பாடம் 11

N1FHMH

C++ - ன் செயற்கூறுகள்

கற்றலின் நோக்கங்கள் :

இந்தப் பாடப்குதியைக் கற்றபின் மாணவர் அறிந்து கொள்வது.

- செயற்கூறுகளின் வரையறை மற்றும் செயற்கூறுகளின் பயன்களைப் பற்றி அறிந்து கொள்ளுதல்.
- முன்னரே வரையறுக்கப்பட்ட மற்றும் பயனர் வரையறுக்கும் செயற்கூறுகளை பற்றி அறிந்து கொள்ளுதல்.
- சிக்கல்களைத் தீர்க்க கணித செயற்கூறுகளைப் பயன்படுத்துதல்.
- சரம் மற்றும் குறியுறு செயற்கூறுகளைப் பயன்படுத்தி சரம் மற்றும் குறியுறு தரவுகளை கையாளுதல் .
- பெரிய சிக்கல்களை சிறு சிறு செயற்கூறுகளாக உருவாக்கி அவற்றை செயல்படுத்தும் நிரலாக்கம் (**Modular Programming**) பற்றி அறிந்து கொள்ளுதல்.
- செயலுருபுகளின் பங்குகள் மற்றும் செயலுருபுகளின் பல்வேறு வழிமுறைகளை ஒப்பிடுதல்.
- மாறிகள் மற்றும் செயற்கூறுகளின் வரையெல்லையைக் கண்டறிதல்.

11.1 முன்னுரை

ஒரு பெரிய நிரலை சிறிய துணை நிரலாக பிரிக்க முடியும். அவ்வாறு பிரிக்கப்படும் அத்தனை நிரல் செயற்கூறு என்று அழைக்கப்படுகிறது. ஒவ்வொரு செயற்கூறும் அதற்குரிய செயற்பாடுகளை செயல்படுத்தும். நிரலின் நீளத்தையும் மற்றும் சிக்கற்பாட்டையும் குறைக்கவும் நிரலை எளிதில் புரிந்து கொள்ளவும், பிழைகளைக் கண்டறிந்து திருத்தவும் செயற்கூறுகள் வழிவகுக்கிறது. தானமைவு செயற்கூறுகள் "**உள்ளமைந்த செயற்கூறுகள்** (built – in functions)" என்றும் பயனர் உருவாக்கிய செயற்கூறுகள் "**பயனர் வரையறுத்த செயற்கூறுகள்**" என்றும் அழைக்கப்படுகிறது.

- உள்ளமைந்த செயற்கூறுகள் – C++ மொழியின் பொது நூலகத்தில் (**Standard Library**) உள்ள செயற்கூறுகளாகும்.
- பயனர் வரையறுத்த செயற்கூறுகள் – பயனர் உருவாக்கிய செயற்கூறுகளாகும்.

11.2 செயற்கூறுகளின் தேவை

நிரலின் நீளத்தை மற்றும் சிக்கற்பாட்டை குறைப்பதற்கு செயற்கூறுகள் பயன்படுத்தப்படுகின்றது. நிரலர்கள் அவர்களுடைய சொந்த செயற்கூறுகளை எழுதியோ அல்லது நிலையான நூலகத்திலுள்ள செயற்கூறுகளைப் பயன்படுத்தியோ துணை நிரல்களை (Sub Programs) உருவாக்கலாம்.

1. பிரித்தலும் மற்றும் சேர்த்தலும் (Divide and Conquer)

- சிக்கலான நிரல்களை மேலாண்மை செய்ய அவற்றை துணை நிரல்களாகப் பிரிக்கலாம்.
- தனிப்பட்ட செயற்கூறுகளை உருவாக்குதல், பிழை கண்டறிதல் மற்றும் சோதனை செய்தல் போன்றவற்றில் நிரலர்கள் கவனம்

செலுத்தலாம்.

- பல நிரலர்கள் ஒரே நேரத்தில் பல்வேறு செயற்கூறுகளில் பணியாற்றலாம்.

