulus)	

கணக்கீட்டுச் செயற்குறிகளை முழுஎண் (int) மற்றும் மிதவை (float) மதிப்புளுக்குப் பயன்படுத்த முடியும். வகுமீதி (modulus) செயற்குறி முழு எண் மதிப்புகளில் மட்டுமே செயல்படும். வகுத்தபின் வரும் மீதியை விடையாகத் தரும். முழுஎண் வகுத்தலின் ஈவு முழுஎண்ணாகவே இருக் கும். பின்னப் பகுதி புறக்கணிக்கப்படும்.

எடுத்துக்காட்டு:

5 / 2 = 2 (பின்னப் பகுதி புறக்கணிக்கப்பட்டுள்ளது)

5 % 2 = 1 (வகுத்தபின் வரும் மீதி)

முழுஎண்களைப் பொறுத்தவரை, வகுத்தல் செயற்குறி, வகுத்தபின் வரும் ஈவினை விடையாகத் தரும். வகுமீதி செயற்குறி, வகுத்தபின் வரும் மீதியை விடையாகத் தரும். வகுத்தல், பெருக்கல், வகுமீதி செயற் குறிகள், கூட்டல் மற்றும் கழித்தல் செயற்குறிகளைக் காட்டிலும் உயர் முன்னுரிமை பெற்றவை.

குறிப்பு: * குறியீடு, பெருக்கலுக்கான இருமச் செயற்குறியாக வும், சுட்டு மாறிகளின் ஒருமச் செயற்குறியாகவும் (உள் நோக்கல்-indirection) பயன்படுத்தப்படுகிறது. * குறியீடு பயன் படுத்தப்படும் சூழலைப் பொறுத்து அதன் செயல்பாடு அமை யும்.

4.2.4.3 ஒப்பீட்டுச் செயற்குறிகள் (Relational Operators)

ஒப்பீட்டு அல்லது பூலியன் செயற்குறிகள் எப்போதுமே இரண்டு செயலேற்பிகளின் மீது செயல்படுவதால் அவை இருமச் செயற்குறி களாகவே கருதப்படுகின்றன. ஒரேயொரு விதிவிலக்கு ! என்னும் பூலி யன் செயற்குறி (எதிர்மறைச் செயற்குறி என்றும் அழைக்கப்படும்) ஒருமச் செயற்குறி ஆகும். அதாவது, ஒரேயொரு செயலேற்பியின் மீதே செயல் படும். அட்டவணை 4.3-ல் ஒப்பீட்டுச் செயற்குறிகளின் பட்டியலைக் காண்க. அனைத்து ஒப்பீட்டுச் செயற்குறிகளின் திசைமுகம் இடமிருந்து வலமாகும். அதாவது, ஒன்றுக்கு மேற்பட்ட செயற்குறிகள் ஒரு கணக் கீட்டுக் கோவையில் இடம்பெற்றிருக்குமாயின் இடமிருந்து வலமாக ஒவ் வொன்றாகச் செயல்படுத்தப்படும்.

அட்டவணை 4.3 ஒப்பீட்டுச் செயற்குறிகள்

குறியீடு	செயல்பாட்டு வகை	திசைமுகம்
==	நிகரான (equla to)	
<	விடச் சிறிய (less than)	
>	விடப் பெரிய (greater than)	இடமிருந்து வலம்
<=	விடச் சிறிய அல்லது நிகரான (less than or equal to)	
>=	விடப் பெரிய அல்லது நிகரான (greater than or equal to)	
!=	நிகர் அல்லாத (not equal to)	

ஒப்பீட்டுச் செயற்குறிகள் இரண்டு மதிப்புகளை (உறுப்புகளை) ஒப் பிட்டு விடை அறியப் பயன்படுத்தப்படுகின்றன. விடை, சரி (True), தவறு (False) ஆகிய இரண்டில் ஒன்றாக இருக்கும். &&, || ஆகிய தருக்கச் செயற்குறிகள், இரண்டு அல்லது அதற்கு மேற்பட்ட ஒப்பீட்டுக் கோவை களை இணைக்கப் பயன்படுத்தப்படுகின்றன. && செயற்குறி, ''தருக்க உம்'' (logical AND) செயற்குறி ஆகும். இந்தச் செயற்குறி, தான் இணைக் கின்ற இரண்டு ஒப்பீட்டுக் கோவைகளுமே 'சரி' என விடை தருமாயின் இதுவும் 'சரி' என்ற விடையைத் தரும். இரண்டில் ஒன்று அல்லது இரண்டுமே 'தவறு' என விடை தருமாயின் இதுவும் 'தவறு' என விடை தரும். 'தருக்க அல்லது' (logical OR) செயற்குறி இதிலிருந்து மாறுபட்டது. தான் இணைக்கின்ற இரண்டு ஒப்பீட்டுக் கோவைகளுமே 'தவறு' என விடை தருமாயின் இதுவும் 'தவறு' என்ற விடை தரும். இரண்டில் ஒன்று அல்லது இரண்டுமே 'சரி' என்று விடை தருமாயின் இதுவும் 'சரி' என்ற விடைதரும். ஒப்பீட்டுச் செயற்குறிகள், கணக்கீட்டுச் செயற்குறிகளைக் காட்டிலும் குறைவான முன்னுரிமை கொண்டவை. &&, || ஆகியவை ஒப் பீட்டுச் செயற்குறிகளை விடவும் முன்னுரிமை குறைந்தவை.

எடுத்துக்காட்டு:

(10<15) && (14<23) என்னும் கோவை எப்போதுமே 'சரி' என்ற விடை யையே தரும். இந்தக் கோவை இடமிருந்து வலமாக மதிப்பிடப்படும். இரண்டு ஒப்பீட்டுக் கோவைகள், தருக்க && செயற்குறியால் இணைக்கப் பட்டுள்ளன. && செயற்குறி முழுக்கோவையையும் மதிப்பிடுவதற்கு முன் பாக, இரண்டு ஒப்பீட்டுக் கோவைகளும் அடுத்தடுத்து மதிப்பிடப்படும். இன்னோர் எடுத்துக்காட்டைப் பார்ப்போம்.

(10<15) || (14<23) என்னும் கோவையும் 'சரி' என்ற விடையையே தரும். தருக்க || செயற்குறி, இரண்டு ஒப்பீட்டுக் கோவைகளை இணைக் கின்றன. 'தருக்க அல்லது' விதியின்படி, இரண்டு ஒப்பீட்டுக் கோவை களுமே 'சரி' எனில், ஒட்டுமொத்தக் கோவையும் 'சரி' தான். இந்த எடுத் துக்காட்டில் முதல் ஒப்பீட்டுக் கோவை 'சரி' என்பதால், இரண்டாவது ஒப்பீட்டுக் கோவை மதிப்பிடப்படாது. இந்தக் கருத்துருவுக்குப் பெயர் 'குறுக்குச் சுற்று மதிப்பீடு' (Short Circuit Evaluation) ஆகும்.

'தருக்க உம்' (logical AND-&&) செயல்பாட்டில், இரண்டு கோவை களுள் முதல் கோவை 'தவறு' எனில், ஒட்டுமொத்தக் கோவையுமே 'தவறு' ஆகும். எனவே இரண்டாவது கோவை மதிப்பிடப்படாது.

4.2.4.4 மதிப்பிருத்து செயற்குறிகள் (Assignment Operators)

மதிப்பிருத்தும் செயற்குறி (=), வலப்பக்கமுள்ள செயலேற்பியின் மதிப்பை, இடப்பக்கச் செயலேற்பியில் இருத்தும். சி-மொழியில், +=, -=, *=, /=, %= ஆகிய கணக்கிட்டு-மதிப்பிருத்தும் செயற்குறிகளும் உள்ளன.

i = i + 1 ;

என்னும் கூற்றை நோக்குங்கள். இதில், i என்னும் மாறியின் முந்தைய மதிப்பில் ஒன்று மிகுக்கப்பட்டு, மிகுக்கப்பட்ட மதிப்பு, i- ன் புதிய மதிப்பாக இருத்தப்படுகிறது. இந்தக் கூற்றினை, கணக்கிட்டு-மதிப்பிருத்தும் செயற்குறியின் உதவியுடன்,

i += 1 ;

என்று எழுதலாம்.

மேற்கண்ட கூற்றில், i-ன் மதிப்பில் ஒன்று மிகுக்கப்பட்டு புதிய மதிப்பை i- யிலேயே இருத்துவதற்கு += செயற்குறி பயன்படுத்தப்பட்டுள்ளது. இதேபோல, i *= 2 ; என்னும் கூற்றில், i - ன் முந்தைய மதிப்பு 2-ஆல் பெருக்கப்பட்டு, புதிய மதிப்பு i-யிலேயே இருத்தப்படுகிறது.

அனைத்து செயற்குறிகளிலும், மதிப்பிருத்தும் செயற்குறிகளே மிகக் குறைந்த முன்னுரிமை கொண்டவை.

4.2.4.5 மதிப்பிடல் வரிசை (The order of evaluation)

செயற்குறிகளின் முன்னுரிமை பற்றி ஏற்கெனவே விவாதித்துள்ளோம். செயற்குறிகளின் முன்னுரிமை அடிப்படையில் ஒரு கோவையை எவ்வாறு மதிப்பிடுவது எனப் பார்ப்போம். எல்லாவற்றுக்கும் முன்பாக, பிறை அடைப்புக்குறிகளுக்குள் உள்ள கோவை மதிப்பிடப்பட வேண்டும்.

5 * 2 + 8 + 3 (3 - 2) * 5

என்னும் கோவையை எடுத்துக்கொள்வோம். இது, கீழே உள்ளவாறு மதிப்பிடப்படும்:

5 * 2 + 8 + 1 * 5

(அடைப்புக்குறிகளுக்குள் இருந்த கோவை முதலில் மதிப்பிடப்பட்டுள்ளது)

10 + 8 + 5 (பெருக்கல், கூட்டலைவிட முன்னுரிமை பெறுகிறது)

23 (கூட்டல் முடிந்தபின் கோவையின் மதிப்பு 23)

4.2.4.6 மும்மச் செயற்குறி (Ternary Operator)

சி-மொழியில் ஒரு மும்மச் செயற்குறி உள்ளது. இதுவொரு நிபந்தனைச் செயற்குறி (Conditional Operator) ஆகும். இச் செயற்குறிக்குப் பயன்படும் குறியீடு ? : ஆகும். இது மூன்று செயலேற்பிகளைக் கொண்டிருக்கும். நிபந்தனைச் செயற்குறியின் கட்டளை அமைப்பு:

(நிபந்தனைக் கோவை) ? கோவை1 : கோவை2 ;

நிபந்தனைக் கோவை 'சரி' எனில், கோவை1 மதிப்பிடப்படும். நிபந்தனைக் கோவை 'தவறு' எனில், கோவை2 மதிப்பிடப்படும்.

எடுத்துக்காட்டு:

j = (i < 0) ? -i : i;

என்ற கட்டளையில் நிபந்தனைச் செயற்குறியின் பயன்பாட்டை நோக்குங்கள். i-ன் முற்று மதிப்பை (absolute value) j- யில் இருத்துவதே இதன் நோக்கம். i- ன் மதிப்பு 0-ஐவிடச் சிறிதெனில், j-யில் -i இருத்தப்படும். i-யின் மதிப்பு 0-வுக்கு நிகர் அல்லது 0-ஐவிடப் பெரிதெனில் j- ல் i-இருத்தப்படும்.

4.2.5 நிறுத்தற்குறிகளும் சிறப்புச் சொற்களும் (Punctuations and Keywords):

சி-மொழியில் பயன்படுத்தப்படும் நிறுத்தற் குறிகளும் அவற்றின் பயன்களும் பட்டியலிடப்படுகின்றன:

[ ] - அணியில் சுட்டு எண்ணை [index] குறிக்கப் பயன்படுகிறது [சதுரஅடைப்புக்குறி].

{ } - செயல்கூறின் உடற்பகுதிக்கு வரம்பிடப் பயன்படுகிறது {நெளிவு அடைப்புக்குறி}.

( ) - செயல்கூறை குறிப்பிடவும், உறுப்புகளைக் குழுவாக்கவும், கோவைகளைக் குழுவாக்கவும் பயன்படுகிறது (சாதாரண அடைப்புக்குறிகள்).

< > - முன்-செயலிக் (preprocessor) கூற்றில், தலைப்புக் கோப்பின் பெயரை உட்படுத்தப் பயன்படுகிறது <கோண அடைப்புக்குறிகள்>

" " - சர நிலையுருக்களைக் குறிப்பிடப் பயன்படுகிறது (இரட்டை மேற்கோள்).

' ' - ஒற்றைக் குறியுரு மாறிலிகளைக் குறிப்பிடப் பயன்படுகிறது (ஒற்றை மேற்கோள்).

/* */- குறிப்புரை (Comment) அமைப்பதற்கு.

; - ஒரு கூற்றின் முடிவுக்குறி (அரைப்புள்ளி)

, - உறுப்புகளைப் பிரித்துக்காட்டப் பயன்படுகிறது (காற்புள்ளி)

வெற்று, வெண் இடவெளிகள் - நிரலின் படிப்பெளிமையை (read ability) மேம்படுத்தப் (Blank, white spaces) பயன்படுகின்றன.

முக்கிய சிறப்புச் சொற்களின் பட்டியல் கீழே தரப்பட்டுள்ளது. ஒரு நிரலில் மாறிகளின் பெயர்களாக இவற்றைப் பயன்படுத்தக் கூடாது.

auto	**break**	**case**	**char**	**continue**	**default**	**do**
else	**if**	**float**	**for**	**int**	**return**	**static**
switch	**while**					

சிறப்புச் சொற்கள், பட்டியலில் எவ்வாறு உள்ளனவோ அவ்வாறே துல்லியமாகத் தரப்படவேண்டும். எடுத்துக்காட்டாக, auto என்பதுதான் சிறப்புச் சொல். Auto, AUTO ஆகியவை சிறப்புச் சொற்கள் அல்ல.

4.3 மாதிரி சி–நிரல்

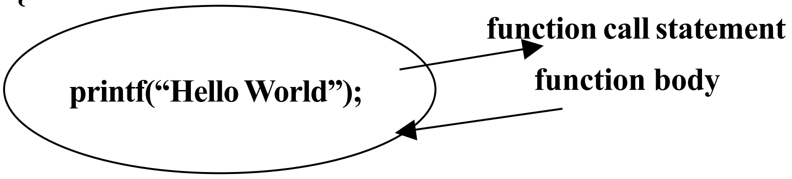
விரும்புகின்ற விடையைப் பெறுவதற்காக, தொடர்ச்சியாகச் செயல் படுத்த வேண்டிய ஆணைகளின் தொகுதியே 'நிரல்' (Program) எனப்படு கிறது. ஒரு தொகுதியே பெரிய நிரலின் சிறிய பணியைச் செய்து முடிப் பதற்காகப் பயன்படுத்தப்படுகிற ஒரு நிரலே 'செயல்கூறு' (Function) ஆகும். சி- நிரல்கள் மிகச் சிறியவையாகவும் இருக்கலாம். சி-நிரல்கள் செயல் கூறுகளால் ஆனவை. செயல்கூறு இல்லாமல் சி-மொழியில் ஒரு நிரலை எழுத முடியாது. செயல்கூறு, முன்பே வரையறுக்கப்பட்டு சி-மொழியில் உள்ளிணைக்கப்பட்டதாக இருக்கலாம் அல்லது பயனர் தாமே வரை யறுத்தாக இருக்கலாம். ஒரு சி - நிரலின் குறிமுறையைக் கீழே காண்க:

```
#include <stdio.h>
main()
{
  printf("Hello World");
}
```

சி–மொழியில் ஒரு மாதிரி நிரல்

இந்த நிரலை இயக்கினால், Hello world என்னும் செய்தியைத் திரையில் காட்டும். சி-நிரலில், main() என வரையறுக்கப்படும் ஒரு செயல்கூறு கட்டாயமாக இருக்க வேண்டும். main() - பயனர் வரையறுக்கும் செயல் கூறாகும். இதற்கான கட்டளைத் தொகுதியைப் பயனர் எழுதவேண்டும். ஒரு சி-நிரலை இயக்கும்போது, கட்டுப்பாடு main() செயலுகூறுக்கு மாற்றப்படுகிறது. இந்தச் செயல்கூறு, நிரலின் நுழைவு வாயில் எனப்படுகிறது. மேலேயுள்ள நிரலில் இரண்டு செயல்கூறுகள் உள்ளன. அதில் ஒன்று, நிரலின் நுழைவு வாயிலாக விளங்கும் main( ) - பயனர் வரையறுத்தது. இன்னொன்று, printf() - முன் வரையறுக்கப்பட்டது. அடிப்படை வெளியீட்டில் (திரை அல்லது திரையகம்) நிரலின் விடையைக் காட்டுவதற்குப் பயன்படுத்தப்பட்டுள்ளது. செயல்கூறு என்பதை உணர்த்திட பிறை அடைப்புக்குறிகள் ( ) பயன்படுத்தப்பட்டுள்ளன. இரண்டு செயல் கூறுகளுள் main( ), அழைக்கும் செயல்கூறாகும். printf() அழைக்கப்படும் செயல்கூறாகும். Hello world நிரல் தேவையான விளக்கங்களுடன் கீழே தரப்பட்டுள்ளது.

```
#include <stdio.h>    /* <=  preprocessor statement */
main()                /* <= function header statement */
/* function definition starts  <= comment statement */
{
```




```
}
```

நிரலின் முதல்வரி #include <stdio.h> என்பது முன்-செயலிக் (Preprocessor) கூற்றாகும். #include என்பது முன்-செயலி நெறியுறுத்தம் ஆகும். முன் -செயலி என்பது, நிரலை மொழி பெயர்க்கும் போது (compile time) மூலக் குறிமுறையை விரித்தெழுதும் ஒரு மென்பொருள் நிரலாகும். # include <stdio.h> என்னும் கூற்று, stdio.h (standard input and output header file) என்னும் கோப்பின் உள்ளடக்கத்தை நிரலில் main() செயல்கூறின் முன்பாகச் சேர்த்துக் கொள்கிறது. printf(), scanf() போன்ற முன்-வரையறுக்கப்பட்ட உள்ளீட்டு, வெளியீட்டு செயல்கூறுகளின் அறிவிப்புக் கூற்றுகளே stdio.h கோப்பில் இடம் பெற்றுள்ளன.

printf() , scanf() ஆகிய செயல்கூறுகளின் அறிவிப்புகள்,

int printf(char *, ...);

int scanf (char *, ...) ;

என்று அமைந்திருக்கும்.

மேற்கண்ட செயல்கூறுகள் ஒவ்வொன்றும் வெவ்வேறு எண்ணிக்கையில் அளபுருக்களை (variable numbers of parameters) ஏற்க முடியும் என்பதை முப்புள்ளிகள்(...) உணர்த்துகின்றன. முதலாவதாக இடம்பெறும் அளபுரு ஒரு சரமாகவே (string) இருக்கும். அளபுரு என்பது, அழைக்கப்படும் செயல்கூறினுக்கு வழங்கப்படும் தரவு அல்லது தகவலைக் குறிக்கும். பொதுவாக, செயல்கூறுகளுக்கு இத்தகைய அளபுருக்கள் சில அனுப்பப்படலாம் அல்லது அனுப்பப்படாமலும் இருக்கலாம். ஒன்றுக்கு மேற்பட்ட அளபுருக்கள் இடம்பெறும்போது அவை, செயல்கூறின் பெயரை அடுத்து, அடைப்புக்குறிகளுக்குள் காற்புள்ளியிட்டு ஒன்றன்பின் ஒன்றாகக் குறிப்பிடப்படுகின்றன. மேலேயுள்ள நிரலில் printf( ) செயல்கூறில் ஒரேயொரு அளபுரு மட்டுமே இடம்பெற்றுள்ளது. அது, திரையகத்தில் காட்டப்படவேண்டிய ஒரு சரமாகும். முதல் அளபுருவாக இடம்பெற்றுள்ள char * என்னும் (சர இன) அளபுரு, "குறியுருச் சுட்டு" (Character pointer) என்று அழைக்கப்படுகிறது. இதைப்பற்றிப் பிறகு படிப்போம்.

சி-மொழி நிரல்பெயர்ப்பி (Compiler), முன்-வரையறுக்கப்பட்ட செயல்கூறுகளை அடையாளம் கண்டுகொள்ளும். காரணம், நிரலில் நாம் சேர்த்துக்கொண்டுள்ள தலைப்புக் கோப்புகளில் (Header Files) அவை முறைப்படி அறிவிக்கப்பட்டுள்ளன. தலைப்புக் கோப்புகளின் பயன்பாட்டினைக் கீழேயுள்ள நிரலில் காண்க:

```
#include <stdio.h>
#include <conio.h>
main()
{
clrscr();
printf("hello");
}
```

இந்த நிரலில் பயன்படுத்தப்பட்டுள்ள clrscr() என்னும் முன்-வரைய றுக்கப்பட்ட செயல்கூறின் மாதிரி வடிவம் (அறிவிப்பு) conio.h என்னும் தலைப்புக் கோப்பில் இடம்பெற்றுள்ளது. எனவே, அக்கோப்பினை இந்த நிரலில் இணைத்துக்கொண்டுள்ளோம். # include <conio.h> என்னும் கூற்று இந்த நிரலில் இணைக்கப்படவில்லையெனில், சி-மொழி நிரல் பெயர்ப்பி (Compiler), clrscr() செயல்கூறினை நிரலரே வரையறுத்துள்ளாரா எனப் பார்க்கும். அப்படி இல்லையெனில், பிழை என அறிவிக்கும்.

4.3.1 கூற்றுகள் (Statements)

சி - நிரலின் ஒவ்வொரு வரிகளும் ஒரு கூற்றாகக் (Statement) கருதப் படுகின்றன. பொதுவாக, நான்கு வகையான கூற்றுகள் உள்ளன. அவை

- முன்-செயலிக் கூற்றுகள் (Preprocessor Statements)
- செயல்கூறு தலைப்புக் கூற்றுகள் (Function Header Statements)
- அறிவிப்புக் கூற்றுகள் (Declaration Statements)
- செயல்பாட்டுக் கூற்றுகள் (Executable Statements)

நீங்கள் ஏற்கெனவே அறிந்தபடி, முன் -செயலிக்கூற்று, குறிப்பிட்ட தலைப்புக் கோப்புகளிலிருந்து செயல்கூறுகளின் அறிவிப்புக் கூற்றுகளை எடுத்துச் சேர்த்து, மூல நிரலை விரிவுபடுத்தும் பணியைச் செய்கின்றன. செயல்கூறு தலைப்புக் கூற்று, ஒரு செயல்கூறின் வரையறையில் முதல் வரியாக இடம்பெறுகிறது. அறிவிப்புக் கூற்றுகளை, மாறி அறிவிப்புக் கூற்று (variable declaration statement), செயல்கூறு அறிவிப்புக் கூற்று (function declaration statement) என மேலும் இருவகையாகப் பிரிக்கலாம்.

```
#include <stdio.h>        => முன்-செயலிக் கூற்று
main()                    => செயல்கூறு தலைப்புக் கூற்று
{
    int a,b,c;            => மாறி அறிவிப்புக் கூற்று
    int add(int,int);     => செயல்கூறு அறிவிப்புக் கூற்று
    a = 10;               => செயல்பாட்டுக் கூற்று
}
```

பயனர் வரையறுக்கும் செயல்கூறுகளின் அறிவிப்பு மற்றும் வரை யறை பற்றி “செயல்கூறுகள்” என்னும் பகுதியில் விளக்கப்பட்டுள்ளது. செயல்பாட்டுக் கூற்றுகளில் பல வடிவங்கள் உள்ளன. அவற்றுள் மதிப் பிருத்து கூற்று (assignment statement) அடிப்படையான ஒன்று. ஒரு மாறி யில் மதிப்பை இருத்த அது பயன்படுகிறது.

4.3.1.1 மதிப்பிருத்து கூற்றுகள் (Assignment Statements)

மதிப்பிருத்து கூற்று இவ்வாறு அமையும்:

மாறி **(Variable) =** கோவை **(Expression) ;**

ஓர் அரைப்புள்ளி கூற்றை முடித்துவைக்கிறது. கோவை என்பது பல வகைப்படும். அதுபற்றிப் பிறகு பார்ப்போம். ஒரு கோவை மதிப் பிடப்படும்போது, அது ஒரு விடையைத் தரும். அதாவது, அது எப் போதுமே ஒற்றை மதிப்பாகச் சுருக்கப்பட்டுவிடும். கோவையில் அந்த ஒற்றை மதிப்பு, இடப்பக்கமுள்ள மாறியில் இருத்தப்படுகிறது. = என்னும் குறி, மதிப்பிருத்து செயற்குறியாகும். ஒரு மாறியில் மதிப்பை இருத்து வதற்குப் பயன்படுகிறது. மதிப்பிருத்து செயற்குறிக்கு (=) வலப்பக்கம் இடம்பெற்றுள்ள அனைத்தும் ஒரு கோவையாகக் கருதப்படுகின்றன.

ஏற்கெனவே நாம் அறிந்தபடி, ஒரு செயல்கூறில் இடம்பெற்றுள்ள அனைத்துக் கூற்றுகளும் நெளிவு அடைப்புக்குறிகளுக்குள் அமைக்கப் படுகின்றன.

printf (“hello”) ;

என்பது ஒரு செயல்கூறு அழைப்புக் கூற்று (function call statement) ஆகும். இந்தச் செயல்கூறு இயக்கப்படும்போது, hello என்னும் செய்தி திரையில் காட்டப்படும். திரையில் எத்தனை எழுத்துகள் காட்டப்பட்டன என்கிற எண்ணிக்கையை செயல்கூறு திருப்பி அனுப்பும். printf() செயல்கூறின் வரையறையை நோக்கும்போதே, அது ஒரு முழுஎண் மதிப்பைத் திருப் பியனுப்பும் என்பதை நாம் புரிந்துகொள்ள முடியும். எடுத்துக்காட்டாக, ஒரு நிரலின் பகுதியைக் காணுங்கள்:

int n ; /* மாறி அறிவிப்புக் கூற்று ***/**

n = printf (‘hello”) ; */ மதிப்பிருத்து கூற்று */

இங்கே, n என்னும் மாறி ஒரு முழுஎண் (integer) என அறிவிக்கப் பட்டுள்ளது. மதிப்பிருத்து கூற்று செயல்படும்போது, முதலில் வலப்பக்க முள்ள கோவை மதிப்பிடப்படும். பிறகு, அதன் மதிப்பு இடப்பக்க மாறி யில் இருத்தப்படும். இந்த எடுத்துக்காட்டில், மதிப்பிருத்து கூற்றின் வலப்பக்கம், ஒரு செயல்கூறு அழைப்புக் கோவை (function call expression) இடம்பெற்றுள்ளது. எனவே, இப்போது printf() என்னும் செயல்கூறு இயக்கப்பட்டு, hello என்னும் செய்தி திரையில் காட்டப்படும். printf() செயல் கூறு, 5 எழுத்துகளைத் திரையில் காட்டுவதால் 5 என்னும் மதிப்பைத் திருப்பியனுப்பும். அதுவே வலப்பக்கக் கோவையின் மதிப்பாகும். 5 என்னும் இம்மதிப்பு இடப்பக்கமுள்ள n என்னும் மாறியில் இருத்தப்படும்.

4.3.1.2 மிகுப்பு, குறைப்புக் கூற்றுகள்
(Increment and Decrement Statements)

ஒரு மாறியின் முந்தைய மதிப்பில் 1 கூட்டிப் புதிய மதிப்பை அதே மாறியில் இருத்திவைக்க,

i = i + 1 ;

என்னும் கூற்று பயன்படுகிறது.

இதே பணியை, மிகுப்புக் கூற்று (Increment Statement) மூலமாகவும் செய்து முடிக்கலாம். இவ்வாறாக:

i ++ ; /* பின்னொட்டு வடிவம் */

+ + i ; / * முன்னொட்டு வடிவம் */

மேலே காட்டியவாறு, மிகுப்புச் செயற்குறியைத் தனிநிலைக் கூற்றுகளில் (stand-alone statements) பயன்படுத்தினால், முன்னொட்டுக்கும் பின்னொட்டுக்கும் வேறுபாடு எதுவுமில்லை. இரண்டிலுமே i- ன் மதிப்பு ஒன்று மிகுக்கப்படும். i-ன் முந்தைய மதிப்பில் ஒன்றைக் குறைக்க, குறைப்புச் செயற்குறியைப் பயன்படுத்தவேண்டும். i -- ; அல்லது -- i; மிகுப்பு, குறைப்புச் செயற்குறிகள் வேறு சூழ்நிலைமைகளில் வேறுவித விளைவுகளைத் தரும். அதுபற்றிப் பிறகு பார்ப்போம்.

4.3.1.3 கோவை (Expression)

கோவை என்பது பெரும்பாலும், ஒரு மதிப்பிருத்து செயற்குறிக்கு வலப்பக்கமாக அமைவதுண்டு. அதை மதிப்பிடும்போது, ஒரு 'மதிப்பு' விடையாகக் கிடைக்கும். கோவைகள் பல்வேறு வடிவம் கொண்டவை. அவற்றுள் சில:

int a,b,c; /* மாறி அறிவிப்புக் கூற்று */

a =10 ; /* மதிப்பிருத்து கூற்று */

இரண்டாவது கூற்றில், 10 என்னும் மாறா மதிப்பு வலப்பக்கம் இடம்பெற்றுள்ளது. எனவே இது 'மாறிலிக் கோவை' (constant expression) எனப்படுகிறது. அதன் மதிப்பு 10.

b= a ;

இதில், வலப்பக்கம் ஒரு 'மாறிக் கோவை' (variable expression) இடம் பெற்றுள்ளது. அதன் மதிப்பு 10. மதிப்பிருத்து கூற்றின் வலப்பக்கம் இரு வகையான மதிப்புகள் இடம்பெற முடியும். ஒரு மாறியின் மதிப்பு அல்

லது ஒரு கோவையின் மதிப்பு. இந்த எடுத்துக்காட்டில் இரண்டும் ஒன்றே. ஆனாலும், கோவையின் மதிப்பே இடப்பக்க மாறியில் இருத்தப்படுகி றது என்பதை எப்போதும் நினைவில் கொள்க.

பயன்படுத்தப்படும் செயற்குறிகளின் அடிப்படையில் கோவைகள் அழைக்கப்படுவதுண்டு. வேறுசில கோவைகள் கீழே:

கோவை	வலப்பக்கம்
c = a + b ;	கணக்கீட்டுக் கோவை
c = a > b ;	ஒப்பீட்டுக் கோவை
f = d = e ;	மதிப்பிருத்து கோவை

ஒப்பீட்டுக் கோவையைப் பொறுத்தவரை, 'சரி' (true) அல்லது 'தவறு' (false) என்னும் மதிப்பு. இடப்பக்க மாறியில் இருத்தப்படும். அதாவது 1 அல்லது 0 என்னும் மதிப்பு c என்னும் மாறியில் இருத்தப்படும். = என்னும் குறி, மதிப்பிருத்து செயற்குறியாகப் பயன்படுத்தப்படுகிறது. எனவே, சி-மொழியில் மதிப்பிருத்து கோவை ஏற்கப்படுகிறது என்று பொருள். தொடர் மதிப்பிருத்தலும் அனுமதிக்கப்படுகிறது. மூன்றாவது கூற்றில், மதிப்பிருத்து கோவையின் மதிப்பு (அதாவது, d- ல் இருத்தப் படும் மதிப்பே, மதிப்பிருத்து கோவையின் மதிப்பாகும்) f என்னும் மாறியில் இருத்தப்படுகிறது.

f = d =10;

என்றும் கூற்றை எடுத்துக்கொள்வோம். இதில் மதிப்பிருந்து கோவை யின் மதிப்பு 10 ஆகும். இம்மதிப்பு d-ல் இருத்தப்படுகிறது. பிறகு, மதிப்பிருந்து கோவையின் இந்த மதிப்பு f-ல் இருத்தப்படுகிறது.

4.3.1.4 பின்னொட்டு, முன்னொட்டு மிகுப்புக் கோவைகள் (Postfix and Prefix Increment Expressions)

கீழேயுள்ள நிரல் பகுதியை நோக்குக:

```
int x, i;
i = 10;
x = i++;  /* வலப்பக்கம் பின்னொட்டு மிகுப்புக்கோவை*/
printf("%d %d\n", x, i);
```

ஒருமச் செயற்குறி ++ மீவுயர் முன்னுரிமை பெற்றது. மதிப்பிருத்து செயற்குறி மீக்குறைந்த முன்னுரிமை பெற்றது. பின்னொட்டு வடிவில், முதலில் மாறியின் மதிப்பு, கோவையின் மதிப்பாகப் பயன்படுத்தப்படு

கிறது. அதன்பிறகே, மாறியின் மதிப்பு, ஒன்று மிகுக்கப்படுகிறது. எனவே, வலப்பக்கத்தில் இரண்டு வெவ்வேறு மதிப்புகள் உள்ளன. ஒன்று, கோவையின் மதிப்பு; மற்றது மாறியின் மதிப்பு. மேலேயுள்ள மதிப்பிருத்து கூற்றில் கோவையின் மதிப்பு 10. அதுவே இடப்பக்க மாறி x-ல் இருத்தப்படுகிறது. இப்போது i-ன் மதிப்பு 11 ஆகிறது. ஒரு மதிப்பிருத்து கூற்றில் இடப்பக்க மாறியில் ஒரு மதிப்பை இருத்த முனையும்போது, வலப்பக்கக்கோவையின் மதிப்பையே எப்போதும் கணக்கில் எடுத்துக்கொள்ள வேண்டும். எனவே, printf() செயல்கூறின் வெளியீடு,

10 11

என்று அமையும். கீழேயுள்ள நிரல் பகுதியை நோக்குக:

```
int x, i;
i = 10;
x = ++i;          /*வலப்பக்கம் முன்னொட்டு மிகுப்புக்கோவை*/
printf("%d %d\n", x, i);
```

முன்னொட்டு வடிவில், முதலில் மாறியின் மதிப்பு, ஒன்று மிகுக்கப்பட்டு, புதிய மதிப்பு, கோவையின் மதிப்பாகப் பயன்படுத்தப்படுகிறது. கோவையின் மதிப்பு 11 என்பதால் அதுவே இடப்பக்க மாறி x-ல் இருத்தப்படுகிறது. எனவே வெளியீடு,

11 11

என அமையும். ஒப்பீட்டுக் கோவை இடம்பெற்றுள்ள எடுத்துக்காட்டைக் கீழே காண்க:

```
nt x, z;
x = 100;
z = (x == x++);       /* பின்னொட்டு மிகுப்புக் கோவையை ஒரு
                செயலேற்பியாகக் கொண்ட ஒப்பீட்டுக் கோவை*/
printf("%d %d\n", z, x);
```

இதன் வெளியீடு என்னவாக இருக்கும்?

மேற்கண்ட மதிப்பிருத்து கூற்றில் இடம்பெற்றுள்ள ஒப்பீட்டுக் கோவை, சரி அல்லது தவறு அதாவது, 1 அல்லது 0 என்னும் மதிப்பைத் தரும். எனவே, z என்னும் மாறியின் மதிப்பு 1 அல்லது 0 ஆக இருக்கலாம். ++ செயற்குறியின் மீவுயர் முன்னுரிமை காரணமாக, ஒப்பீட்டுக் கோவையின் வலப்பக்க செயலேற்பியே முதலில் மதிப்பிடப்படும். அது பின்னொட்டுக் கோவை என்பதால், கோவையின் மதிப்பு

100 ஆகும். அதன்பிறகு x-ன் மதிப்பு மிகுக்கப்பட்டு அதன் மதிப்பு 101 ஆகிறது. இப்போது ஒப்பீட்டுக் கோவையின் இடப்பக்கச் செயலேற்பி (ஒரு மாறிக் கோவை) 101 என்னும் மதிப்பைப் பெறும். எனவே, ஒப் பிடவேண்டிய மதிப்புகள் 101 மற்றும் 100 ஆகும். அவையிரண்டும் நிக ரானவை அல்ல. எனவே, z-ல் 0 இருத்தப்படும். இதன் வெளியீடு,

0 101

என அமையும்.

நினைவில் கொள்க: மதிப்பிருத்து கூற்றில், கோவையின் மதிப் புக்கே முக்கியத்துவம் தரவேண்டும். மாறியின் மதிப்புக்கு அல்ல.

4.3.1.5 உள்ளீட்டு, வெளியீட்டுக் கூற்றுகள்:
(Input and Output Statements)

நாம் ஏற்கெனவே அறிந்தபடி, விடையை அடிப்படை வெளியீட்டில் (திரையில்) காண்பிக்க printf() செயல்கூறு பயன்படுகிறது. திரையகத்தில் ஒரு சரத்தைக் காட்டுவதற்கு printf() பயன்பட்டதை ஏற்கெனவே நாம் பார்த்தோம். உண்மையில், printf() செயல்கூறின் முதல் அளபுரு ஒரு சரம் ஆகும். வெளியீட்டைக் கட்டுப்படுத்த (தேவைக்கேற்ப அமைத் துக்கொள்ள) அது பயன்படுகிறது. எனவே அது 'கட்டுப்பாட்டுச் சரம்' (control string) என்று அழைக்கப்படுகிறது. இந்த அளபுரு, வெளியீட்டைத் திரையில் காட்டுவதற்கு ஏற்றவாறு வடிவமைக்கப்படுவதால் இதனை 'வடிவமைப்புச் சரம்' (formatting string) என்றும் அழைக்கலாம்.

எடுத்துக்காட்டு:

ஒரு முழுஎண் மதிப்பை வெளியிடல்:

```
int n ;  /*  முழு எண் மாறியாக n அறிவிக்கப்படுகிறது */
n =  10;
printf (“%d” ,n);
```

இந்த எடுத்துக்காட்டில், “%d” என்னும் 'வடிவமைப்புக் குறியுரு' (formatting character) printf() செயல்கூறின் கட்டுப்பாட்டுச் சரத்துள்ஒரு முழு எண் மதிப்பை வெளியிட, பயன்படுத்தப்பட்டுள்ளது. printf() செயல்கூறின் கட்டுப்பாட்டுச் சரம், மூன்று வகையான குறியுருக் களை ஏற்றுக்கொள்ளும்.

- சாதாரண குறியுருக்கள் (Ordinary Characters)
- வடிவமைப்புக் குறியுருக்கள் (Formatting Characters)
- விடுபடு வரிசைக் குறியுருக்கள் (Escape Sequence Characters)

கட்டுப்பாட்டுச் சரத்திலுள்ள சாதாரணக் குறியுருக்கள் அப் படியே உள்ளவாறே காட்டப்படும். printf (“hello”); என்னும் கூற்றில் கட்டுப் பாட்டுச் சரம் சாதாரணக் குறியுருக்களையே கொண்டுள்ளது. எனவே, அவை உள்ளவாறே திரையில் காட்டப்படுகின்றன. பல்வேறு தரவின மதிப்புகளை திரையில் காட்டப் பயன்படும் வடிவமைப்புக் குறியுருக்க ளின் பட்டியலை அட்டவணை 4.4-ல் காண்க:

வடிவமைப்புக் குறியுரு	தரவினம்
%d	int
%f	float
%c	char
%s	char [ ]
%ld	long int
%lf	long float or double

அட்டவணை 4.4 வடிவமைப்பு வரையறுப்புகள்

நாம் ஏற்கெனவே அறிந்தபடி, விடுபடு வரிசைக் குறியுருக்கள் என்பவை, ஒரு பின்சாய்வுக் கோட்டை அடுத்து ஓர் எழுத்தைக் கொண் டிருக்கும். உண்மையில் அவை ஒற்றைக் குறியுரு மாறிலிகளே ஆகும். அவை ஒற்றைக் குறியுருக்களாகவே சேமிக்கப்படுகின்றன, கையாளப் படுகின்றன. வெளியீட்டின் வடிவமைப்பை ஓரளவு கட்டுப்படுத்த விடு படு வரிசைகள் பயன்படுகின்றன. printf() செயல்கூறின் கட்டுப்பாட்'ரீச் சரத்தில் அடிக்கடி பயன்படுத்தப்படுகின்ற விடுபடு வரிசைகள் சில:

‘\n’ - புதிய வரிக் குறியுரு

‘\t’ - தத்தல் குறியுரு

‘\b’ - பின் இடவெளிக் குறியுரு

கீழேயுள்ள printf() செயல்கூறு அழைப்புக் கூற்றை நோக்குக:

int i =15;

printf (‘the value of i = &%d\n”, i);

திரையில் காட்டப்படும் வெளியீடு:

the value of i = 15

மேற்கண்ட printf() செயல்கூறில், சாதாரணக் குறியுருக்கள், வடிவ மைப்புக் குறியுருக்கள், விடுபடு வரிசைக் குறியுருக்கள் ஆகிய மூன்று வகை குறியுருக்களுமே பயன்படுத்தப்பட்டுள்ளன. சாதாரணக் குறியுருக் கள் உள்ளவாறே காட்டப்பட்டுள்ளன. %d என்னும் வடிவமைப்புக் குறி யுருவின் காரணமாக அடுத்துள்ள முழு எண் மதிப்பு திரையில் காட்டப் பட்டுள்ளது. புதிய வரிக் குறியுரு '\n' இருப்பதன் காரணமாய், வெளியீடு காட்டப்பட்ட வரிக்கு அடுத்த வரியில், காட்டி வந்து நிற்கிறது. அடுத்து வரும் வெளியீடுகள் புதிய வரியில் காட்டப்படுகின்றன.

```
printf ("one\n two\n three\n") ;
```

என்னும் கூற்று,

```
one
two
three
```

எனத் திரையில் காட்டும்.

புதியவரிக் குறியுரு \n இடம் பெற்றுள்ள காரணத்தால் ஒவ்வொரு சொல்லும் தனி வரியில் காட்டப்படுகிறது.

கீழே வரும் நிரலின் பகுதியை நோக்குக:

```
int x;
float y;
x = 10;
y = 10.5;
printf("%d %f", x, y);
```

இதன் வெளியீடு,

```
10        10.500000
```

என இருக்கும்.

மிதவைப் புள்ளி மதிப்புகள் முன்னியல்பாக (by default) ஆறு இலக் கத் துல்லியத்தில் காட்டப்படும். இரண்டு இலக்கத் துல்லியத்தில் வெளி யிட விரும்பினால், வடிவமைப்பு வரையறுப்பு % 0.2f என்று அமைய வேண்டும். இதில், பதின்மப் பின்னம் இரண்டு இலக்கத் துல்லியமாகக் கட்டுப்படுத்தப்படுகிறது. எனவே,

printf (" %0.2f ", y);

என்னும் கூற்றின் வெளியீடு 10.50 என அமையும். ஆக, வெளியீட் டைத் திரையில் காட்டும்போது அதன் அகலத்தைக் கட்டுப்படுத்த, வடி வமைப்பு வரையறுப்புகளில் அகலத்தையும் உடன் சேர்த்துக்கொள்ள முடியும்.

printf (" %10d %10.2f ", x, y);

என்னும் கூற்று,

bbbbbbbb10bbbbb10.50 /* b என்பது வெற்று இடவெளியைக் குறிக்கிறது */

விசைப்பலகையிலிருந்து உள்ளீடு (Input from keyboard)

விசைப் பலகையிலிருந்து (அடிப்படை உள்ளீடு) மதிப்புகளை உள் ளீடாகப் பெற scanf() செயல்கூறு பயன்படுகிறது. scanf()- ன் மாதிரி வடிவம் (prototype), printf()-ன் மாதிரி வடிவத்தை ஒத்ததாகும். இதுவும் வேறுபட்ட எண்ணிக்கையில் அளபுருக்களை ஏற்கும்.