2. மறுபயனாக்கம் (Reusability):

- நிரலில் உள்ள சில வரிகளை மீண்டும் மீண்டும் வெவ்வேறு சந்தர்ப்பங்களில் பயன்படுத்தலாம். ஒரு நிரலின் பல்வேறு நகல்களை நீக்க செயற்கூறுகள் பயன்படுகின்றன. மேலும் செயல்கூறுகள் நிரலை பராமரிக்கவும் அவற்றின் அளவை குறைக்கவும் பயன்படுகின்றது.
- சில செயற்கூறுகளை எத்தனை முறை வேண்டுமானாலும் பல்வேறு உள்ளீடுகளை கொண்டு அழைக்கலாம்.

11.3 செயற்கூறுகளின் வகைகள்

செயற்கூறுகளை இரு வகைகளாக பிரிக்கலாம்.

1. முன் வரையறுக்கப்பட்ட (அல்லது) உள்ளமைந்த (அல்லது) நூலக செயற்கூறுகள்.
2. பயனர் வரையறுக்கப்பட்ட செயற்கூறுகள்.

பல்வேறு செயற்பாட்டிற்கு உடனே பயன்படுத்தும் வகையில் C++ மொழியில் உயரிய சேகரிப்புகளாக பல செயற்கூறுகள் உள்ளன. **தலைப்பு கோப்புகளில்** இவ்வகை செயற்கூறுகளின் வரையறைகளை முன்னரே எழுதப்பட்டு, பிழை திருத்தி மற்றும் நிரல் பெயர்க்கப்பட்ட (Complied) அவற்றைத் தொகுத்து சேமிக்கப்பட்டுள்ளன. இவ்வாறு நம் தேவைக்கு உடனே உபயோகிக்கப்படுத்தப்படும் துணை நிரல்களை முன் **வரையறுக்கப்பட்ட செயற்கூறுகள்** அல்லது **உள்ளமைந்த செயற்கூறுகள்** என்றழைக்கப்படுகின்றன. குறிப்பிட்ட பணிக்கான புதிய செயற்கூறுகளை பயனர் தேவைகேற்ப உருவாக்கும் வசதிகள் C++ மொழியில் உள்ளது. அவ்வாறு உருவாக்கும் பணிக்கான பெயர் மற்றும் தரவுகளின் தேவை (செயலுருப்புகள்) போன்றவற்றை பயனரே தீர்மானிப்பதால் இவ்வகை செயற்கூறுகளை **பயனர் வரையறுத்த** செயற்கூறுகள் என்கின்றோம்.

11.4 C++ -ல் உள்ள தலைப்பு கோப்புகள் மற்றும் உள்ளமைந்த செயற்கூறுகள்

நூலக செயற்கூறுகளுக்கு தேவையான செயற்கூறு முன்மாதிரி மற்றும் வரையறுப்புகளை தலைப்புக் கோப்புகள் கொண்டுள்ளது. நூலக செயற்கூறுகளில் உபயோகிக்கக்கூடிய தரவின் வகைகள் மற்றும் மாறிலிகளை பற்றிய வரையறுப்புக்களும் தலைப்புக் கோப்பு கொண்டுள்ளது. தலைப்பு கோப்பின் விரிவாக்கம். h என்று அறியப்படும். ஒரு தலைப்புக்கோப்பில் பல உள்ளிணைந்த செயற்கூறுகளைப் பற்றிய வரையறுப்புகள் கொண்டிருக்கும்.

உதாரணமாக: **stdio.h** என்ற தலைப்பு கோப்பில் **உள்ளீட்டு / வெளியீட்டிற்கான** செயற்கூறுகளை பற்றி முன்னரே வரையறுக்கப்பட்டுள்ளது.

11.4.1 உள்ளீடு / வெளியீடு (stdio.h)

இந்த தலைப்பு கோப்பில் (உள்ளீடு / வெளியீடு) (I/O) செயற்கூறுகளான **getchar()**, **putchar()**, **gets()** மற்றும் **puts()** போன்றவற்றை முன்னரே வரையறுக்கப்பட்டுள்ளது.