ஒரு முழுஎண் மாறி x-ன் மதிப்பை விசைப்பலகை மூலம் பெற,

```
int  x ;
scanf ("%d",  &x);
```

என்னும் குறிமுறை (code) பயன்படுத்தப்படுகிறது.

scanf() செயல்கூறு இயக்கப்படும்போது, பயனரின் உள்ளீட்டுக்காகக் கணிப்பொறி காத்திருக்கும். பயனர் விசைப்பலகை மூலமாகத் தரவினை உள்ளிடவேண்டும். பயனர் Enter விசையை அழுத்திய பிறகே, உள் ளிட்ட தரவு நினைவகத்தில் x-க்குரிய இடத்தில் இருத்தப்படுகிறது. scanf() செயல்கூறின் இரண்டாவது அளபுருவாக &x இடம்பெற்றுள்ளது. இது x என்னும் மாறியின் நினைவக முகவரியைக் குறிக்கிறது. & என்பது முக வரி சுட்டும் செயற்குறி என்பதை அறிவோம். அதனை ஒரு மாறியுடன் சேர்த்துப் பயன்படுத்தும்போது, அந்த மாறியின் முகவரியைக் குறிக்கிறது.

4.3.2 பயனர் வரையறுக்கும் செயல்கூறுகள் (User -defined Functions)

ஒரு நிரலில் ஒரு குறிப்பிட்ட பணியை மீண்டும் மீண்டும் செய்ய வேண்டியிருந்தாலோ, அப்பணியை அந்நிரலில் வெவ்வேறு இடங்களில் நிறைவேற்ற வேண்டிய தேவை ஏற்பட்டாலோ, அப்பணியை ஒரு செயல் கூறிடம் ஒப்படைத்துவிட்டு, வேண்டியபோது அந்தச் செயல்கூறினை அழைத்துக்கொள்ளலாம். பயனர் அல்லது நிரலர் (Programmer) குறிப் பிட்ட பணிகளை நிறைவேற்றுவதற்காகத் தாமே செயல்கூறுகளை உரு

வாக்கி வைத்துக்கொண்டு, நிரலின் பல்வேறு இடங்களில் அவற்றைப் பயன்படுத்திக்கொள்ள முடியும். எனவே, செயல்கூறு என்பது, விரும் பும் விடையைப் பெறுவதற்காக, வரிசையாகச் செயல்படுத்த வேண்டிய கட்டளைத் தொகுதியை உள்ளடக்கிய ஒருவகை நிரலே ஆகும். வேறொரு செயல்கூறை அழைக்கின்ற செயல்கூறு ''அழைக்கும் செயல்கூறு'' (calling function) எனப்படும். அழைக்கப்படுகின்ற செயல்கூறு, ''அழைக்கப்படும் செயல்கூறு'' (called function) எனப்படும். அழைக்கப் படும் செயல்கூறு அளபுருக்களைக் கொண்டிருக்கலாம். இல்லாமலும் இருக்கலாம்.

செயல்கூறுகள், செயல்கூறு அழைப்பின் மூலம் இயக்கப்படுகின் றன. செயல்கூறு அழைப்பு, குறிப்பிட்ட செயல்கூறின் பெயரைக் குறிப் பிட்டு அழைக்கிறது. அவ்வாறு அழைக்கும்போது, குறிப்பிட்ட பணி யைச் செய்து முடிக்கும் பொருட்டு, அழைக்கப்பட்ட செயல்கூறினுக்குத் தேவைப்படும் தகவல்களை அளபுருக்கள் வடிவில் அனுப்பிவைக்கிறது. மாறிகளை அறிவிப்பதுபோன்றே செயல்கூறுகளும் அறிவிக்கப்படுவது சி-மொழியில் பின்பற்றப்படும் சிறந்த நடைமுறையாகும். ஒரு செயல் கூறு அறிவிப்பு (function declaration) என்பது, 'செயல்கூறின் மாதிரி வடி வம்' (Function Prototype) அல்லது 'செயல்கூறு மாதிரியம்' (Function Model) என்று அழைக்கப்படுகிறது. செயல்கூறின் மாதிரி வடிவம் நான்கு கூறு களைக் கொண்டது:

- செயல்கூறின் பெயர்
- திருப்பி அனுப்பப்படும் மதிப்பின் தரவினம்
- அளபுருக்களின் எண்ணிக்கை
- ஒவ்வோர் அளபுருவின் தரவினம்

எடுத்துக்காட்டாக, இரண்டு எண்களைக் கூட்டி, விடையை, அழைப் பவர்க்கு அனுப்பிவைக்கும் ஒரு பயனர்-வரையறுத்த செயல்கூறினைக் காண்க:

int add (int, int);

மேற்கண்ட செயல்கூறு அறிவிப்புக் கூற்றில் add என்பது செயல் கூறின் பெயர். இரண்டு அளபுருக்களை ஏற்கிறது. இரண்டும் int இனத்தைச் சார்ந்தவை. இறுதியில் ஓர் int மதிப்பையே திருப்பியனுப்பும். செயல்கூறு இவ்வாறு வரையறுக்கப்படுகிறது:

```
int add(int a, int b)   /* function header  statement */
{
   return (a+b);
}
```

பயனர்–வரையறுத்த செயல்கூறு

செயல்கூறின் மாதிரி வடிவம் அதாவது செயல்கூறு அறிவிப்புக் கூற்று, அரைப்புள்ளியுடன் முற்றுப்பெறுகிறது. ஆனால் செயல்கூறின் தலைப்புக் கூற்று அரைப்புள்ளியுடன் முற்றுப்பெறுவதில்லை. ஒரு செயல் கூறு வரையறுக்கப்படும்போது முதல் கூற்றாக அமைவது, செயல்கூறு தலைப்புக் கூற்றாகும்.

ஒரு செயல்கூறை வரையறுப்பது என்பதன் பொருள், அதற்குரிய கட்டடளைகளை நெளிவு அடைப்புக்குறிகளுக்குள் { } எழுதுவதாகும். நெளிவு அடைப்புக்குறிகளுக்குள் எழுதப்படும் குறிமுறை (code), செயல் கூறின் உடல்பகுதி (body) அல்லது தொகுதி (block) என்று அழைக்கப் படும். ஒரு செயல்கூறினுள் அறிவிக்கப்படும் மாறிகள் அனைத்தும் உள் ளமை மாறிகள் (local variables) ஆகும். அவை எந்தக் செயல்கூறினுள் வரையறுக்கப்பட்டனவோ அந்தச் செயல்கூறினுக்கு மட்டுமே தெரிந்தவை. ஒரு செயல்கூறின் அளபுருக்களும் உள்ளமை மாறிகளே. அழைக்கும் செயல்கூறு, அழைக்கப்படும் செயல்கூறு-இரண்டுக்கும் இடையே தகவல் பரிமாற்றத்துக்கான ஒரு வழிமுறையை அளபுருக்கள் வழங்குகின்றன.

add() செயல்கூறினை இயக்கும் முழு நிரலைக் கீழே காண்க:

```
#include <stdio.h>
#include <conio.h>
main()
{
    int a, b,c;
   int add(int, int);
    a = 12;
    b = 11;
     c = add(a,b);       /* a யும் b யும் மெய்யான அளபுருக்கள் */
   printf("%d\n", c);
}
int add(int x, int y)  /*  x உம் y உம் முறையான அளபுருக்கள்*/
{
     return(x+y);
}
```

அழைக்கும் செயல்கூறில் வரையறுக்கப்படும் அளபுருக்கள் 'மெய்யான அளபுருக்கள்' (Actual Parameters) எனப்படுகின்றன. அழைக்கப்படும் செயல்கூறினுக்கு அனுப்பப்படவேண்டிய மெய்யான மதிப்புகளை அவை கொண்டுள்ளன.

அழைக்கப்படும் செயல்கூறில் வரையறுக்கப்படும் அளபுருக்கள் 'முறையான அளபுருக்கள்' (Formal Parameters) எனப்படுகின்றன. செயல்கூறு இயக்கப்படும்போது, மெய்யான அளபுருக்களின் மதிப்புகளை இவையே பெற்றுக்கொள்கின்றன.

மேற்கண்ட நிரலில்,

c = add (a+b);

என்னும் மதிப்பிருத்து கூற்று செயல்படுத்தப்படும்போது, நிரலின் கட்டுப்பாடு add() செயல்கூறினுக்கு மாற்றப்படுகிறது. a, b ஆகியவை மெய்யான அளபுருக்கள் ஆகும். காரணம், add( ) செயல்கூறு இயக்கப்படும்போது, அதற்கு அனுப்பிவைக்கப்படவேண்டிய மெய்யான மதிப்புகளை இந்த அளபுருக்களே கொண்டுள்ளன. இந்த மதிப்புகளை அழைக்கப்பட்ட செயல்கூறின் முறையான அளபுருக்களான x, y ஆகியவை பெற்றுக்கொள்கின்றன. add( ) செயல்கூறு அழைக்கப்படும்போது, மெய்யான அளபுருக்களின் மதிப்புகள், முறையான அளபுருக்களில் ஒன்

றுக்கு ஒன்று பொருத்தமுறும்படியாக, நகலெடுக்கப்படுகின்றன. இத்த கைய செயல்நுட்பம் 'மதிப்பு மூலம் அழைப்பு' (call by value) என்று அழைக்கப்படுகிறது. add( ) செயல்கூறு, விடையை அழைக்கும் செயல் கூறினுக்கு திருப்பி அனுப்புகிறது. C = add(a+b) என்னும் கூற்றின் வலப் பக்கம் ஒரு செயல்கூறு அழைப்புக் கோவை உள்ளது. அக்கோவையின் மதிப்பு, add() செயல்கூறு திருப்பி அனுப்பும் மதிப்பாகும்.

செயல்கூறு இயங்கி முடிந்ததும், நிரலின் கட்டுப்பாடு, அழைக்கும் செயல்கூறில், எந்த இடத்திலிருந்து கட்டுப்பாடு மாற்றப்பட்டதோ அந்த இடத்துக்கே திரும்பிவரும். செயல்கூறு அழைப்புக் கூற்றைப் பொறுத்த வரை நிரலின் கட்டுப்பாடு, அழைக்கும் செயல்கூறில் அடுத்தக் கூற்றுக் குத் திரும்பிவரும். இன்னொரு முக்கிய விவரத்தை மனதில் கொள்ள வேண்டும். அழைக்கப்பட்ட செயல்கூறிலுள்ள உள்ளமை மாறிகளின் மதிப் புகள் இழக்கப்படுகின்றன. செயல்கூறு செயல்பட்டு முடிந்ததும் அழைக் கப்பட்ட செயல்கூறின் உள்ளமை மாறிகள் அழிக்கப்படுகின்றன. ஒரு செயல்கூறினுக்குள் அறிவிக்கப்படுகின்ற, வரையறுக்கப்படுகின்றன, பயன் படுத்தப்படுகின்ற மாறிகளே உள்ளமை மாறிகள் எனப்படுகின்றன.

மேற்கண்ட எடுத்துக்காட்டில், add() செயல்கூறு, 'மதிப்பு மூலம் அழைத்தல்' முறையில் இயக்கப்படுகிறது. மதிப்பு மூலம் அழைத்தல் முறையில் அளபுருக்கள் அனுப்பப்படும்போது, அளபுரு மதிப்புகளின் நகல்கள் உருவாக்கப்பட்டு, அழைக்கப்பட்ட செயல்கூறினுக்கு அனுப்பி வைக்கப்படுகின்றன. அழைக்கப்பட்ட செயல்கூறினுள் அம்மதிப்புகளில் செய்யப்படும் மாற்றங்கள், அழைக்கும் செயல்கூறிலுள்ள மூல மாறிக ளின் மதிப்புகளை எவ்வகையிலும் பாதிப்பதில்லை. அழைக்கும் செயல் கூறிலுள்ள உள்ளமை மாறிகளின் முகவரிகளை அழைக்கப்படும் செயல் கூறு அறிந்து வைத்திருப்பின், அழைக்கப்பட்ட செயல்கூறு, அழைக்கும் செயல்கூறிலுள்ள உள்ளமை மாறிகளின் மதிப்புகளை மாற்றியமைக்க முடியும். 'முகவரி மூலம் அழைப்பு' (call by address) கருத்துரு மூலம் இப்பணியைச் சாதிக்க முடியும்.

எடுத்துக்காட்டு:

கீழே உள்ள நிரலை நோக்குக:

```
#include <stdio.h>
main()
{
    int i;
    void change(int *);
    i = 20;
    change(&i);
   printf("%d\n", i);
}
void change(int *x)
{
        *x = 23;
}
```

இந்த நிரலின் வெளியீடு 23 ஆக இருக்கும். அதாவது, அழைக்கும் செயல்கூறிலுள்ள உள்ளமை மாறியின் மதிப்பு மாற்றப்பட்டு விட்டது. change() செயல்கூறின் முறையான அளபுரு x, ஒரு முழுஎண் சுட்டு (in teger pointer) ஆகும். இது, அழைக்கும் செயல்கூறின் உள்ளமை மாறி i-ன் முகவரியை ஏற்றுக்கொள்கிறது. எனவே x, i- ஐச் சுட்டுகிறது. அழைக்கப் பட்ட செயல்கூறில், *x = 23; என்ற மதிப்பிருத்து கூற்று, x சுட்டும் முகவரி யில் புதிய மதிப்பை இருத்துகிறது. x, i-ஐச் சுட்டுவதால், i- ன் மதிப்பு மாறிவிடுகிறது.

ஒரு சுட்டு, சுட்டுகின்ற நினைவக முகவரியில் இருக்கும் மதிப்பைப் பெற்றுத்தர, உள்நோக்கு செயற்குறி * பயன்படுத்தப்படுகிறது என்பதை ஏற்கெனவே பார்த்தோம். கீழேயுள்ள நிரல்பகுதியை நோக்குக:

```
int i = 32;
int * p;
int m;
p = &i;
m = *p;
```

மதிப்பிருத்து கூற்றின் வலப்பக்கம் இருக்கும் *p, p-யினால் சுட்டப்ப டும் முகவரியிலுள்ள மதிப்பைப் பெற்றுத் தருகிறது. அம்மதிப்பு, i என்ற மாறியின் மதிப்பே ஆகும். வலப்பக்கம் *p பயன்படுத்துவதன் மூலம், ஒரு நினைவக முகவரியில் இருக்கும் மதிப்பைப் பெற்றுத்தர முடியுமே ஒழிய அங்குள்ள மதிப்பை மாற்றியமைக்க முடியாது. change() செயல்

கூறில், சுட்டு மாறி p- உடன், உள்நோக்கு செயற்குறி * சேர்த்து (*p என), மதிப்பிருத்து கூற்றின் இடப்பக்கம் பயன்படுத்தியுள்ளோம். இவ்வாறு, உள்நோக்கு செயற்குறியுடன் சுட்டு மாறி, இடப்பக்கம் இடம்பெறுமாயின் அது, நினைவக இருப்பிடத்தின் முகவரியைத் தருமே அல்லாது அதிலுள்ள மதிப்பினைத் தராது. எனவே *x = 23; என்ற கூற்றின் மூலம், x சுட்டும் முகவரியில் 23 என்கிற மதிப்பு இருத்தப்படுகிறது.

சுருங்கக்கூறின், உள்நோக்கு செயற்குறியை(*), வலப்பக்கம் பயன் படுத்தினால், அது தரவின் மதிப்பைப் பெற்றுத்தரும்; மதிப்பை மாற்றி அமைக்காது. அது 'படிக்க மட்டும்' ஆன மதிப்பு. உள்நோக்கு செயற் குறி, இடப்பக்கம் பயன்படுத்தப்படுமாயின், அது இருப்பிட முகவரி யைத் தரும்; அதிலுள்ள மதிப்பை மாற்றியமைக்க முடியும்.

செயல்கூறு பற்றிய மற்றுமொரு கருத்துருவை நீங்கள் அறிந்து கொள்ள வேண்டும். கீழேயுள்ள நிரல்பகுதியை நோக்குக. இது செயல் படும்விதம் சற்றே வேறுபாடானது. இதை நீங்கள் அறிந்திருக்கமாட்டீர்கள்.

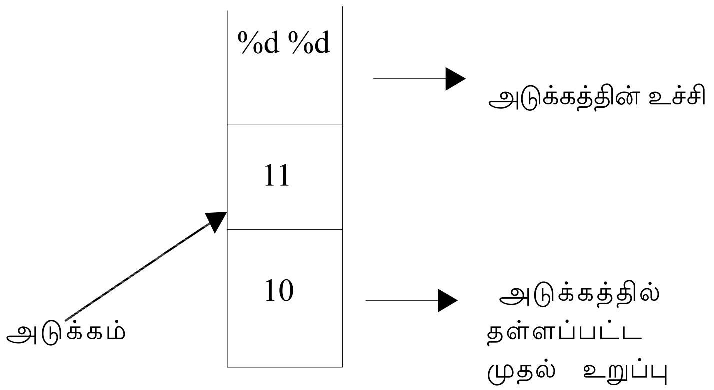
int i = 10;

printf ("%d %d", i, i++);

இதன் வெளியீடு 10 10 அல்லது 10 11 என இருக்கும் என்று நீங்கள் எண்ணினால் ஏமாற்றம் அடைவீர்கள். இந்த நிரல்பகுதியில் ஒரு கருத்துரு மறைந்து கிடக்கிறது. சி-மொழியில், ஒரு செயல்கூறு இயக் கப்படும்போது, அதன் அளபுருக்கள் வலமிருந்து இடமாக, ஓர் அடுக் கத்தில் (Stack) இருத்தப்படுகின்றன. 'அடுக்கம்' என்பது 'கடைபுகு-முதல் விடு' (Last - In- First - Out /LIFO) அமைப்பில் இருக்கும்.

சாப்பிடும் தட்டுகள் அல்லது டம்ளர்கள் அடுக்கிவைக்கப்படுவதை நோக்குக. சாப்பாட்டு அறையில், கழுவப்பட்ட பின், அவை ஒன்றன்மீது ஒன்றாக அடுக்கிவைக்கப்படுகின்றன. கடைசியாக வைக்கப்பட்ட தட்டு அல்லது டம்ளர் (உச்சியில் இருப்பது) முதலில் எடுக்கப்படும். பிறகு அடுத்தது எடுக்கப்படும்.

'அடுக்கம்' என்பதைப் படத்தில் காண்க:



மேலே கண்ட printf() செயல்கூறு அழைக்கப்படும்போது, அடுக்கத்தில் முதலில் தள்ளப்படும் மதிப்பு, i++ கோவையின் மதிப்பாகும். இது ஒரு பின்னொட்டுக் கோவை என்பதால், கோவையின் மதிப்பு 10 அடுக்கத்தில் தள்ளப்படும். அதன்பிறகு மிகுப்புச் செயற்குறியின் (++) பலனாக, i-ன் மதிப்பு ஒன்று மிகுக்கப்பட்டு, 11 என ஆகும். வலமிருந்து இடமாக அடுத்து இடம்பெறும் அளபுரு i . அதன் மதிப்பு இரண்டாவதாக அடுக்கத்தில் தள்ளப்படும். அதாவது, 11 அடுக்கத்தின் மேலிருக்கும். கடைசி அளபுருவாக, "%d %d" என்னும் கட்டுப்பாட்டுச் சரம் அடுக்கத்தில் தள்ளப்படும். இனி தள்ளப்படுவதற்கு அளபுருக்கள் எதுவும் இல்லாதபோது, printf() செயல்கூறு, அடுக்கத்தின் உறுப்புகளை உச்சியிலிருந்து ஒவ்வொன்றாக எடுக்கத் தொடங்கும். முதலில் வெளியில் எடுக்கப்படும் உறுப்பு "%d %d". செயல்கூறு கட்டுப்பாட்டுச் சரத்தை ஆய்வு செய்யும். முதலில் %d இருப்பதால், அடுக்கத்திலிருந்து அடுத்த உறுப்பை (மதிப்பு 11) வெளியிலெடுத்து ஒரு முழுஎண்ணாகக் காட்டும். அடுத்துள்ள %d-க்கு, அடுக் கத்திலுள்ள அடுத்த உறுப்பை (மதிப்பு 10) வெளியில் எடுத்து ஒரு முழுஎண் மதிப்பாகக் காட்டும். இப்போது அடுக்கம் காலி ஆகிவிட்டது. திரையில்,

11 10

என்னும் வெளியீட்டைக் காணலாம். சி-மொழி செயல்கூறு செயல்படும் விதத்தை இப்போது நன்கு புரிந்திருப்பீர்கள்.

4.4 சேமிப்பு இனக்குழுக்கள் (Storage Classes)

ஒரு மாறியின் பண்புக்கூறுகளாக இதுவரையில் நாம் பார்த்தவை

- பெயர் (Name)
- இனம் (Type)
- மதிப்பு (Value)

'சேமிப்பு இனக்குழு' என்பது, மாறியுடன் தொடர்புடைய மற்றொரு பண்புக்கூறாகும். சி-மொழி நான்கு இனக்குழுக்களை வழங்குகிறது.

- auto
- static
- register
- extern

ஒரு மாறியின் வரையெல்லையையும் வாழ்நாளையும் (scope and lifetime) தீர்மானிப்பதற்கு அதன் சேமிப்பு இனக்குழு பயன்படுகிறது. auto மாறிகள் உண்மையில் உள்ளமை (local) மாறிகளாகும். அவை அறிவிக் கப்பட்டுள்ள செயல்கூறினுள் நிரலின் கட்டுப்பாடு நுழையும்போது அவை உருவாக்கப்படுகின்றன. செயல்கூறைவிட்டு வெளியேறும்போது அவை அழிக்கப்படுகின்றன. ஒரு செயல்கூறினுள் உருவாக்கப்படும மாறிகள் அச்செயல்கூறின் உள்ளமை மாறிகள் ஆகும். அவற்றின் வரையெல் லையும் வாழ்நாளும் அச்செயல்கூறினுள்ளேயே முடிந்துவிடுகின்றன. ஒரு செயல்கூறு செயல்பட்டு முடியும்போது, அனைத்து உள்ளமை மாறிகளும் அழிக்கப்படுகின்றன என்பதே இதன்பொருள். உள்ளமை மாறிகளின் மதிப்புகளை அச்செயல்கூறினுக்கு வெளியே அணுக முடியாது. அதா வது, உள்ளமை மாறிகளின் வரையெல்லை அவை அறிவிக்கப்பட்டுள்ள செயல்கூறினுள் அடங்கிவிடுகிறது. கீழேயுள்ள நிரலை நோக்குக:

```
#include <stdio.h>
main()
{
 add();
 add();
}
add()
{
    int i = 0;
    i = i + 1;
   printf("%d\n", i);
}
```

இந்த நிரலின் வெளியீடு,

1

1

என அமையும். i என்பது, add() செயல்கூறின் உள்ளமை மாறி. ஒவ்வொரு முறை add() செயல்கூறு அழைக்கப்படும்போதும், i என்னும் மாறி புதிதாய் உருவாக்கப்பட்டு 0 என்னும் தொடக்கமதிப்பு இருத்தப்படுகிறது. எனவே விடை எப்போதுமே 1 ஆக இருக்கும். இதே மாறி static இனமாக அறிவிக்கப்பட்டிருப்பின், செயல்கூறு செயல்பட்டு முடிந்த பின்னும் அதன் மதிப்பு தக்கவைத்துக் கொள்ளப்பட்டிருக்கும். add() செயல்கூறினை கீழ்க்காணுமாறு மாற்றி அமைக்கவும்:

```
add()
{
    static int i = 0;
    i = i + 1;
   printf("d\n", i);
}
```

இதில், i என்னும் மாறி static இனமாக அறிவிக்கப்பட்டுள்ளது. இது add() செயல்கூறின் உள்ளமை மாறியாகும். இதன் வரையெல்லை add() செயல்கூறுக்குள் மட்டுமே. இதன் மதிப்பை வேறெந்த செயல்கூறும் அணுக முடியாது. static மாறிகள், முதன்முதலில் செயல்கூறு அழைக்கப்படும்போது ஒரேயொரு முறை மட்டுமே உருவாக்கப்படுகின்றன. செயல்கூறு செயல்பட்டு முடிந்தபின்னும் அவற்றின் மதிப்புகளைத் தக்கவைத்துக் கொள்ள முடியும். இதுவே static மாறிகளின் மிகப் பெரிய அனுகூலமாகும். எனவே ஒவ்வொரு முறை add() செயல்கூறு அழைக்கப்படும் போதும் i-ன் மதிப்பு ஒன்று மிகுக்கப்பட்டு வெளியீடு,

```
1
2
```

என அமையும்.

மேற்கண்ட நிரலில், i என்னும் மாறியை ஒரு முழுதளாவிய (global) மாறியாக அறிவித்து, சற்றே மாற்றி எழுதிப் பார்ப்போம். முழுதளாவிய மாறிகள் main() செயல்கூறினுக்கு மேலே அறிவிக்கப்படுகின்றன. இந்த மாறிகளை, நிரலில் உள்ள அனைத்துச் செயல்கூறுகளும் அணுகலாம், மாற்றியமைக்கலாம்.

```
#include <stdio.h>
int i = 0;
main()
{
    add();
    add();
}
add()
{

  i = i + 1;
   printf("%d\n",i);
}
```

இந்த நிரலின் வெளியீடு இவ்வாறு அமையும்:

1

2

மேற்கண்ட நிரலில், i என்பது ஒரு முழுதளாவிய மாறியாகும். அனைத்துச் செயல்கூறுகளும் இதனை அணுகவும் மாற்றியமைக்கவும் முடியும். static மாறிகள், உள்ளமை மாறிகளின் பண்பியல்புகளைக் கொண்டுள்ளன. முழு நிரலும் செயல்பட்டு முடிந்தபிறகே static மற்றும் முழுதளாவிய மாறிகளின் வாழ்நாள் முடிவுக்கு வருகின்றது. static மாறியின் வரையெல்லை அது அறிவிக்கப்பட்டுள்ள செயல்கூறுக்குள் மட்டுமே. ஆனால் முழுதளாவிய மாறியின் வரையெல்லை, நிரலிலுள்ள அனைத்து செயல்கூறுகளுக்கும் பரவிக் கிடக்கிறது. இந்த வேறுபாடுகள் தவிர, ஒரு செயல்கூறினுக்கு வெளியே அறிவிக்கப்படும் முழுதளாவிய மாறிகள் இயல்பாகவே static இனக்குழுவைச் சார்ந்தவையே. ஆனால், static மாறி, ஒரு முழுதளாவிய மாறி அல்ல என்பதை மனதில் கொள்க.

register மாறிகள், auto மாறிகளைப் போன்றே இயங்குகின்றன. ஒரு மாறி register இனமாக அறிவிக்கப்படுமாயின், அதன் மதிப்பு, கணிப்பொறியின் அதிவேக வன்பொருள் பதிவகங்கள் (registers) ஒன்றில் இருத்தி வைக்கப்படுகிறது. நினைவக அணுகல் நேரத்தைக் குறைத்து, செயல்பாடுகளை வேகப்படுத்த register மாறிகள் பயன்படுகின்றன.

முழுதளாவிய மாறிகளை, ஒரு நிரல் கோப்பில் உள்ள செயல்

கூறுகளால் அணுகமுடியும். முழுதளாவிய மாறிகள் அறிவிக்கப்பட்டுள்ள கோப்பு அல்லாத பிற கோப்புகளில் உள்ள செயல்கூறுகள் அணுக வேண்டிய தேவையிருப்பின், அவற்றை extern இனக்குழுவில் அறிவித்துக் கையாளலாம். எடுத்துக்காட்டாக, count என்னும் முழுதளாவிய முழுஎண் மாறியை ஒரு கோப்பில் அறிவித்து, அதனை வேறொரு கோப்பில் எடுத்தாள வேண்டுமெனில், இரண்டாவது கோப்பில், அந்த மாறியைப் பயன்படுத்துவதற்கு முன்பாக,

extern int count ;

என்கிற அறிவிப்பு இருக்க வேண்டும். extern மாறிகள் முழுதளாவிய வரையெல்லை கொண்டவை. அவற்றின் வாழ்நாள், நிரல் செயல்பட்டுக் கொண்டிருக்கும் வரை தொடர்ந்து நிலைத்திருக்கும்.

4.5 நிபந்தனைக் கூற்றுகள் (Conditional Statements)

4.5.1 if கூற்று

சி-மொழியில் நிபந்தனைக் கூற்றுகள் பூலியன் கோவைகள் என்னும் கருத்தையே சார்ந்துள்ளன. if கூற்று, நிபந்தனைக் கிளைபிரித்தலைக் கட்டுப்படுத்துகிறது. பூலியன் கோவை அல்லது வேறு பெயரில் கூறுவ தெனில் ஒப்பீட்டுக் கோவை, ‘சரி’ அல்லது ‘தவறு’ என்கிற மதிப்பையே விடையாகத் தரும். சி-மொழியில் 0 என்பது ‘தவறு’, 0 அல்லாத மதிப்பு ‘சரி’ எனக் கொள்ளப்படும். நிபந்தனைக் கோவையின் மதிப்பு ‘சரி’ அதாவது 0 அல்லாத மதிப்பாக இருந்தால் if கூற்றின் உடற்பகுதியி லுள்ள கட்டளைகள் நிறைவேற்றப்படும். if கூற்றுக்கு இரண்டு வடிவங் கள் உள்ளன:

```
1. if (ஒப்பீட்டுக் கோவை)
          கட்டளை;
2. if  (ஒப்பீட்டுக் கோவை)
          கட்டளை-1;
   else
          கட்டளை-2  ;
```

முதல் வடிவில், ஒப்பீட்டுக் கோவை ‘சரி’ (0 அல்லாத மதிப்பு) எனில், அடுத்துள்ள கட்டளை செயல்படுத்தப்படும். கோவையின் மதிப்பு ‘தவறு’ (0 மதிப்பு) எனில், அடுத்துள்ள கட்டளை புறக்கணிக்கப்படும். இரண்டாவது வடிவில், else பயன்படுத்தப்பட்டுள்ளது. கோவையின் மதிப்பு ‘தவறு’ எனில் கட்டளை-2 நிறைவேற்றப்படும். இரண்டு வடிவங்களி

லும் நிரலின் கட்டுப்பாடு, அடுத்து if கூற்றுக்குப் பிறகுள்ள கட்டளைக்குத் தாவிவிடும்.
சாதாரண if கூற்றை விளக்கும் சி-நிரலைக் காண்க:

```
#include <stdio.h>
main()
{
        int x;
        printf("Enter an integer: ");
        scanf("%d", &x);
        if (x > 0)
           printf("The value is positive\n");
}
```

இந்த நிரல், பயனிடமிருந்து ஓர் எண்ணை ஏற்கும். if கூற்றின் நிபந்தனைக் கோவைமூலம் அந்த எண் 0-ஐவிடப் பெரிதா எனப் பரிசோதிக்கும். பெரிதெனில் ஒரு செய்தியைத் திரையில் காட்டும். இல்லையேல் நிரல் மவுனமாகிவிடும். வெளியீடு எதுவும் இராது. நிரலில் உள்ள (x > 0) என்னும் பகுதி பூலியன் கோவை எனப்படுகிறது. சி-மொழி, இந்தக் கோவையை மதிப்பிட்டு, செய்தியைக் காட்டலாமா கூடாதா என்பதைத் தீர்மானிக்கிறது. பூலியன் கோவையின் மதிப்பு 'சரி' எனில், if கூற்றுக்கு அடுத்திருக்கும் ஒற்றைக் கட்டளையை (அல்லது if கூற்றுக்கு அடுத்திருக்கும் நெளிவு அடைப்புக் குறிகளுக்குள் இருக்கும் கட்டளைகளின் தொகுதியை) சி-மொழி நிறைவேற்றும். பூலியன் கோவை 'தவறு' எனில், if கூற்றுக்கு அடுத்திருக்கும் கட்டளையை அல்லது கட்டளைகளின் தொகுதியை சி-மொழி நிறைவேற்றாமல் தவிர்த்துவிடும். இன்னோர் எடுத்துக்காட்டைப் பாருங்கள்:

```
#include <stdio.h>
main()
{  int x;
    scanf("%d", &x);
    if (x < 0)
       printf("The value is egative\n");
     else if (x == 0)
      printf("The value is zero\n");
    else
      printf("The value is positive\n");
}
```

இந்த எடுத்துக்காட்டில், if-else-if கட்டளை அமைப்புப் பயன் படுத்தப்பட்டுள்ளது. இதனைப் பின்னல் (nested) if-else அமைப்பு என அழைக்கலாம். தெரிவுகளின் (choices) அடிப்படையில் சில செயல்பாடு களை நிறைவேற்றப் பின்னல் if-else அமைப்பு பயன்படுத்தப்படுகிறது. கணிதத்தில் ஓர் எளிமையான கணக்கை எடுத்துக்கொள்வோம். பயன ரின் தெரிவு அடிப்படையில் கூட்டல், கழித்தல், பெருக்கல், வகுத்தல் ஆகிய செயல்பாடுகள் நிறைவேற்றப்படவேண்டும். தெரிவுகளைத் திரை யில் காட்டி பயனரின் விருப்பத்தைப் பெறலாம். இந்த நிரலை இரண்டு வழிகளில் எழுதமுடியும்:

எளிய if பயன்படுத்தி-

```
#include <stdio.h>
main()
{   int a,b,c,choice;
    scanf("%d%d", &a,&b);           /*b சுழியம் இல்லை */
    printf("1. addition\n");        /* தேர்வு 1 */
    printf("2. subtraction\n");     /* தேர்வு 2 */
    printf("3. multiplication\n");  /* தேர்வு 3 */
    printf("4. division\n");        /* தேர்வு 4 */
    scanf("%d", &choice);
    if(choice == 1)
      c = a + b;
    if(choice == 2)
      c = a - b;
    if(choice == 3)
      c = a * b;
    if(choice == 4)
      c = a / b;
    printf("the result = %d\n", c);
}
```

மேற்கண்ட நிரல், தெரிவின் அடிப்படையில் கணக்கீட்டைச் செய்து முடித்து விடையைத் திரையில் காட்டும். பயனர் தெரிவு எதுவாயினும், இந்த நிரலில் செய்யப்படும் ஒப்பீடுகள் நான்கு ஆகும். தேவையற்ற ஒப்பீடுகளும் செய்யப்படுகின்றன. (choice ==1) என்பது 'சரி' ஆக இருக்

கும்போது, பிற ஒப்பீடுகளைச் செய்யத் தேவையில்லை; தவிர்த்துவிட லாம். அவ்வாறு செய்யவேண்டுமெனில் பின்னல் if-else பயன்படுத்த வேண்டும்.

பின்னல் if-else அமைப்பைப் பயன்படுத்தி-

```
#include <stdio.h>
main()
{          int a,b,c;
   int choice;
   printf("Enter two integers: ");
    scanf("%d%d", &a,&b);         /* b சுழியம் இல்லை */
    printf("1. addition\n");          /* தேர்வு 1 */
    printf("2. subtraction\n");      /* தேர்வு 2 */
   printf("3. multiplication\n");  /* தேர்வு 3 */
    printf("4. division\n");          /* தேர்வு 4 */
   printf("Enter your choice: ");
   scanf("%d", &choice);
   if(choice == 1)
     c = a + b;
    else
     if(choice == 2)
         c = a - b;
      else
       if(choice == 3)
         c = a * b;
        else
          if(choice == 4)  /* ஒப்பிடு கட்டாயமில்லை */
           c = a / b;
   printf("the result = %d\n", c);
}
```

மேற்கண்ட நிரல், பின்னல் if-else அமைப்பைப் பயன்படுத்தியுள்ளது. முதல் நிபந்தனை 'சரி' எனில், அதாவது (choice == 1) என்பது 'சரி' எனில், c = a + b ; என்னும் கட்டளை நிறைவேற்றப்படும். else- க்குப்

பிறகு அமைந்துள்ள கட்டளை தவிர்க்கப்படும். எனவே, முதல் நிபந் தனை 'சரி' ஆக இருப்பின் ஒரேயோர் ஒப்பீடு மட்டுமே செய்யப்படு கிறது; பிற ஒப்பீடுகள் தவிர்க்கப்படுகின்றன. முதல் நிபந்தனை 'தவறு' என ஆகும்போதுதான், நிரலானது இரண்டாவது ஒப்பீட்டுக்குச் செல் லும். இதுபோலவே, அடுத்தடுத்த நிபந்தனை சரி பார்க்கப்படும். இந்த நிரல் முந்தைய நிரலைவிட வேகமாகச் செயல்படும். ஆனால் தெரிவு 4 எனில், இரண்டு நிரல்களுமே நான்கு ஒப்பீடுகளைச் செய்யவேண்டியி ருக்கும்.

4.5.2 switch-case கூற்று

சி-மொழி நிரலில் சில சூழ்நிலைகளில் சிக்கல்மிகுந்த பின்னல் if-else அமைப்புக்கு மாற்றாக switch-case கூற்றினைப் பயன்படுத்த முடி யும். சிக்கலான நிபந்தனை மற்றும் கிளைபிரித்தல் செயல்பாடுகளைக் கட்டுப்படுத்த switch-case கூற்று உதவுகிறது. switch கூற்று நிரலின் கட்டுப் பாட்டை அதன் உடற்பகுதியிலுள்ள ஒரு கூற்றுக்கு மாற்றுகிறது. switchcase கூற்றின் தொடரமைப்பைக் காண்க:

```
switch (நிபந்தனைக் கோவை)
{
   case   மாறிலிக்கோவை-1:
          ..............................
          break ;
   case   மாறிலிக்கோவை-2:
          ..............................
          break ;
     ..............................
   ..............................
   default :
          ....................................
}
```

switch கூற்றினுள் இருக்கும் நிபந்தனைக் கோவை மற்றும் மாறிலிக் கோவைகளின் தரவினம் முழுஎண் இனமாக இருக்கவேண்டும். எந்த case-ன் மாறிலிக் கோவை மதிப்பு, switch- ன் நிபந்தனைக் கோவை மதிப்புடன் ஒத்துள்ளதோ அந்த case கூற்றுக்கு நிரலின் கட்டுப்பாடு மாற்றப்படும். switch கூற்றில் எத்தனை case கூற்றுகள் வேண்டுமானா லும் இருக்கலாம். ஆனால், ஒரு switch கூற்றுக்குள் இரண்டு case கூற்றுகள் ஒரே மாறிலிக் கோவையைக் கொண்டிருக்கக்கூடாது. switch கூற்றின் உடற்பகுதிக்குள் நிறைவேற்றத்துக்குத் தேர்ந்தெடுக்கப்பட்ட case

கூற்றில் தொடங்கி, உடற்பகுதியின் இறுதி வரையிலோ அல்லது ஒரு break கூற்றைச் சந்திக்கும் வரையிலோ அனைத்து case கூற்றுகளின் கட்டளைகளும் நிறைவேற்றப்படும். ஒரு குறிப்பிட்ட case கூற்றின் கீழுள்ள கட்டளைத்தொகுதி நிறைவேற்றப்பட்டபின் நிரலின் கட்டுப்பாடு switch-ன் உடற்பகுதியை விட்டு வெளியே வந்துவிட break கட்டளையை பயன்படுத்த வேண்டும். இல்லையேல் அடுத்துள்ள case கூற்றின் கட்டளைத் தொகுதிகளும் நிறைவேற்றப்படும்.

எந்தவொரு case -ன் மாறிலிக் கோவை மதிப்பும் switch -ன் நிபந்தனைக் கோவை மதிப்புக்கு நிகராய் இல்லையெனில், default கூற்றுக்குப் பின்னுள்ள கட்டளைத் தொகுதி நிறைவேற்றப்படும். switch- ன் உடற்பகுதியில் default பகுதி இல்லாதபோது, எந்தவொரு case கூற்றின் மதிப்பும் நிபந்தனைக் கோவையின் மதிப்போடு ஒத்துப்போகவில்லை எனில் switch-ன் உடற்பகுதியிலுள்ள எந்தவொரு கூற்றின் கட்டளைகளும் நிறைவேற்றப்பட மாட்டாது. default கூற்று கட்டாயமற்றது. தேவையெனில் குறிப்பிடலாம். உடற்பகுதியின் இறுதியில்தான் இடம்பெறவேண்டும் என்பதில்லை. எங்கு வேண்டுமானாலும் இடம்பெறலாம். எளிய கணிதச் செயல்பாடுகளை நிறைவேற்ற பின்னல் if-else அமைப்பைப் பயன்படுத்தி எழுதிய நிரல், switch-case கூற்றைப் பயன்படுத்தி எழுதப்பட்டுள்ளது காண்க:

```
#include <stdio.h>
main()
        {  int a,b,c;
           int choice;
           printf("Enter two integers: ");
           scanf("%d%d", &a,&b);            /*b சுழியம் இல்லை*/
           printf("1. addition\n");         /* தேர்வு 1 */
           printf("2. subtraction\n");      /* தேர்வு 2 */
           printf("3. multiplication\n");   /* தேர்வு 3 */
           printf("4. division\n");         /* தேர்வு 4 */
           printf("Enter your choice: ");
           scanf("%d", &choice);
           switch(choice)
          {
             case 1:
                c = a + b;
                printf("%d", c);
                break;
             case 2:
                c = a - b;
                printf("%d", c);
                break;
```

```
            case 3:
                c = a * b;
                printf("%d", c);
                break;
            case 4:
                c = a / b;
                printf("%d", c);
                break;
          default:
                printf("the choice is out of range\n");
    }
```

இந்த நிரல் கூறுநிலை (Modular) அமைப்பில் உள்ளது. பின்னல் if-else அமைப்பு பயன்படுத்தப்பட்ட நிரலைக் காட்டிலும் படித்துப் புரிந்து கொள்ள எளிமையாய் உள்ளது. தரப்பட்ட எழுத்து, உயிரெழுத்தா (vowel), மெய்யெழுத்தா (consonant) என்பதைக் கண்டறியும் இன்னோர் எடுத்துக் காட்டு நிரல்பகுதி காண்க:

```
char ch;
ch = 'a';
switch(ch)
{
     case 'a':
     case 'e':
     case  'i':
     case  'o':
    case  'u':  printf("the given character is  vowel");
                    break;
     default:  printf("the given character is consonant");
}
```

மேலேயுள்ள எடுத்துக்காட்டில், ch என்னும் மாறியின் மதிப்பு 'a' என்பதால் முதல் case 'சரி' என்றாகிறது. அதில் break கூற்று இல்லை என்பதால், நிரல், அடுத்த case கூற்றுக்குச் செல்லும். அங்கும் break இல்லை. break கூற்றைச் சந்திக்கும் வரை அடுத்தடுத்த case கூற்றுகளுக்குச் செல்லும். இறுதியாக case 'u': கூற்றை அடையும். தரப்பட்ட எழுத்து 'a',

‘e’, ‘i’, ‘o’ , ‘u’ இவற்றுள் எதுவாக இருந்தாலும், நிரலானது u-வுக்கு வந்து சேரும். அதற்குரிய கட்டளைகளை நிறைவேற்றும். given character is vowel என்ற செய்தி திரையில் காட்டப்படும். அடுத்து, break கட்டளை இருப்பதால், நிரல், switch கூற்றைவிட்டே வெளியேறும். தரப்பட்ட எழுத்து உயிர் எழுத்தாய் இல்லாதபோது, default பகுதிக்குரிய கட்டளைகள் நிறை வேற்றப்படும். அதன்பின் நிரல், நெளிவு அடைப்புக் குறிகளை விட்டே வெளியேறும்.

4.6 கட்டுப்பாட்டுக் கூற்றுகள் (Control Statements)

if கூற்றில் நிபந்தனை ‘சரி’ எனில், அதாவது if -ல் உள்ள ஒப்பீட் டுக் கோவையின் மதிப்பு ‘சரி’ எனில், if- கூற்றின் உடற்பகுதி ஒரே யொரு முறை மட்டுமே நிறைவேற்றப்படும். சில சூழ்நிலைமைகளில், குறிப்பிட்ட நிபந்தனை ‘சரி’ ஆக இருந்து கொண்டிருக்கும்வரை, ஒரு கட்டளைத் தொகுதியை திரும்பத் திரும்ப நிறைவேற்ற வேண்டிய தேவை இருக்கலாம். இத்தேவையை நிறைவுசெய்யக் கட்டுப்பாட்டுக் (control) கூற்றுகள் அதாவது மடக்குக் (looping) கூற்றுகள் தேவை. மடக்கு (Loop) என்பது நிரலின் ஒரு பகுதி. தொடங்கிய இடத்துக்கே மீண்டும் வந்து தேவையான தடவைகள் திரும்பத் திரும்ப நிறைவேற்றப்படும் கட்ட ளைகளின் தொகுதி. சி-மொழியில் மூன்று வகையான கட்டுப்பாட்டுக் கூற்றுகள் (மடக்குகள்) உள்ளன. அவை: while, for,do while ஆகியன.

4.6.1 while கூற்று

குறிப்பிட்ட நிபந்தனை ‘சரி’ என்ற நிலையில் இருக்கும்வரை, கட் டளைகளின் தொகுதியைத் திரும்பத் திரும்ப செயல்படுத்த while கூற்று பயன்படுகிறது. while கூற்றில், கட்டளைத் தொகுதிக்குள் நுழையும் நிலை யிலேயே நிபந்தனை சரிபார்க்கப்படுகிறது. (மடக்கு, எத்தனை முறை செயல்படுத்தப்பட வேண்டும் என்பதை ஒரு கட்டுப்பாட்டு மாறி மூலம் தீர்மானிக்கலாம்). கீழேயுள்ள நிரல் ஒன்று முதல் பத்து வரையிலான எண்களை அடுத்தடுத்த வரிகளில் திரையில் காட்டும். இங்கே, printf( ) செயல்கூறை 10 முறை செயல்படுத்த ‘மடக்கு’ தேவைப்பட்டுள்ளது.

```
#include <stdio.h>
main()
{
   int i;
    i =  1;                    /* தொடக்க மதிப்பிருத்தல் */
    while(i <= 10)            /*நிபந்தனை */
    {
         printf("%d\n", i);  /*செயலாக்கக் கூற்று */
        i = i + 1;            /*புதுப்பித்தல் */
    }
}
```

இந்த நிரலில், i என்னும் மாறி கட்டுப்பாட்டு மாறியாகப் பயன்படுகிறது. while மடக்கின் நிறைவேற்றத்தை அதுவே கட்டுப்படுத்துகிறது. while கூற்றுக்கு முன்பே கட்டுப்பாட்டு மாறி முறைப்படி அறிவிக்கப்பட்டு, தொடக்க மதிப்பிருத்தப்பட்டுள்ளது. இந்த நிரல், முதல் பத்து இயல்பு எண்களைக் காட்டவேண்டும் என்பதாலும், காட்டவேண்டிய முதல் எண் 1 என்பதாலும் கட்டுப்பாட்டு மாறியில் தொடக்க மதிப்பாக 1 இருத்தப் பட்டுள்ளது. மடக்கு, 10 முறை நிறைவேற்றப்படவேண்டும். எனவே நிபந் தனை i <= 10 என அமைக்கப்பட்டுள்ளது. குறிப்பட்ட நிபந்தனை 'சரி' எனில் while மடக்கின் உடற்பகுதி 10 முறை நிறைவேற்றப்படும். ஒவ் வொரு முறை மடக்கு நிறைவேற்றப்படும் போதும் கட்டுப்பாட்டு மாறி யின் மதிப்பு 1 மிகுக்கப்படுகிறது. காரணம், சரியாகப் பத்துமுறை மட் டுமே மடக்கு நிறைவேற்றப்படவேண்டும். சோதனை நிபந்தனை 'தவறு' எனில், நிரலின் கட்டுப்பாடு மடக்கினை விட்டு வெளியே வந்துவிடும். வெளியேறியபின், மடக்கின் உடற்பகுதியை அடுத்து இடம் பெற்றுள்ள கட்டளையிலிருந்து, நிரலின் செயலாக்கம் தொடரும். மீண்டும் ஒருமுறை நிரலை உற்று நோக்குங்கள். கட்டுப்பாட்டு மாறியில் தொடக்க மதிப்பு இருத்தப்பட்ட உடனே அதன் மதிப்பு while கூற்றில் பரிசோதிக்கப்படு கிறது. while மடக்கு முடிவதற்குச் சற்று முன்பாக (நெளிவு அடைப்புக்கு றிக்கு முன்பாக) கட்டுப்பாட்டு மாறியின் மதிப்பு, 1 மிகுக்கப்படுகிறது. while கூற்றின் தொடரமைப்பு:

```
/*  கட்டுப்பாட்டு மாறிகளைத் தொடங்கிவைத்தல்  */
while (நிபந்தனை)
{
   ................. ;  /*  கட்டளைகள்  */
   .................  ; ↗
   கட்டுப்பாட்டு மாறியைப் புதுப்பித்தல் ;
}
```

கட்டுப்பாட்டு மாறியின் மதிப்பு, while கூற்றின் நிபந்தனையோடு ஒப் பிட்டுப் பார்க்கப்படுகிறது. while மடக்கினை முறையாக முடித்துவைக்க மடக்குக்குள் கட்டுப்பாட்டு மாறி முறையாகப் புதுப்பிக்கப்படவேண்டும். i- ன் மதிப்பைப் புதுப்பிக்கும்.

i = i +1 ;

என்கிற வரி, விடுபட்டுப் போகுமாயின், கட்டுப்பாட்டு மாறியின் மதிப்பு எப்போதும் 1 ஆகவே இருக்கும். மடக்கு எப்போதும் முடிவுக்கு வராது.

மேற்கண்ட நிரலில் தொடக்க மதிப்பிருத்தல், நிபந்தனை, புதுப்பிக் கும் கட்டளை ஆகியவற்றை நீக்கிவிட்டுப் பார்த்தால், printf (“%d\n”, i) ; என்னும் ஒரேயொரு கட்டளை மட்டுமே மிஞ்சுகிறது. அதுவே, இந்த நிரலின் நோக்கத்தை நிறைவேற்றி வைக்கிறது.

பிறிதோர் எடுத்துக்காட்டு பார்ப்போம்.

1, 2, 4, 7, 11, 16........ என்ற எண் வரிசையில் 15 எண்களைக் காட்டுவ தற்கு ஒரு நிரலை எழுதிப் பார்ப்போம்.

எண் வரிசையில் இரண்டாவது எண், முந்தைய எண்ணோடு ஒன் றைக் கூட்டியபின் பெறப்படுகிறது. மூன்றாவது எண், முந்தைய எண் ணோடு இரண்டைக் கூட்டியபின் பெறப்படுகிறது. நான்காவது எண், முந்தைய எண்ணோடு மூன்றைக் கூட்டியபின் பெறப்படுகிறது. இவ் வாறாகப் பிற எண்களும் பெறப்படுகின்றன. இந்த மடக்கு 15 முறை நிறைவேற்றப்படவேண்டும். இதோ நிரல்:

```
#include <stdio.h>
main()
{
  int term;
  int i;
   i = 1;              /* செயலாக்க மாறி*/
   term = 1;            /*கட்டுப்பாட்டு மாறி*/
  while(term <=15)
   {
     printf(“%d\n”, i);
      i = i + term;
      term = term + 1;
   }
}
```

இந்த நிரலில், term என்னும் மாறி, மடக்கின் செயல்பாட்டைக் கட்டுப்படுத்தப் பயன்படுத்தப்பட்டுள்ளது. i என்னும் மாறி செயலாக்க மாறி யாகப் பயன்பட்டுள்ளது. இந்த நிரலில் அமைந்துள்ள தருக்கம் (logic), i=i+term என்னும் கட்டளையைச் சார்ந்துள்ளது. வரிசையில் முந்தைய எண்ணுடன் term -ன் மதிப்பைக் கூட்டி அடுத்த எண் பெறப்படுகிறது. ஒவ்வொரு முறை மடக்கு செயல்படும்போதும் term-ன் மதிப்பு 1 மிகுக்கப் படுகிறது. while மடக்கினைப் பயன்படுத்தியுள்ள இந்த நிரலின் வெற்றி, கட்டுப்பாட்டு மற்றும் செயலாக்க மாறிகளை முறையாகப் பயன்படுத்து வதையே சார்ந்திருக்கிறது. கட்டுப்பாட்டு மாறியில் தொடக்க மதிப்பி ருத்த வேண்டும்; நிபந்தனையில் பரிசோதிக்க வேண்டும், மடக்கினுள் புதுப்பிக்க வேண்டும். செயலாக்க மாறியும் தன் பங்கை ஆற்றியுள்ளது.

ஓர் while கூற்றினுள் இன்னோர் while கூற்றைப் பின்னலாய் அமைக்க முடியும். உள்ளிருக்கும் while மடக்கு வெளி while மடக்கைவிட வேக மாகச் செயல்படும். அதாவது வெளி while மடக்கு முடிவுக்கு வரும் முன்பே உள் while மடக்கு முடிவுக்கு வந்துவிடும். ஒவ்வோர் while மடக் குக்கும் கட்டுப்பாட்டு மாறிகளில் முறையாகத் தொடக்க மதிப்பிருத்த வேண்டும். மடக்குகள் திரும்பத் திரும்பச் செயல்படும் எண்ணிக்கை யைக் கட்டுப்படுத்த முறையான நிபந்தனையை அமைக்கவேண்டும். மடக்கினுள், கட்டுப்பாட்டு மாறிகள் முறைப்படி புதுப்பிக்கப்படவேண்டும்.

எடுத்துக்காட்டு:

ஒரு நிரல் எழுதுக: 1 ஒருமுறை, அடுத்த வரியில் 2 இருமுறை, அடுத்த வரியில் 3 மும்முறை... இவ்வாறாக 10 வரை திரையில் காட்ட வேண்டும். பத்தாவது வரியில் 10 பத்துமுறை இடம்பெற வேண்டும். இந்த நிரலுக்கு இரு மடக்குகள் தேவை. உள்-மடக்கு எத்தனை முறை செயல்படவேண்டும் என்பதை வெளிமடக்கு தீர்மானிக்கும். அதாவது, வெளி-மடக்கின் கட்டுப்பாட்டு மாறியின் மதிப்புக்கு ஏற்ப, ஒவ்வொரு முறையும் உள்-மடக்கு திரும்பச் செயல்படும் எண்ணிக்கை அமையும். நிரலை நோக்குக:

```
#include <stdio.h>
main()
{          int i, j;
            i = 1;
            while(i<=10)
            {
             j = 1;
             while(j <= i)
             {
                  printf("%d ", i);
                  j++;
             }
            printf("\n");
             i++;
             }
}
```

இந்த நிரலின் வெளியீடு இவ்வாறு அமையும்:

```
1
2 2
3 3 3
...................
10 10 10 10 10 10 10 10 10 10
```

இந்த நிரலில், ஒவ்வோர் எண்ணுக்கும் உள்-மடக்கு i தடவைகள் திரும்பவும் இயக்கப்படுகிறது. i என்பது வெளி-மடக்கின் கட்டுப்பாட்டு மாறியாகும்.

4.6.2 for கூற்று

சி-மொழியின் for மடக்கு, while கூற்றினை சுருக்கெழுத்து முறையில் எழுதுவதே ஆகும். எடுத்துக்காட்டாக, ஒரு சி-நிரலின் ஒரு பகுதியைக் காண்க:

```
x=1;
while (x<=10)
{
   printf("%d\n", x);
    x++;
}
```

இப்பகுதியை for மடக்குப் பயன்படுத்திக் கீழே உள்ளவாறு மாற்றி யமைக்கலாம்:

```
for(x=1; x<=10; x++)
{
  printf(“%d\n”, x);
}
```

while மடக்கில் ஒரு தொடக்க மதிப்பிருத்தல் (x=1), ஒரு பரிசோதனை (x<=10), ஒரு மதிப்பு மிகுப்பு (x++) ஆகிய மூன்று படிநிலைகள் இருப் பதை நோக்குக. இந்த மூன்று பகுதிகளையும் ஒரே வரியில் அமைத் துக்கொள்ள for மடக்கு இடம் கொடுக்கிறது. for மடக்கிலும், நுழையும் நிலையிலேயே நிபந்தனை பரிசோதிக்கப்படுகிறது. for மடக்கின் உடற் பகுதி 10முறை செயல்படுகிறது. இதில், கட்டுப்பாட்டு மாறியில் முதலில் தொடக்க மதிப்பு இருத்தப்படுகிறது. பிறகு அதன் மதிப்பு பரிசோதிக்கப் படுகிறது. நிபந்தனை ‘சரி’ எனில், மடக்கின் உடற்பகுதி செயல்படுத்தப் படுகிறது. இல்லையேல், மடக்கு முடித்துக்கொள்ளப்பட்டு, மடக்குக்கு அடுத்து இடம்பெற்றுள்ள கூற்றுகளை நிறைவேற்றத் தொடங்குகிறது.

மடக்கின் உடற்பகுதி செயல்படுத்தப்பட்டபின், நிரலின் கட்டுப்பாட்டு மீண்டும் for கூற்றுக்குள் வரும். கட்டுப்பாட்டு மாறி புதுப்பிக்கப்படும். அடுத்து, நிபந்தனை பரிசோதிக்கப்படும். நிபந்தனை ‘சரி’ ஆக இருக் கும்வரை மடக்கு திரும்பத் திரும்ப இயக்கப்படும்.

for மடக்கின் தொடரமைப்பு:

```
for (தொடக்க மதிப்பு இருத்தல்; நிபந்தனை; புதுப்பித்தல்)
{
        மடக்கின் உடற்பகுதி ;
}
```

for மடக்கையும் ஒன்றினுள் ஒன்றாய் அமைக்கலாம். 1 ஒரு முறை, 2 இருமுறை, 3 மும்முறை, ... எழுதும் நிரலை பின்னல் for மடக்கு பயன்படுத்தி எழுதிப் பார்ப்போம்:

```
#include <stdio.h>
main()
{
        int i;
        int j;

        for(i=1;i<=10;i++)
        {

        for(j=0;j<=i;j++)        ──┐
        printf("%d ", i);        ──┘──▶ உள்-மடக்கு

         printf("\n");
         }
}
```

உள் for மடக்கினுள் ஒரேயொரு கட்டளை உள்ளது. வெளி for மடக்கினுள் இரண்டு கட்டளைகள் உள்ளன. மடக்கின் உடற்பகுதியில் ஒரேயொரு கட்டளை மட்டுமே இருக்குமாயின், மடக்கின் உடற்பகு தியை நெளிவு அடைப்புக்குறிகளுக்குள் அமைக்க வேண்டியதில்லை.

while மடக்கு பயன்படுத்திச் செய்யப்படும் ஒரு பணியை for மடக்குப் பயன்படுத்தியும் செய்ய முடியும். எத்தனை முறை மடக்கினைச் செயல் படுத்தவேண்டும் என்கிற எண்ணிக்கை துல்லியமாக முன்பே தெரியு மெனில் அப்பணிக்கு for மடக்கினைப் பயன்படுத்துவது சிறந்த நிரலாக்க நடைமுறை ஆகும். எத்தனை முறை என்பதை மடக்கு தொடங்கும் முன்பே நிச்சயிக்க முடியாத சூழ்நிலைகளில் while மடக்கு உகந்தது.

எடுத்துக்காட்டு:

```
#include <stdio.h>
main()
{
   char ch;
    int count = 0;
    ch = getchar();
     while(ch != '\n')        /* நிபந்தனை*/

     {
            count++;
            ch = getchar();
      }

printf("the number of characters entered: %d\n", count);
}
```

மேற்கண்ட எடுத்துக்காட்டில், பயனர் விசைப்பலகையில் Enter விசையை அழுத்தும் வரை மடக்கு செயல்படும். அதுவரையில், எத்தனை முறை மடக்கு செயல்படவேண்டும் என்பதை முன்கூட்டியே கணிக்கமுடியாது. ch என்னும் மாறி, புதிய வரிக் குறியுருவை ('\n') விசைப் பலகை வழியாகப் பெற்றவுடன் (பயனர் Enter விசையை அழுத்தியவுடன்), நிபந்தனை 'தவறு' என்று ஆகும். நிரலின் கட்டுப்பாடு while மடக்கின் உடற்பகுதிக்கு அடுத்து இடம்பெற்றுள்ள printf() கூற்றுக்குச் சென்று விடும். இந்தப் பணிக்கு while மடக்கு உகந்தது. காரணம், பயனர் எத்தனை எழுத்துகளை உள்ளீடு செய்வார் என்பதைத் துல்லியமாக முன்பே கணிக்க முடியாது. இந்த நிரலில், விசைப்பலகை யிலிருந்து ஒரு நேரத்தில் ஓர் எழுத்தைப் பெறுவதற்கு getchar() என்னும் செயல்கூறு பயன்படுத்தப்பட்டுள்ளது. இது முன்-வரையறுக்கப்பட்ட செயல் கூறாகும்.

4.6.3 do-while கூற்று

while மடக்கில் நுழையும் நிலையிலேயே நிபந்தனை சரிபார்க்கப் படுகிறது. முதல் தடவையே நிபந்தனை 'தவறு' ஆகிப்போனால், மடக்கின் உடற்பகுதி ஒருமுறைகூடச் செயல்படுத்தப்படாது. do-while கூற்றில், வெளியேறும் நிலையில்தான் நிபந்தனை சரிபார்க்கப்படுகிறது. எனவே, நிபந்தனை 'சரி' அல்லது 'தவறு' எப்படி இருப்பினும், மடக்கின் உடற்

பகுதி குறைந்தது ஒரு முறையேனும் செயல்படுத்தப்படும். do-while மடக்கின் இறுதியில் நிபந்தனை சரிபார்க்கப்பட்டு, 'சரி' எனில் மடக்கு மீண்டும் ஒரு முறை செயல்படுத்தப்படும். நிபந்தனை 'சரி' ஆக இருக்கும் வரை இச்செயலாக்கம் தொடரும். நிபந்தனை 'தவறு' ஆனவுடன், மடக்கு முடிவுக்கு வரும். நிரலின் கட்டுப்பாடு, do-while கூற்றுக்கு அடுத்துள்ள கட்டளைக்கு மாற்றப்படும்.

எடுத்துக்காட்டு:

```
x = 14;
do
{
      y = x + 2;
      x—;
 } while (x > 0);
```

இந்த do-while கூற்றில், x-ன் தொடக்க மதிப்பு என்னவாக இருந்தாலும்,

y = x + 2 ;

x -- ;

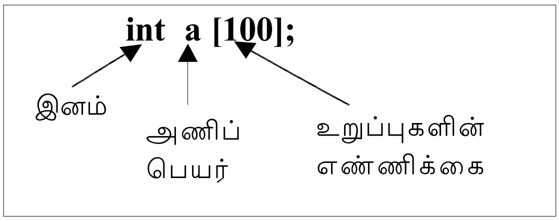
என்ற இரு கட்டளைகளும் முதல்முறை செயல்படுத்தப்படும். அதன்பிறகு x > 0 என்னும் நிபந்தனை மதிப்பிடப்படும். x, 0 -ஐவிடப் பெரிதாக இருப்பின், மடக்கின் உடற்பகுதி மீண்டும் செயல்படுத்தப்படும். பிறகு x > 0 என்னும் நிபந்தனை மீண்டும் மதிப்பிடப்படும். x-ன் மதிப்பு 0-ஐவிடப் பெரிதாக இருக்கும்வரை மடக்கின் உடற்பகுதி மீண்டும் மீண்டும் செயல்படுத்தப்படும். x-ன் மதிப்பு 0 ஆகும்போதோ, எதிர்ம (negative) எண் ஆகும்போதோ do-while கூற்றின் செயலாக்கம் நிறுத்தப்படும். ஆக, do-while மடக்கில் உடற்பகுதி குறைந்தது ஒரு முறையேனும் செயல்படுத்தப்படும்.

4.7 அணிகள் (Arrays)

ஒருபடித்தான உறுப்புகளின் அதாவது ஒரே தரவின உறுப்புகளின் தொகுப்பு 'அணி' எனப்படுகிறது. அணிகளைப் பயன்படுத்தினால் பல நேரங்களில் நிரலாக்கம் (programming) எளிதாகிப் போகும். எடுத்துக்காட்டாக, மூன்று முழுஎண்களில் பெரிய எண்ணைக் கண்டறிய வேண்டும். மூன்று தனித்தனி மாறிகளைப் பயன்படுத்த முடியும். எளிய if கூற்றுகளைப் பயன்படுத்தி எழுதப்பட்ட நிரலைக் கீழே காண்க:

```
#include <stdio.h>
main ()
{        int a, b, c;
         int max;
        printf("Enter the 3integers:");
         scanf("%d %d %d",&a,&b,&c);
         max = a;
          if(b > max)
            max = b;
         if(c > max)
            max = c;
         printf ("%d is the maximum ",max);
}
```

இந்த நிரலை, 100 எண்களில் பெரிய எண்ணைக் கண்டறியுமாறு நீட்டிப்போம் எனில், 100 தனித்தனி மாறிகளைப் பயன்படுத்தவேண்டும். நூற்றுக்கு மேற்பட்ட if- கூற்றுகளைப் பயன்படுத்த வேண்டியிருக்கும். இது நிரலை அதிகச் சிக்கலாக்கும். மிக நீண்டதாக்கும். மேலும், இது சரியான நிரலாக்க வழிமுறை ஆகாது. இந்தச் சிக்கலைத் தீர்க்க எளிய வழி 100 முழு எண்களுக்கான ஒரு அணியை அறிவிப்பதே:



ஒரு வரிசைக்குள் இருக்கும் 100 வெவ்வேறு எண்களை உரிய சுட்டு எண் (index) மூலம் அணுகமுடியும். 100 வெவ்வேறு மதிப்புகளை இருத்தி வைக்க 100 மாறிகளைப் பயன்படுத்த வேண்டிய சிக்கற்பாட்டைக் குறைக் கிறது. ஓர் அணி அறிவிப்பு, அணியின் பெயரைக் குறிப்பிடுகிறது; அதன் உறுப்புகளின் தரவினத்தைத் தெரிவிக்கிறது. சதுர அடைப்புக்குறி களுக்குள் ஒரு மாறிலிக் கோவையை பயன்படுத்த வேண்டும். அது உறுப்புகளின் எண்ணிக்கையைக் குறிக்கிறது. அதன் மதிப்பு எப்போதும் 0-ஐவிடப் பெரியதாக இருக்க வேண்டும். ஓர் அணியின் சேமிப்பிடம் என்பது அந்த அணியிலுள்ள அனைத்து உறுப்புகளையும் சேமிக்கத் தேவையான இடத்தைக் குறிக்கிறது. ஓர் அணியின் உறுப்புகள் முதல்

உறுப்பில் தொடங்கி கடைசி உறுப்பு வரை, நினைவகத்தில் அடுத்தடுத்த இருப்பிடங்களில் இருத்தப்படுகின்றன.

சி-மொழியில் அணி (array) என்பது ஒரு பரிமாணம் கொண்டதாக இருக்கலாம் அல்லது பலபரிமாணம் கொண்டதாகவும் இருக்கலாம். அணி யின் உறுப்புகளை சுட்டு எண்கள் மூலம் அணுக முடியும். 10 முழு எண் கள் கொண்ட ஓர் ஒருபரிமாண அணியை இவ்வாறு அறிவிக்கலாம்.

int a[10] /* அணி அறிவிப்புக் கூற்று */

நிரல் செயல்படத் தொடங்கும்பொழுது, நிரல்பெயர்ப்பி (Compiler) இந்த வரிசையின் 10 உறுப்புகளை இருத்திவைக்க முதன்மை நினைவ கத்தில் 20 பைட்டுகளை ஒதுக்கிவைக்கும். காரணம் ஒரு முழுஎண் ணுக்கு இரண்டு பைட்டு நினைவக இடம் தேவை. இந்த 10 உறுப்பு களும் அடுத்தடுத்துள்ள நினைவக இடங்களில் இருத்தப்படுகின்றன. அணி யின் சுட்டுஎண் 0-ல் தொடங்கும். ஓர் அணியில் n உறுப்புகள் இருக்கு மாயின் 0-ல் தொடங்கி n-1 வரையுள்ள சுட்டு எண்கள் மூலம் அவற்றை அணுகலாம். அணி உறுப்புகளில் கீழே உள்ளவாறு மதிப்பிருத்தப்படு கின்றன.

a [0] = 10;

a[1] = 20 ;

....

a[9] = 100;

அணியின் சுட்டு எண்ணிடல் பற்றிய சிறப்புச் செய்தி என்னவெனில், சுட்டு எண்களைக் கையாள ஒரு மடக்கினைப் பயன்படுத்த முடியும். எடுத்துக்காட்டாக, கீழேயுள்ள நிரல்பகுதி, அணியின் உறுப்புகளில் வரி சையாகத் தொடக்க மதிப்பு இருத்துகிறது.

```
int i;
int a[10];
for(i=0;i<10;i++)
        scanf("%d", &a[i]);
```

மேலேயுள்ள scanf() கூற்றில் உள்ள இரண்டாவது அளபுரு &a[i] என்பது, i = 0 முதல் 9 வரை இருக்கும் நிலையில் i-வது உறுப்பின் முகவரியைச் சுட்டுகிறது. scanf() செயல்கூறு விசைப்பலகையிலிருந்து மதிப்

பினைப் பெற்று, நினைவகத்தில் உரிய இடத்தில் (முகவரியில்) இருத்தி வைக்கிறது.

ஓர் அணியில் இப்படியும் தொடக்க மதிப்பிருத்தலாம்.

int a[3] = {10,15,20} ;

அணியின் உறுப்புகள் நினைவகத்தில் இவ்வாறு இருத்தப்பட்டிருக்கும்:

948	**950**	**952**	→ முகவரி
10	**15**	**20**	→ உறுப்புகளின் மதிப்புகள்
a[0]	**a[1]**	**a[2]**	→ உறுப்புகளை அணுகும் பெயர்கள்

ஒவ்வொரு முழுஎண்ணும் 2 பைட்டு நினைவகத்தை எடுத்துக் கொள்கிறது. நிரல்பெயர்ப்பி (Compiler) தொடர்ச்சியாக ஆறு பைட்டுகளை ஒதுக்கித் தருகிறது. இது நினைவகத் தொகுதி (block of memory) எனப் படுகிறது. இத்தொகுதியின் தொடக்க முகவரி படத்தில் கண்டுள்ளபடி 948. a[0], a[1], a[2] ஆகிய உறுப்புகளின் முகவரிகள் முறையே 948, 950, 952 ஆகும்.

முதல் உறுப்பின் முகவரி &a[0] என்று குறிப்பிடப்படுகிறது. நமது எடுத்துக்காட்டில் இந்த முகவரி 948 ஆகும். இந்த முகவரியில் பதிவாகி யுள்ள மதிப்பு 10. a[0] என்ற, உறுப்பின் பெயர் மூலமாக இந்த மதிப்பைப் பெற்றுக்கொள்ள முடியும். எனவே, a[0] -ன் மதிப்பு 10, முகவரி 948 எனக் கொள்ளவேண்டும். நிரல்பெயர்ப்பி, இந்த அணிக்குரிய நினை வகத் தொகுதியை ஒதுக்கீடு செய்தபிறகு, அத்தொகுதியின் தொடக்க முகவரியை (948), அணியின் பெயரிலேயே இருத்திவைக்கிறது. எனவே, a, &a[0] ஆகிய இரண்டுமே அணியின் தொடக்க முகவரியைக் குறிக்கின் றன. 'அணியின் பெயர் a, அணியைச் சுட்டுகிறது' எனச் சொல்லலாம். இதன்பொருள், a, தொடக்க முகவரியைச் சுட்டுகிறது என்பதாகும். வேறு வகையில் சொல்வதெனில், a, அணியின் முதல் உறுப்பினைச் சுட்டுகிறது எனலாம்.

	948	**950**	**952**
a ↗	**10**	**15**	**20**
	a[0]	**a[1]**	**a[2]**

நாம் ஏற்கெனவே அறிந்தபடி உள்நோக்கு செயற்குறியைப் பயன்படுத்தி நினைவக இடத்திலுள்ள மதிப்பைப் பெற்றுத்தர முடியும். எனவே, *a என்பது 10 என்னும் மதிப்பைப் பெற்றுத்தரும். மேலும்,

a[0]

***a**

***(&a[0])**

ஆகிய மூன்று கோவைகளுமே 10 என்கிற மதிப்பையே தரும் என்பதறிக. ஆக,

a[0] ⇔ *a ⇔ *(&a[0])

எனச் சொல்லலாம். இங்கே, ⇔ என்னும் குறியீடு, "எல்லாவகையிலும் இரண்டும் ஒன்றே" என்னும் பொருளைத் தருகிறது. இக்குறியீடு சி-மொழி செயற்குறி அன்று.

மேற்கண்ட விளக்கங்களிலிருந்து, ஓர் அணியின் தொடக்க முகவரி (அல்லது தள முகவரி) அந்த அணியின் பெயரிலேயே இருத்தப்படுகிறது என்பதைப் புரிந்து கொண்டிருப்பீர்கள். அணியின் பெயர், 'முகவரியைக்' கொண்டிருப்பதால் அது ஒரு சுட்டு (pointer) ஆகிறது. அந்தப் பெயர், அணியின் முதல் உறுப்பினைச் சுட்டிக்கொண்டிருக்கிறது. அணியின் பெயர், எப்போதும் வரிசையின் தொடக்க முகவரியைச் சுட்டிக் கொண்டிருக்கும் என்பதை மனதில் கொள்க. ஓர் அணியின் தள முகவரியை (base address) மாற்றியமைக்க முடியாது. அதாவது, அணியின் பெயர் இரண்டாவது உறுப்பைச் சுட்டிக்கொண்டிருக்குமாறு மாற்றியமைக்க முடியாது. அணியின் பெயரில் இருத்தப்பட்டுள்ள முகவரியை மாற்றியமைக்க முடியாது என்பதால் அது 'சுட்டு மாறிலி' (pointer constant) ஆகிறது.

அணிகளும் சுட்டுகளும் ஒன்றுக்கொன்று நெருக்கமான தொடர்புடையவை. x என்னும் சுட்டு மாறியை எடுத்துக்கொள்வோம்.

int *x;

ஏற்கெனவே விளக்கியபடி, மூன்று முழுஎண் உறுப்புகள் கொண்ட ஓர் அணியை எடுத்துக்கொள்வோம்.

int a[3] = {10, 15, 20};

இதில் x என்பது ஒரு சுட்டு மாறி (pointer variable). இது int இனம் என்பதால், இன்னோர் int இன மாறியின் முகவரியை ஏற்றுக்கொள்ளும். a என்பது, ஓர் int இன மதிப்பைச் சுட்டும். அதாவது, அணியின் முதல் உறுப்பினைச் சுட்டும் சுட்டு மாறிலி (pointer constant) அணியின் தள முகவரியை x என்னும் சுட்டு மாறியில் இருத்தி வைக்கமுடியும்.

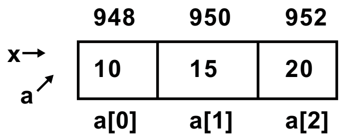
x = a ;

இப்போது, x என்னும் சுட்டு மாறி, அணியின் தொடக்க முகவரி யைச் சுட்டிக் கொண்டிருக்குமாறு செய்துள்ளோம். x என்பது (சுட்டு) மாறி (variable) என்பதால், நிரல் இயங்கிக்கொண்டிருக்கும் போதே அதன் மதிப்பை மாற்றியமைத்து, அணியில் வேறெந்த உறுப்பினையும் சுட்டு மாறு செய்ய முடியும். இத்தகைய சுட்டுக் கணக்கீடு (pointer arithmatic) சி-மொழியின் உன்னதமான சிறப்புக்கூறு ஆகும்.

சுட்டுகளைக் கொண்டு கீழ்க்காணும் செயல்பாடுகளை நிகழ்த்த முடியும்:

- ஒரு சுட்டின் மதிப்பில் ஒரு முழுஎண்ணைக் கூட்டவோ, கழிக் கவோ முடியும்.
- இரண்டு சுட்டுகளின் மதிப்புகளிடையே கழித்தல் செய்யமுடியும்.

கீழேயுள்ள விளக்கப் படம் காண்க:



x = a என்ற கூற்றின் காரணமாக, x, a ஆகிய இரண்டுமே அணியின் முதல் உறுப்பையே சுட்டிக்கொண்டுள்ளன. அதாவது, x, a இரண்டும் 948 என்னும் முகவரியைச் சுட்டுகின்றன. இரண்டிலும் 0-ஐக் கூட்டினால், x, a ஆகியவற்றின் முகவரி மதிப்புகளில் எவ்வித மாற்றமும் ஏற்படாது. உள் நோக்கு செயற்குறி * பயன்படுத்தி அந்த முகவரியிலுள்ள மதிப்பினைப் பெற்றுக்கொள்ள முடியும். எனவே, *(a+0), * (x+0) ஆகிய இரண்டுமே 10 என்னும் மதிப்பைப் பெற்றுத் தருகின்றன. *(a+0) என்னும் கோவை, அணி யில் முதல் உறுப்பின் மதிப்பை அணுகுவதற்கான குறிமானம் (notation) ஆகும். அணிசையில் முதல் உறுப்பின் மதிப்பை வழக்கமாக a[0] என்னும் அணிக் குறிமானம் கொண்டு பெற முடியும் என்பதை அறிவோம். கீழே யுள்ள நிகர்ப்பாடுகளைக் காண்க:

x + 0 ⇔ a + 0

***(x + 0)⇔ *(a + 0)**

***(a + 0), a [0]** ஆகிய இரண்டும் ஒன்றே என்பதால்,

*** (x + 0) ⇔ x [0]**

என்று எழுத முடியும். இதில், x என்னும் சுட்டு மாறியுடன் 0 என்னும் கீழ்ஒட்டு (subscript) அல்லது சுட்டு எண் (index) சேர்த்து, அது சுட்டிக் கொண்டிருக்கும் அணியின் முதல் உறுப்பை அணுகமுடிகிறது. ஒரு சுட்டினை சுட்டுவரிசை ஆக்க முடியும் (a pointer can be idexed) என்பதை இதன்மூலம் அறியலாம். ஒரு சுட்டு, தொடர்ச்சியான நினைவகத் தொகுதியைச் சுட்டிக் கொண்டிருக்கும்போதுதான் இது சாத்தியமாகும். x+0 என்பது முதல் உறுப்பின் முகவரி. *(x+0) என்பது அந்த முகவரியில் உள்ள முதல் உறுப்பின் மதிப்பு. அணியின் பெயர் a அல்லது சுட்டு மாறி x-உடன் 1-ஐக் கூட்டினால், இரண்டாவது உறுப்பின் முகவரி கிடைக்கும்.

x + 1 ⇔ a + 1

இங்கே, ஒரு முகவரியில் ஒரு முழுஎண் கூட்டப்படுகிறது. இது, சாதாரணக் கணக்கீடு அன்று, சுட்டுக் கணக்கீடு ஆகும். முக்கியமான இந்த விதிமுறையை நினைவில் கொள்க. "சுட்டுக் கணக்கீடுகளில், அச் சுட்டு சார்ந்த தரவினத்தின் அடிப்படையிலேயே அளவீடுகள் அமையும்". இங்கே x, int இன மதிப்பைச் சுட்டுகிறது. int இனம் 2 பைட்டுகளைக் கொண்டது. எனவே அளவீட்டுக் காரணி (scale factor) 2 ஆகும். ஆக,

x+1 ⇔ 948 + 1 * 2 = 950

a+1 ⇔ 948 + 1 * 2 = 950 ⇔ &a[1]

....

....

a + i ⇔ &a[i] ⇔ x + i ⇔ &x[i]

a+i என்பது அணியில் i- வது உறுப்பின் முகவரியைக் குறிக்கிறது. *(a+i), i-வது உறுப்பின் மதிப்பைக் குறிக்கிறது.

மேற்கண்ட விளக்கங்களிலிருந்து நீங்கள் புரிந்துகொள்ள வேண்டியவை:

- அணிகளும் சுட்டுகளும் நெருக்கமான உறவு கொண்டவை.
- ஒரு சுட்டினை, ஓர் அணி போலவே பாவிக்க முடியும்.

எடுத்துக்காட்டு:

10 முழுஎண்கள் கொண்ட ஒரு அணிக்குரிய மதிப்புகளை விசைப் பலகையிலிருந்து ஏற்றுக்கொள்ள ஒரு நிரல் எழுதுக:

```
#include <stdio.h>
main()
{
   int i;
   int a[10];
   for(i=0;i<10;i++)
     scanf("%d", a+i);
}
```

scanf() செயல்கூறில், &a[i] என்று இருக்க வேண்டிய இடத்தில் a + i என்று பயன் படுத்தப்பட்டுள்ளதை நோக்குக. காரணம், இரண்டும் ஒன் றுக்கொன்று நிகரானவை என்பதை ஏற்கெனவே நிறுவியுள்ளோம். i = 0 ஆக இருக்கும்போது, a + i என்பது முதல் உறுப்பின் முகவரியைத் தரு கிறது. i = 1 ஆக இருக்கும்போது, இரண்டாவது உறுப்பின் முகவரியைத் தருகிறது. இவ்வாறாக, a + i மூலம் அனைத்து முகவரிகளையும் பெற முடிகிறது.

இதுவரையில் நாம் முழுஎண் அணிகளையும், சுட்டுகள் மூலம் அவற்றை எவ்வாறு கையாள்வது என்பதையும் பார்த்தோம். நமது அடுத்த நோக்கம் குறியுரு அணி ஆகும். 24 எழுத்துகள் (பெரும்பாலான சூழ் நிலைகளில் குறியுரு என்பது எழுத்தையே குறிக்கிறது) கொண்ட ஓர் அணியை அறிவிப்போம்.

```
char name [24];
```

இதில், name என்பது 24 எழுத்துகள் கொண்ட ஓர் அணியாகும். நினைவகத்தில் தொடர்ச்சியாக 24 பைட்டுகள் ஒதுக்கப்படும். தொடக்க முகவரி அணியின் பெயர் name- ல் இருத்தப்படும். சரம் (string) என்பது, இன்மக் குறியுரு (null character - '\0') வுடன் முடிவுறும் எழுத்துகளின் தொகுப்பு என வரையறுக்கலாம். எனவே, ஒரு சரத்தைக் கையாள, எழுத்துகளின் அணி தேவைப்படுகிறது. விசைப்பலகையிலிருந்து ஒரு சரத்தைப் பெறுவதற்கு,

```
scanf ("%s", name);
```

என்னும் கூற்றுப் பயன்படுகிறது.

இதில், ஒரு சரத்தைப் படிக்க %s என்னும் வடிவமைப்பு வரைய றுப்பு பயன்படுத்தப்பட்டுள்ளது. %s, சரத்திலிருக்கும் வெற்று இட வெளியை (blank space) வரம்பெல்லைக் குறியுருவாகக் (delimiting character) கருதிக்கொள்ளும். எனவே, நடுவில் வெற்று இடவெளி கொண்ட சரத்தைப் பெறுவதற்கு இதனைப் பயன்படுத்த இயலாது. மேற்கண்ட

கூற்றை இயக்கும்போது, பயனரின் உள்ளீட்டுக்காகக் காத்திருக்கும். பயனர் உள்ளிடும் அனைத்து எழுத்துகளும் name சுட்டிக் கொண்டிருக்கும் முக வரியில் தொடங்கி, வரிசையாக இருத்தப்படும். இறுதியில் '\0' என்னும் இன்மக் குறியுரு தானாகவே சேர்த்துக்கொள்ளப்படும். சரத்தைத் திரை யில் காட்ட,

printf ("%s", name);

என்ற கூற்றினை அமைக்கலாம்.

வடிவமைப்பு வரையறுப்புக் குறியுருவான '%s', name சுட்டும் தொடக்க முகவரி தொடங்கி, இன்மக் குறியுரு வரையுள்ள எழுத்துக ளைத் தொடர்ச்சியாகப் பெற்றுத் தரும்.

சரம் என்பது, எழுத்துகளின் தொகுதி (ஓர் அணி) என்பதால் அதன் தரவினம் char * ஆகும். சர மாறிலி (string constant) யுடன் தொடர்புடைய தரவினம் ஒரு சுட்டு (char *) என்பதால், அதன் மதிப்பு ஒரு முகவரியா கத்தான் இருக்க முடியும், சரத்தின் உள்ளடக்க மதிப்பாக இருக்க முடி யாது. ஆக, ஒரு சர மாறிலியின் மதிப்பு, அச்சரம் இருத்தப்பட்டுள்ள நினைவகத் தொகுதியின் தொடக்க முகவரியாகும்.

கீழேயுள்ள நிரல் பகுதியை நோக்குக:

```
if("rama" == "rama")
   printf("equal");
 else
    printf("not equal");
```

இதில், பிற மாறிலிகள் ஒப்பிடப்படுவதைப் போலவே நிகர்ச் செயற்குறி மூலம் இரண்டு சரங்கள் ஒப்பிடப்பட்டுள்ளன. சி-மொழி நிரல்பெயர்ப்பி (Compiler) இத்தொடரமைப்பை ஏற்றுக் கொள்ளும். இந்த நிரல்பகுதி not equal என்னும் விடையைத் தரும். இதில், முழுக்கவும் ஒரே மாதிரியான இரு சரங்கள் ஒப்பிடப்பட்டபோதும், இரண்டு சரங்களும் நினைவகத்தில் வெவ்வேறு இடங்களில் இருத்தப்பட்டுள்ளன என்பதை மறக்கக் கூடாது. ஒரு சர மாறிலியின் மதிப்பு, அச்சரத்தின் தொடக்க முகவரிதான், சரத்தின் உள்ளடக்கத்தைக் குறிக்காது என்பதை நாம் ஏற்கெனவே அறிவோம். எனவே, மேற்கண்ட ஒப்பீட்டில், நினைவக இடத்தின் முகவரிகளே ஒப் பிடப்பட்டுள்ளன. அவை ஒன்றானவை அல்ல. எனவேதான், மேற்கண்ட நிரல்பகுதி not equal என்னும் வெளியீட்டைத் தருகிறது.

குறிப்பிட்ட சரத்திலுள்ள எழுத்துகளை எண்ணிச் சொல்லும் ஒரு நிரலைக் காண்போம். string.h என்னும் தலைப்புக் கோப்பு, சரத்தைக்

கையாளும் செயல்கூறுகள் பலவற்றை உள்ளடக்கியுள்ளது. ஒரு சரத்தின் நீளத்தைக் கண்டறிய strlen() என்னும் செயல்கூறு பயன்படுகிறது. strlen() என்பது முன்-வரையறுக்கப்பட்ட செயல்கூறு. இதன் மாதிரி வடிவம் string.h கோப்பில் அறிவிக்கப்பட்டுள்ளது. strlen() செயல்கூறின் மாதிரி வடிவம்:

```
int strlen (char * );
```

strlen() செயல்கூறு, அளபுருவாக ஒரு சரத்தை ஏற்று, அதன் நீளத்தை ஒரு முழுஎண்ணாகத் திருப்பி அனுப்பும்.

```
#include <stdio.h>
#include <string.h>
main()
{       char name[24];
        int len;
        printf("enter a string: ");
        scanf("%s", name);
        len = strlen(name);
        printf("%d\n", len);
}
```

இந்த நிரல் விசைப்பலகையிலிருந்து ஒரு சரத்தைப் பெற்று, அதன் நீளத்தைக் கண்டறிந்து விடையைத் திரையில் காட்டும்.

ஒரு செயல்கூறினுக்கு, ஓர் அணியை அளபுருவாக அனுப்பி வைக்க வேண்டியிருந்தால், சுட்டுகளைப் பயன்படுத்திக்கொள்ள முடியும். சரத்தின் நீளத்தை அறிய, நமது சொந்தச் செயல்கூறினை எழுதிப் பார்ப்போம். கீழேயுள்ள குறிமுறையில், lenstr() என்னும் செயல்கூறு, பயனர்-வரையறுத்த செயல்கூறு (user-defined function) ஆகும்.

வடிவம்– 1:

```
int lenstr(char *s)
{
      int count = 0;
      while(s[count] != '\0')
       count++;
      return(count);
}
```

"rama" என்னும் சரத்தை அளபுருவாக lenstr() செயல்கூறினுக்கு அனுப்பிவைத்தால், செயல்கூறின் முறையான அளபுருவால் (s) பெற்றுக் கொள்ளப்படும். அது ஒரு குறியுருச் சுட்டு (character pointer) ஆகும்.

எனவே, s என்பது "rama" என்னும் சரத்தைச் சுட்டிக்கொண்டிருக்கும். while மடக்கு count = 0-லிருந்து தொடங்குகிறது என்பதால், s[0] சரத்தின் முதல் எழுத்தைக் குறித்து நிற்கிறது. அதாவது r என்னும் எழுத்தைக் குறிக்கிறது. அது இன்மக்குறியுரு அன்று. while கூற்றின் நிபந்தனை 'சரி' என்றாகிறது. எனவே, while மடக்கின் உடற்பகுதி செயல்படுத்தப்படுகிறது. count -ன் மதிப்பு ஒன்று மிகுக்கப்படுகிறது. s[count] == '\0' என்றாகும் வரை, மடக்கு தொடர்ந்து செயல்படுத்தப்படும். இறுதியில் செயல்கூறு count- ன் மதிப்பைத் திருப்பி அனுப்பும். அதுவே, சரத்திலுள்ள எழுத்துகளின் எண்ணிக்கை ஆகும்.

வடிவம் – 2

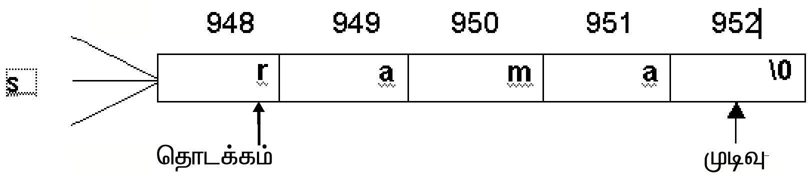
```
int lenstr(char *s)

{       int count = 0;
         while(*s != '\0')

        {
             count++;
             s++;
        }

         return(count);
}
```

நிரலின் கட்டுப்பாடு முதல் முறை மடக்கினுள் நுழையும்போது, *s- ல், "rama" என்ற சரத்தின் முதல் எழுத்தான r என்ற எழுத்து இருக்கும். while மடக்கின் உடற்பகுதியில், ஒவ்வொரு முறையும், சுட்டு மிகுக்கப்பட்டு, சரத்திலுள்ள அடுத்தடுத்த எழுத்துகளைச் சுட்டும். *s, '\0' ஆகும்வரை இந்தச் செயலாக்கம் தொடரும்.

வடிவம் – 3

```
int lenstr(char *s)
{
          char *start, *end;
          start = end = s;
          while(*end)
            end++;
          return(end – start);
      }
```

lenstr() செயல்கூறு இயக்கப்படும்போது, start, end, s ஆகிய அனைத்துக் குறியுருச் சுட்டுகளும் "rama" என்னும் சரத்தைச் சுட்டுமாறு அமைக்கப்பட்டுள்ளன. "rama" என்னும் சரம் அளபுருவாக அனுப்பிவைக்கப் படுகிறது.



start, end, s ஆகியவை, சரத்தின் முதல் எழுத்தின் முகவரியாகிய 948-ஐச் சுட்டுகிறது. end சுட்டினை '\0' ஐச் சுட்டும்வரை நகர்த்திச் செல்ல while மடக்கினைப் பயன்படுத்தியுள்ளோம். மடக்கு, செயல்பாட்டை முடிக்கும்போது, end சுட்டு 952 என்னும் முகவரியைச் சுட்டிக்கொண்டிருக்கும். while கூற்றின் நிபந்தனையை நோக்குங்கள். தொடக்கத்தில் *end -ல் r என்னும் மதிப்பு இருக்கும். அது 0 அல்ல என்பதால் நிபந்தனை 'சரி' என்றாகும். *end -ல் '\0' என்னும் மதிப்பு இருக்கும்போது, அது 0 என்பதால் நிபந்தனை 'தவறு' ஆகிவிடுகிறது. while மடக்கு முடிவுறும் போது, end சுட்டு 952 என்னும் முகவரியைச் சுட்டிக்கொண்டிருக்கும். start எப்போதும் 948 என்னும் முகவரியையே சுட்டிக்கொண்டிருக்கும். end-start (952-948) என்னும் கோவை 4 என்ற விடையைத் தரும். இந்த மதிப்பு தரப்பட்ட சரத்தில் உள்ள எழுத்துகளின் எண்ணிக்கையாகத் திருப்பி அனுப்பப்படும். ஒரு சுட்டிலிருந்து இன்னொரு சுட்டினைக் கழிக்கும்போது, இரண்டு சுட்டுகளுக்கு இடையேயுள்ள பைட்டுகளின் எண்ணிக்கை விடையாகக் கிடைக்கும். சுட்டுகளைக் கொண்டு அணிகளைக் கையாள்வதை, சுட்டுக் கணக்கீடு என்னும் பண்புக்கூறே சாத்தியமாக்கியுள்ளது என்பதைக் கவனத்தில் கொள்க.

4.7.1 பலபரிமாண அணிகள் (Multi dimensional Arrays)

சி-மொழியில் பலபரிமாண அணி என்பது, அணிகளின் அணியாகக் கருதப்படுகிறது. இருபரிமாண அணி இவ்வாறு அறிவிக்கப்படுகிறது:

int a[3][3];

இந்த அறிவிப்பு, 3x3 அணிக் கோவையை (matrix) உருவகிக்கிறது. இதில் 9 உறுப்புகள் உள்ளன. நிரல்பெயர்ப்பி, இந்த அணிக்கோவையின் உறுப்புகளை இருத்திவைக்க, தொடர்ச்சியான 18 பைட்டுகளை ஒதுக்கி வைக்கிறது. முதல் பரிமாணம், கிடக்கைகளின் (rows) எண்ணிக்கையைக்

குறிக்கிறது. இரண்டாவது பரிமாணம் நெடுக்கைகளின் (columns) எண் ணிக்கையைக் குறிக்கிறது. இந்த அணிக்கோவைக்கு (இருபரிமாணம்) மதிப்புகளைப் பெற, நமக்கு இரண்டு சுட்டு வரிசை எண்கள் தேவை. ஒன்று கிடக்கைச் சுட்டு எண் (Row Index), மற்றது நெடுக்கைச் சுட்டு எண் (Column Index). சி-மொழியில் அணிச் சுட்டு எண் 0-வில் தொடங்கு கிறது. முதல் உறுப்பை a[0][0] மூலம் அணுகலாம். கீழேயுள்ள நிரல் பகுதியை நோக்குக:

```
int a[3[[3];
int i;
int j;
for(i=0;i<3;i++)
for(j=0;j<3;j++)
scanf("%d", &a[i][j]);
```

இந்த நிரல்பகுதி, அணிக் கோவைக்குரிய மதிப்புகளை கிடக்கை வாரியாகப் பெறுகிறது. வெளி for மடக்கில் i = 0 ஆக இருக்கும்போது, உள் - மடக்கு j = 0, 1, 2 ஆகிய மதிப்புகளுக்கு மூன்று முறை செயல் படுகிறது. எனவே, முதல் கிடக்கையின் மூன்று எண்ணைப் பெறுகிறது. இதுபோவே, i = 1, i = 2 ஆகிய மதிப்புகளுக்கும் உள்-மடக்கு செயல்படு கிறது. மதிப்புகளை உள்ளிடும்போது, எண்களுக்கிடையே ஒன்று அல் லது மேற்பட்ட வெற்று இடவெளி(கள்) விட்டு ஒன்பது எண்களையும் தொடர்ச்சியாக உள்ளிடவேண்டும்.

இந்த அணிக்கோவையில் மூன்று கிடக்கைகள் உள்ளன. ஒவ்வொரு கிடக்கையும் மூன்று எண்கள் கொண்ட ஓர் அணியைக் குறிக்கின்றன. இது, அணிகளின் அணி வடிவில் உள்ளது. முதல் பரிமாணம் மூன்று உறுப்புகளைக் கொண்டுள்ளது. ஒவ்வோர் உறுப்பும் மூன்று எண்களைக் கொண்ட ஓர் அணியைக் குறிக்கின்றன. ஏனெனில் இரண்டாவது பரி மாணமும் 3 என உள்ளது.

கீழேயுள்ள அணிக் கோவையை நோக்குக:

1	2	3
4	5	6
7	8	9

இதையே, அணிகளின் அணியாக, இவ்வாறு குறிப்பிடலாம்:

a[0] → {1, 2. 3}
a[1] → {4, 5, 6}
a[2] → {7, 8, 9}

முதல் பரிமாணம் 3 உறுப்புகள் கொண்ட ஓர் அணி. ஒவ்வோர் உறுப்பும் ஒரு முழுஎண் அணியாகும். எனவே ஒவ்வோர் உறுப்பும் ஒரு முழுஎண் சுட்டு (integer pointer) ஆகும். ஆக, int a[3] [3] என்னும் அறி விப்புக்கு இவ்வாறு பொருள் விளக்கம் தரலாம். முதல் பரிமாணம் 3 முழு எண் சுட்டுகளைக் கொண்ட ஓர் அணியாகும். இரண்டாவது பரி மாணம் 3 முழு எண்களைக் கொண்ட ஓர் அணியாகும். இரண்டாவது பரிமாணம் 3 முழு எண்களைக் கொண்ட முதல் கிடக்கையைச் சுட்டு கிறது; அது ஒரு முழுஎண் சுட்டு ஆகும். அதே வேளையில் a[0][0] என்பது அணிக் கோவையின் முதல் உறுப்பாகும். அதாவது, முதல் கிடக்கையின் முதல் உறுப்பு. a[1] என்பது மூன்று முழுஎண்கள் கொண்ட இரண்டாவது கிடக்கையைச் சுட்டுகிறது. சி-மொழி நிரல்பெயர்ப்பி, இரு பரிமாண அணியை இந்த முறையில்தான் கையாள்கிறது. ஆனால் நாமோ முதல் பரிமாணம் கிடக்கைகளையும், இரண்டாவது பரிமாணம் நெடுக்கை களையும் குறிக்கின்றன என்று கருதிக்கொள்கிறோம்.

3x3 பரிமாணமுள்ள அணிக்கோவைக்குரிய மதிப்புகளைப் பெறுவது எப்படி என்பதை நாமறிவோம். அடுத்து, அணிக்கோவையின் உறுப்பு களைக் கிடக்கை வாரியாகத் திரையில் காட்டும் நிரல்பகுதியை எழுதிப் பார்ப்போம்.

```
for(i = 0; i < 3; i++)
{
for(j = 0; j < 3; j++)
  printf("%d ", a[i][j]);
printf("\n");
}
```

3x3 பரிமாணம் கொண்ட இரண்டு அணிக்கோவைகளைக்கூட்டி விடையை மூன்றாவது அணியில் எழுதும், ‘அணிக்கோவைக்கூட்டல்’ (matrix Addition) நிரலைக் கீழே காண்க:

```
#include <stdio.h>
#include <conio.h>
main()
{
          int a[3][3], b[3][3], c[3][3];
          int i, j;
    /* read values for the input matrix a */
          for(i=0;i<3;i++)
            for(j=0;j<3;j++)
              scanf("%d", &a[i][j]);

    /* read values for the input matrix b */
          for(i=0;i<3;i++)
            for(j=0;j<3;j++)
              scanf("%d", &b[i][j]);
    /* initialize the output matrix c with all elements 0 */
          for(i=0;i<3;i++)
            for(j=0;j<3;j++)
              c[i][j] = 0;
    /* add matrix a and b and store the result in matrix c */
          for(i=0;i<3;i++)
            for(j=0;j<3;j++)
              c[i][j] = a[i][j] + b[i][j];
    /* print the resultant matrix c */
          for(i=0;i<3;i++)
          {
          for(j=0;j<3;j++)
            printf("%d  ",  c[i][j]);
          printf("\n");
          }
}
```

இந்த நிரலின் ஒவ்வொரு பகுதியையும் நன்கு புரிந்துகொள்ளும் பொருட்டு ஆங்காங்கே விளக்கவுரைகள் (comments) சேர்க்கப்பட்டுள் ளன. ஓர் அணிக்கோவையிலுள்ள கிடக்கைகளை தனித்தனியாகக் கையா ளும் முறையை விளக்கும் நிரலைக் கீழே காண்க. இந்த நிரல், கொடுக்கப் பட்ட 3x3 பரிமாணம் கொண்ட அணிக்கோவையில் ஒவ்வொரு கிடக்கை யிலும் உள்ள எண்களில் பெரும மதிப்பு (maximum value) எதுவெனக் கண்டறிகிறது.

```
#include <stdio.h>
#include <conio.h>
main()
{       int a[3][3], max[3];
        int maximum(int *);  /* declaration of user-defined function */
        int i, j;
        clrscr();            /* to clear the contents of the screen */
        /* read values for the input matrix a */
        for(i=0;i<3;i++)
            for(j=0;j<3;j++)
                scanf("%d", &a[i][j]);
/* find the maximum in each row and store in the array max */
         for(i = 0; i < 3; i++)
              max[i] = maximum(a[i]);
        /*  print the max array */
         for(i =0; i < 3; i++)
               printf("The maximum value of row %d = %d\n", i+1,
                        max[i] );
}
int maximum(int *x)
{              int mvalue;
               mvalue = x[0];
               for(i = 1; i < 3; i++)
                  if(x[i] > mvalue)
                     mvalue = x[i];
               return(mvalue);
}
```

இந்த நிரலில், 3 முழுஎண் உறுப்புகள் கொண்ட அணியில் பெரும மதிப்பைக் கண்டறிய ஒரு பயனர்-வரையறுத்த செயல்கூறு எழுதப்பட்டுள்ளது. அச்செயல்கூறின் பெயர் maximum() . அது, ஒரேயொரு int * இன அளபுருவை ஏற்கும். ஒரு முழுஎண் மதிப்பைத் திருப்பி அனுப்பும். அதுவே பெரும மதிப்பாகும். இந்தச் செயல்கூறு main() செயல்கூறில், ஒரு for மடக்கினுள் 3 முறை அழைக்கப்படுகிறது. ஒவ்வொரு முறை maximum() செயல்கூறு அழைக்கப்படும்போதும், மூன்று முழு எண்கள் கொண்ட ஒரு கிடக்கை அளபுருவாக அனுப்பிவைக்கப்படுகிறது. a[i], i =0,1,2 ஆகிய கிடக்கைகளே அளபுருக்கள் ஆகும். அழைக்கப்பட்ட செயல்கூறில், முழுஎண் சுட்டு x என்னும் முறையான அளபுருவில், ஒவ்வொரு கிடக்கையின் தொடக்க முகவரியும் பெறப்படுகிறது. ஏற்கெனவே நாம் அறிந்தபடி, x என்னும் சுட்டினை சுட்டுவரிசையாக்க (அது 3 உறுப்புகள் கொண்ட வரிசையைச் சுட்டுகிறது என்பதால்) முடியும். அதன்வழியே அணியின் உறுப்புகளை அணுக முடியும். ஒவ்வொரு கிடக்கையின் பெரும மதிப்பும், முறையே mx என்னும் அணியில் இருத்திவைக்கப்படுகின்றன. இவ்வாறாக, ஓர் அணிக்கோவையிலுள்ள ஒவ்வொரு கிடக்கையையும் அதன் தொடக்க முகவரியைப் பயன்படுத்தித் தனித்தனியே கையாளமுடியும் என்பதறிக.

4.8 கட்டுருக்கள் (Structures)

சி-மொழியில் கட்டுரு (structure) என்பது 'தருவிக்கப்பட்ட தரவு இனம்' (derived data type) ஆகும். பிற தரவின மாறிகளைக் கொண்டு உருவாக்கப்படுகின்றது. (பல மாறிகளை ஒன்றாகக் கட்டி உருவாக்கப்படுவதால் 'கட்டுரு' ஆயிற்று). பயனர்-வரையறுக்கும் தரவினங்களை உருவாக்கக் கட்டுருக்கள் பயன்படுகின்றன. பொதுவாக, ஒரு கோப்பினில் சேமிக்கப்படும் ஏடுகளை (record) வரையறுக்கக் கட்டுருக்கள் பயன்படுகின்றன. கோப்பு என்பது ஏடுகளின் தொகுப்பு. ஏடு என்பது தகவல் புலங்களின் (fields of information) தொகுப்பாகும்.

ஒரு மாணவர்-ஏடு கீழ்க்காணும் புலங்களைக் கொண்டிருக்கலாம்:

வரிசை எண், பெயர், வயது

இப்புலங்களின் மதிப்பு,

1001, ஆனந்த், 18

என இருக்கலாம்.

மேற்கண்ட ஏடு ஆனந்த் என்ற மாணவரைப் பற்றிய தகவல்

களைக் கொண்டுள்ளது. அவரது வரிசைஎண் 1001, வயது 18 ஆகும்.

'அணி' என்பது ஒரே தரவின உறுப்புகளின் தொகுப்பாகும். 'கட்டுரு' என்பது வெவ்வேறு தரவின உறுப்புகளின் தொகுப்பாகும். அணி என்பது ஒருபடித்தான (homegeneous) உறுப்புகளின் தொகுப்பு. கட்டுரு என்பது கதம்ப (hetrogeneous) உறுப்புகளின் தொகுப்பு.

மேற்கண்ட எடுத்துக்காட்டில் சொல்லப்பட்ட மாணவர் ஏட்டிலுள்ள புலங்கள் வெவ்வேறு தரவினங்களைச் சார்ந்தவை:

வரிசைஎண்	: முழுஎண் புலம்
பெயர்	: எழுத்துகளின் அணி
வயது	: முழுஎண் புலம்

கீழேயுள்ள கட்டுரு வரையறையை நோக்குக:

```
struct student
{
        int rollno;
        char name[24];
        int age;
};
```

struct என்பது சி-மொழியின் சிறப்புச் சொல். ஒரு கட்டுருவை வரையறுக்கப் பயன்படுகிறது. student என்னும் குறிப்பெயர் 'கட்டுரு ஒட்டு' (structure tag) அல்லது 'ஒட்டுப் பெயர்' (tag name) எனப்படுகிறது. மேற்கண்ட வரையறை, ஏட்டின் கட்டமைப்பு பற்றி, அதாவது எத்தனை புலங்களைக் கொண்டுள்ளது, அவை எந்தத் தரவினத்தைச் சேர்ந்தவை என்ற விவரங்களை நமக்குத் தெரிவிக்கிறது. கட்டுருவின் வரையறை ஏட்டின் வார்ப்புருவை (template) மட்டுமே வழங்குகிறது. அது ஏட்டின் வெறும் உருவரைவே (skeleton) ஆகும். கட்டுரு வரையறையில், நெளிவு அடைப்புக்குறிகளுக்குள் அறிவிக்கப்பட்டுள்ள மாறிகள் (புலங்கள்) கட்டுருவின் உறுப்புகளாகும் (members). ஒரு கட்டுரு உறுப்புகள் அனைத்தும் தனித்த (unique) பெயர்களைக் கொண்டிருக்க வேண்டும். கட்டுரு வரையறை அரைப்புள்ளியுடன் முடிவுறும். கட்டுரு வரையறை நினைவகத்தில் இடம் எதையும் ஒதுக்கீடு செய்யாது. அதாவது, கட்டுருவை வரையறுக்கும்போது அதற்கான நினைவகப் பகுதி ஒதுக்கப்பட மாட்டாது. எனவே, கட்டுரு வரையறைக்குள்ளே உறுப்புகளில் தொடக்க மதிப்பிருத்த முடியாது.

மேற்கண்ட எடுத்துக்காட்டில், கட்டுருவின் இனம் struct student என்பதாகும். இப்போது, struct student என்பது பயனர்-வரையறுத்த ஒரு புதிய தரவினம் ஆகிவிடும். கட்டுரு வரையறை ஒரு புதிய தரவினத்தை உருவாக்குகிறது. அவ்வினத்தில், மாறிகளை அறிவிக்க முடியும். பிற மாறிகளைப் போலவே கட்டுரு மாறிகளை அறிவிக்க முடியும்.

எடுத்துக்காட்டு:

```
struct student  x, y;
```

x, y ஆகியவை struct student இன மாறிகளாகும். ஒவ்வொரு மாறியும் கட்டுருவில் வரையறுத்தபடி மூன்று புலங்களைக் கொண்டுள்ளன. ஒவ்வொரு struct student இன மாறிக்கும் மொத்தம் 28 பைட்டுகள் ஒதுக்கப்படுகின்றன.

கட்டுருவை எப்போதும் ஒரு முழுதளாவிய உருபொருளாய் (global entity) வரையறை செய்வது நல்லது. காரணம், நிரலில் உள்ள அனைத்து செயல்கூறுகளுக்கும் அதன் வரையறை கிடைக்கப்பெறும்.

கட்டுரு மாறிகளையும் முழுதளாவிய மாறிகளாக அறிவிக்க முடியும். எடுத்துக்காட்டாக,

```
struct student
{
        int rollno;
        char name[24];
        int age;
} x, y ;
```

இதில், கட்டுருவை வரையறுக்கும்போதே கட்டுரு மாறிகளும் அறிவிக்கப்பட்டுள்ளன. ஒட்டுப் பெயர் (tag name) இல்லாமலும் ஒரு கட்டுருவை வரையறுக்க முடியும். அப்படி வரையறுத்தால் அதற்கான மாறிகளை அப்போதே அறிவித்துவிடவேண்டும். ஒட்டுப் பெயர் இல்லையேல், கட்டுரு வரையறை பயனர்-வரையறுத்த ஒரு தரவினமாக ஏற்றுக் கொள்ளப்பட மாட்டாது.

எடுத்துக்காட்டு:

```
struct
{
        int empno;
        float salary;
} x, y ;
```

இந்த எடுத்துக்காட்டில் தரப்பட்டுள்ள கட்டுரு வரையறையின் அடிப் படையில், x,y தவிர வேறு புதிய கட்டுரு மாறிகளை உருவாக்க இய லாது. ஒட்டுப்பெயர் இல்லாமலும், புலங்கள் எதுவும் இல்லாமலும் வரை யறுக்கப்படும் கட்டுருக்கள் பயனற்றவை, அடையாளமற்றவை.

கீழேயுள்ள கட்டுரு வரையறையைக் கொண்டு எதுவும்செய்ய இயலாது:

```
struct
{
        int empno;
        float salary ;
} ;
```

4.8.1 கட்டுருவின் உறுப்புகளை அணுகுதல்
(Accessing the members of the structure)

கட்டுருவின் உறுப்புகளை (புலங்கள்) அணுகுவதற்கு, புள்ளிச் செயற்குறி (dot operator) பயன்படுகிறது. புள்ளிச் செயற்குறியுடன் கட்டுரு வின் உறுப்புகளை அணுகும்போது, கட்டுருமாறி ஒரு பண்புர்ணத்தி (Qualifier) யாகப் பயன்படுத்தப்படுகிறது. எடுத்துக்காட்டாக, மாணவரின் வரிசை எண்ணை அணுக, அதாவது, கட்டுரு மாறி x மூலமாக, rollno புலத்தை அணுக, பயன்படுத்தப்படும் குறிமானம்:

```
x.rollno
```

புலங்களில் மதிப்பிருத்தலாம்; புலங்களின் மதிப்பைப் படித்தறிய லாம். x, y என்னும் மாணவர்களின் rollno புலத்தில் மதிப்பிருத்தும் கூற் றுகள்:

```
x.rollno = 1000;
y.rollno = 1001;
```

student ஏட்டின் உறுப்புகளுக்கு மதிப்பைப் பெறுவதற்கு scanf() செயல் கூறை இவ்வாறு அமைக்கலாம்:

scanf ("%d %s %d", &x.rollno, x.name, &x.age);

student என்னும் கட்டுருவில் name என்னும் புலப்பெயர், 24 எழுத்துகள் கொண்ட ஒரு வரிசையின் தொடக்க முகவரியைக் குறிக்கிறது. எனவே, scanf() செயல்கூறில், மாணவர் x-ன் பெயரைப் பெற்றுக் கொள்ள x.name என்று மட்டும் குறிப்பிட்டால் போதும். வெளிப்படையாக & செயற்குறியைப் பயன்படுத்தி, &x.name என்று அமைக்கவேண்டிய தேவையில்லை என்பதை மனதில் கொள்க.

4.8.2 கட்டுருக்களுக்கான சுட்டுகள் (Pointers to Structures)

மூலத் தரவினங்களுக்கான சுட்டுகளைப் போலவே கட்டுருக்களுக்கான சுட்டுகளையும் அறிவிக்க முடியும். எடுத்துக்காட்டாக,

struct student *ptr;

இந்த அறிவிப்புக் கூற்றில், ptr என்பது பயனர்-வரையறுத்த தரவினமான struct student -ஐச் சுட்டுகின்ற ஒரு சுட்டாகும். அதே இனத்தைச் சார்ந்த வேறொரு கட்டுரு மாறியின் முகவரியை ptr- இல் இருத்தமுடியும்.

எடுத்துக்காட்டு:

struct student s1;

ptr = &s1 ; /* ptr எனும் சுட்டு கட்டுரு s1- ஐச் சுட்டுகிறது */

ptr மூலமாக, கட்டுரு s1-ன் உறுப்புகளை அணுக, புள்ளிச் செயற்குறிக்குப் பதிலாக அம்புச் செயற்குறியைப் (->) பயன்படுத்த வேண்டும்.

அதாவது, rollno புலத்தை அணுக ptr->rollno என்னும் கோவையைப் பயன்படுத்தவேண்டும். (*ptr).rollno என்பது பழைய தொடரமைப்பாகும்.

4.8.3 கட்டுருக்களின் அணி (An array of structures)

கட்டுருக்களின் அணியை இவ்வாறு அறிவிக்கலாம்:

struct student x[5] ;

இதில், x என்பது, ஐந்து கட்டுரு உறுப்புகளின் அணியாகும். x[0], x[1], ...x[4] ஆகியவை, struct student இனக் கட்டுருவின் தனித்தனி உறுப்புகளாகும். இவற்றின் புலங்களை, இவ்வாறு அணுகலாம்:

x[0].rollno, x[0].name, x[0].age

அணியில் உள்ள இரண்டாவது உறுப்பின் புலங்களுக்கு மதிப்பு களைப் பெற்றுக்கொள்ள,

scanf ("%d %s %d",& x [1] -rollno, x [1].name, &x [1].age);

5 மாணவர்களின் ஏடுகளுக்குரிய விவரங்களைப் பெறுவதற்கு, for மடக்கினைப் பயன்படுத்தலாம்.

```
for (i=0; i<5; i++)
    scanf ("%d %s %d", &x[i].rollno, x[i].name, &x[i].age);
```

இதேபோல, மாணவர்களின் ஏடுகளை ஒன்றன்பின் ஒன்றாகத் திரை யில் காட்டுவதற்கும் for மடக்கினைப் பயன்படுத்த முடியும். ஏடுகளை வரிசைஎண் வாரியாக, பெயர் வாரியாக அல்லது வயதுவாரியாகச் சேமிக்க முடியும். தனித்தனி அறிக்கைகளைத் தயாரிக்க முடியும். ஏடுகளைக் கட்டுரு வடிவில் கையாள முடியும் என்பதால், தரவுத்தளச் செயல்பாடு களுக்குப் பெரும்பாலும் கட்டுருக்கள் பயன்படுத்தப்படுகின்றன.

பயிற்சி வினாக்கள்

I. சரி அல்லது தவறு எனக் குறிப்பிடுக

1. ஒரு நிரலை எழுதுவதற்கு முன்பாகப் பாய்வுப்படம் வரைவது கட்டாயமாகும்.
2. போலிக் குறிமுறைகளைவிடப் பாய்வுப்படங்கள் எளிதானவை.
3. தீர்வுநெறி (Algorithm) என்பது வரம்புறு எண்ணிக்கையிலான படி நிலைகளைக் கொண்டிருக்க வேண்டும்.
4. நிச்சயித்திடாத பன்முறைச் செயலில் (Indefinite Iteration) சுட்டு எண் மாறி (Index variable) பயன்படுத்தப்படுகிறது.
5. நிச்சயித்திடாத பன்முறைச் செயலில் ஒவ்வொரு முறையும் நிபந் தனை சரிபார்க்கப்படுகிறது.

II புள்ளியிட்ட இடங்களை நிரப்புக

1. நிரல் எழுதுவதற்கு ____________ பாய்வுப்படம் வரையப்படு கிறது.
2. போலிக் குறிமுறையைப் புரிந்து கொள்வது, பாய்வுப்படத்தைப் புரிந்து கொள்வதைவிட ____________.
3. ஒவ்வொரு நிரலையும் ஒரு பாய்வுப்படத்தால் உருவகிக்க ____________.
4. சரிபார்ப்பு (walkthrough) மூலம் வடிவமைப்பில் உள்ள அனைத்துப் பிழைகளையும் கண்டறிய ____________ .
5. தீர்வு நெறியில் (Algorithm) உள்ள ஒவ்வொரு படிநிலையும் ____________ அளவில் ____________யும், ____________யும் எடுத் துக் கொள்ளவேண்டும்.

III. பதில் வரைக

1. பாய்வுப்படத்துக்கும் போலிக் குறிமுறைக்கும் இடையே உள்ள இரண்டு வேறுபாடுகளைக் கூறுக.
2. பாய்வுப்படத்தில் பயன்படுத்தப்படும் பல்வேறு வகையான பெட்டி களை வரைந்து காட்டுக. ஒவ்வொன்றின் பயன்பாட்டையும் விளக்குக.
3. பலவழிக் கிளைபிரித்தலுக்கு இரண்டு எடுத்துக்காட்டுகள் தருக. போலிக் குறிமுறை எழுதிக் காட்டுக.
4. இருவழிக் கிளைபிரித்தலுக்கு இரண்டு எடுத்துக்காட்டுகள் தருக. பாய்வுப்படம் வரைந்து காட்டுக.
5. சுட்டுஎண் மாறிக்கு இரண்டு எடுத்துக்காட்டுகள் தந்து விளக்குக.
6. நிச்சயித்த பன்முறைச் செயலை இரண்டு எடுத்துக்காட்டுகளுடன் விளக்குக.
7. நிச்சயத்திடாத பன்முறைச் செயலை இரண்டு எடுத்துக்காட்டு களுடன் விளக்குக
8. நிச்சயித்த, நிச்சயத்திடாத பன்முறைச் செயல்களுக்குள்ள மூன்று வேறுபாடுகளைக் குறிப்பிடுக.

9. இருவழிக் கிளைபிரித்தலைவிடப் பலவழிக் கிளைபிரித்தல் மிகவும் இயல்பானது என்பதை விளக்க இரண்டு எடுத்துக்காட்டுகள் தருக.

10. போலிக் குறிமுறை மூலம் அடிப்படையான கட்டுப்பாட்டு அமைப்புகளை விளக்குக.

11. தீர்வுநெறி (Algorithm) யின் பண்புக்கூறுகள் எவை?

IV. நிரலாக்கப் பயிற்சிகள்: ஒவ்வொரு நிரலுக்கும்,பொருத்தமான பாய்வுப்படம் வரைந்து, போலிக் குறிமுறையும் எழுதுக.

1. திரையகத்தில் உங்கள் பெயரை 5 முறை காட்ட ஒரு சி-நிரல் எழுதுக

2. இரண்டு மாறிகளின் மதிப்புகளை இடம் மாற்ற ஒரு சி-நிரல் எழுதுக

3. கீழ்க்காணும் பணிகளைச் செய்ய சி-நிரல்கள் எழுதுக:

   (i) ஒரு முக்கோணத்தின் பரப்பளவைக் கண்டறிய

   (ii) வெப்பநிலையை ஃபாரன்ஹீட்டிலிருந்து செல்சியஸுக்கு மாற்ற

   (iii) மணி: நிமிடம்: வினாடி என இருக்கும் நேரத்தை வினாடிகளில் மாற்ற

4. இரண்டு முழு எண்களில் பெரிய எண்ணைக் கண்டறிய சி-நிரல் எழுதுக

   (i) if கூற்று பயன்படுத்தி

   (ii) if கூற்று பயன்படுத்தாமல்
   [உதவிக்குறிப்பு : max =((a+b)+abs(a-b)/2]

5. இன்றைய தேதியில் உங்கள் வயதை இத்தனை ஆண்டுகள், மாதங்கள், நாட்கள் எனக் கணக்கிட்டுச் சொல்ல ஒரு சி-நிரல் எழுதுக. (எளிமை கருதி ஒவ்வொரு மாதத்துக்கும் 30 நாட்கள் என வைத்துக்கொள்க).

6. முதல் பத்து இயல்பெண்களின் கூட்டுத் தொகையைக் கண்டறிய அதாவது 1+2+3+..... +10 கண்டறிய சி-நிரல் எழுதுக.

7. ஃபைபோனாசி வரிசையில் (Fibonacci series) முதல் 15 எண்களைக் கண்டறிய ஒரு சி-நிரல் எழுதுக.

8. உங்கள் பெயரில் உள்ள உயிரெழுத்துகளை எண்ணிச் சொல்லும் சி-நிரலை எழுதுக

9. கொடுக்கப்பட்ட இரு எண்களையும் வகுக்கக்கூடிய மீப்பெரு பொதுக் காரணியைக்(greatest common factor) கண்டறியும் சி-நிரலை எழுதுக.

(உதவிக்குறிப்பு: மீ.பொ.கா. கண்டறியக் கீழ்க்காணும் பயனர்-வரையறுத்த செயல்கூறினைப் பயன்படுத்துக:

```
int gcf(int first, int second)
{   int temp;
    while(second > 0)
    {
        temp = first % second;
        first = second;
        second = temp;
    }
    return (first);
}
```

10. மேற்கண்ட gcf() செயல்கூறைப் பயன்படுத்தி, கொடுக்கப்பட்ட பின்னல் எண்ணைச் சுருக்கும் சி - நிரலை எழுதுக. 16/64 என உள்ளீடு தரப்பட்டால் 1/4 என வெளியீடு அமையவேண்டும்.

11. 3025 என்ற எண்ணின் முதல் பாதியையும் (30), இரண்டாவது பாதியையும் (25) கூட்டி, இரண்டின் மடங்கு (Square) கணக்கிட்டால் அதே எண் (3025) விடையாகக் கிடைக்கும். அதாவது (30+25)2 = 3025. இதேபோல் அமைந்த அனைத்து நான்கு இலக்க எண்களையும் கண்டறிய ஒரு சி-நிரல் எழுதுக.

12. 'ஆடம் எண்' என்பதன் இலக்கணம்: ஓர் எண்ணின் இரண்டின் மடங்கைத் (square) திருப்பிப் போட்டால் வரும் எண், அந்த எண்ணைத் திருப்பிப் போட்டு வரும் எண்ணின் இரண்டின் மடங்குக்கு நிகரானது. எடுத்துக்காட்டாக, அந்த எண் 12 என்க. 12^2 =144. 12 ஐத் திருப்பிப் போட்டால் 21. 21^2 = 441 (144 ஐத் திருப்பிப் போட்டால் கிடைக்கும்)! 10 முதல் 100 வரையுள்ள ஆடம் எண்களைக் கண்டறிய சி-நிரல் எழுதுக.

13. கொடுக்கப்பட்ட எண், ஃபைபோனாசி வரிசையில் இடம்பெறும் எண்ணா என்பதைப் பரிசோதிக்க ஒரு சி-நிரல் எழுதுக.

14. சிறிய எழுத்தில் (lower case) அமைந்துள்ள சரத்தைப் பெரிய எழுத்தில் (upper case) மாற்றவும், பெரிய எழுத்துச் சரத்தைச் சிறிய எழுத்தில் மாற்றவும் ஒரு சி-நிரல் எழுதுக.

15. ஒரு பதின்ம எண்ணை (Decimal Number) அதற்கு ஈடான இரும (binary), எண்ம(octal), பதினறும (hexa decimal) எண்ணாக மாற்ற ஒரு சி-நிரல் எழுதுக.

16. ஓர் இரும எண்ணைப் பதின்ம எண்ணாக மாற்ற சி-நிரல் எழுதுக

17. ஒரு முழுநிறைவு (perfect) எண் என்பது, அவ்வெண் தவிர்த்து பிற வகுஎண்களின் (divisions) கூட்டுத் தொகைக்கு நிகராக இருக்கும். 6 ஒரு முழுநிறைவு எண் ஆகும். காரணம், 6 தவிர்த்த பிற வகு எண்களின் கூட்டுத் தொகை 1+2+3=6. 1 முதல் 10000 வரையுள்ள முழுநிறைவு எண்களைக் கண்டறிய ஒரு சி-நிரல் எழுதுக.

18. ஒரு மிகுநிறைவு (abundant) எண் என்பது, அவ்வெண் தவிர்த்த பிற வகு எண்களின் கூட்டுத் தொகையைவிடச் சிறிதாக இருக்கும். 12 ஒரு மிகுநிறைவு எண் ஆகும். காரணம் 12 தவிர்த்த, பிற வகு எண்களின் கூட்டுத் தொகை (1+2+3+4+6=16) 12ஐ விடப் பெரியதாகும். 1 முதல் 10000 வரையுள்ள மிகுநிறைவு எண்களைக் கண்டறிய ஒரு சி-நிரல் எழுதுக.

19. ஒரு குறைபடு (deficient) எண் என்பது, அவ்வெண் தவிர்த்த பிற வகு எண்களின்கூட்டுத் தொகையைவிடப் பெரிதாக இருக்கும். 9 ஒரு குறைபடு எண் ஆகும். காரணம் 9 தவிர்த்த பிற வகுஎண்களின் கூட்டுத் தொகை (1+3), 9ஐ விடச்சிறியதாகும். 1 முதல் 10000 வரையுள்ள குறைபடு எண்களைக் கண்டறிய ஒரு சி-நிரல் எழுதுக.

20. ஒரு சரத்திலுள்ள எழுத்துகளை முன்பின்னாக (reverse) மாற்ற சி-நிரல் எழுதுக. உள்ளீட்டுச் சரம் rama எனில் வெளியீட்டுச் சரம் amar என இருக்கவேண்டும்.

21. கொடுக்கப்பட்ட ஒரு சரம் பாலிண்ட்ரோம் சரமா (முன்பின்னாக எழுதினாலும் ஒன்றாகவே இருப்பது - விகடகவி, malayalam) என்பதைக் கண்டறிந்து சொல்ல ஒரு சி-நிரல் எழுதுக.

22. ஒரு வரிசையிலுள்ள 10 எண்களைக் குறைந்து செல்லும் வரிசை முறையில் ஒழுங்கமைக்க சி-நிரல் எழுதுக.

23. ஒரு செயல்கூறைப் பயன்படுத்தி இரண்டு எண்களின் மதிப்புகளை இடம்மாற்றி அமைக்க ஒரு சி-நிரல் எழுதுக.

24. 10 எண்களின் சராசரியைக் கண்டறிய ஒரு சி-நிரல் எழுதுக.

25. a என்னும் ஓர் அணி n எண்களைக் கொண்டுள்ளது. எண்கள் ஏறுமுகமாக அமைந்திருக்க வேண்டும். ஏறுமுக அணியைப் பரிசோதித்து, ஏதேனும் ஓர் எண் வரிசைமாறிப் பிழையாக இடம் பெற்றிருப்பின் ஒரு பிழைசுட்டும் செய்தியைக் காட்டவேண்டும். வரிசைப்படி அமைந்திருப்பின் அதற்கான செய்தியைக் காட்டுக.

26. ஓர் அணிக் கோவையில் (Matrix) மூலைவிட்ட (diagonal) உறுப்பு களின் கூட்டுத் தொகையைக் கண்டறிய ஒரு சி-நிரல் எழுதுக.

27. ஓர் அணியின் இடமாற்று அணிக் கோவையைப் (Transpose) பெற ஒரு சி-நிரல் எழுதுக.

28. அணிக் கோவைப் பெருக்கல் (Matrix Multiplication) செய்து விடை காண சி-நிரல் எழுதுக.

29. கொடுக்கப்பட்ட ஓர் அணிக் கோவையில் ஒவ்வொரு நெடுக் கையிலும் பெரும, சிறும மதிப்புகளைக் கண்டறிய ஒரு சி-நிரல் எழுதுக. பெரும, சிறும மதிப்புகளைக் கண்டறிய செயல்கூறுகள் எழுதுக. ஒவ்வொரு செயல்கூறும் ஒரு முழுஎண் சுட்டினை (integer pointer) அளபுருவாகக் கொண்டிருக்கவேண்டும். (உதவிக் குறிப்பு: ஒவ்வொரு நெடுக்கையும் ஓர் அணியாகச் செயல்கூறி னுக்கு அனுப்பிவைக்கப்பட வேண்டும். ஒவ்வொரு நெடுக்கையின் உறுப்புகளையும் ஒரு தற்காலிக வரிசையில் இருத்திவைத்து, செயல்கூறினுக்கு அனுப்பிவைக்கவும்).

30. 10 பெயர்களை ஒரு அணியில் சேமித்து வைத்து, ஒவ்வொரு பெயரையும் ஒரு வரியில் காட்டுமாறு, சி-நிரல் எழுதுக.