11.4.1.1 getchar() மற்றும் putchar() செயற்கூறுகள்

முன்னரே வரையறுக்கப்பட்ட **getchar()** என்ற செயற்கூறினைப் பயன்படுத்தி விசைப்பலகையின் மூலம் ஒரு எழுத்தையும் உள்ளீடு செய்யலாம் மற்றும் **putchar()** என்ற செயற்கூறின் மூலம் அந்த எழுத்தையும் வெளியீடும் செய்யலாம்.

நிரல் 11.1 C++ மொழியில் ஒரு எழுத்தை உள்ளீடவும் மற்றும் வெளியிடுவதற்கான நிரல்

```
#include<iostream>
#include<stdio.h>
using namespace std;
int main()
{
    cout<<"\n Type a Character : ";
    char ch = getchar();
    cout << "\n The entered Character is: ";
    putchar(ch);
    return 0;
}
```

வெளியீடு:

```
        Type a Character : T
        The entered Character is: T
```

11.4.1.2 gets() மற்றும் puts() செயற்கூறுகள்

gets() செயற்கூறின் மூலம் standard உள்ளீடு மூலம் ஒரு சரத்தை உள்ளீடு செய்து அதை சரத்திற்கான மாறியில் சேமித்து வைக்கலாம். puts() செயற்கூறு, gets() செயற்கூறின் மூலம் உள்ளீடாக பெற்ற சரத்தை ஒரு புதிய வரியில் வெளியிடச் செய்யும்.

நிரல் 11.2 சரத்தை உள்ளீடு மற்றும் வெளியீடு செய்வதற்கான C++ நிரல் :

```
#include<iostream>
#include<stdio.h>
using namespace std;
int main()
{
        char str[50];
        cout<<"Enter a string : ";
        gets(str);
        cout<<"You entered: "<< puts(str);
        return(0);
}
```

வெளியீடு:

```
        Enter a string : Computer Science
        You entered: Computer Science
```

11.4.2 குறியுறு செயற்கூறுகள் (ctype.h)

இந்த தலைப்பு கோப்பில் குறியுறுக்கு தேவையான பல்வேறு செயல்பாடுகள் வரையறுக்கப்பட்டுள்ளது. இந்த தலைப்பு கோப்பில் உள்ள பல்வேறு குறியுறு செயற்கூறுகளைக் கீழே விரிவாக விளக்கப்பட்டுள்ளது. இவ்வகை செயற்கூறுகளை நிரலில் பயன்படுத்தும் போது **ctype.h** என்ற தலைப்பு கோப்பை நிரலில் சேர்க்கப்பட வேண்டும்.

11.4.2.1 isalnum()

இந்த செயற்கூற்றைப் பயன்படுத்தி ஒரு குறியுறுவை ஆங்கில எழுத்தா என்று கண்டறியலாம். உள்ளீடு எண் அல்லது எழுத்தாக இருந்தால் இந்த செயற்கூறு சுழி அல்லாத மதிப்பைத் திருப்பி அனுப்பும், அல்லது சுழியம் என்ற மதிப்பைத் திருப்பி அனுப்பும்.

பொது வடிவம் :

```
isalnum (char c)
```

எடுத்துக்காட்டு:

```
int r = isalnum(‘5’);

cout << isalnum('A') <<’\t’<<r;
```

கீழே கொடுக்கப்பட்டுள்ள கூற்றில், உள்ளீடு செய்யப்பட்ட குறியுறு ஒரு ஆங்கில எழுத்தாகவோ அல்லது எண்ணாகவோ இல்லை அதனால் n என்ற மாறியில் 0 (பூஜ்ஜியம்) என்ற மதிப்பை இருத்தும்.

```
char c = '$';

int n = isalnum(c);

cout<<c;
```

வெளியீடு

```
0
```

நிரல் 11.3

```
#include<iostream>
#include<stdio.h>
#include<ctype.h>
using namespace std;
int main()
{
      char ch;
      int r;
      cout<<"\n Type a Character :";
      ch = getchar();
      r = isalnum(ch);
      cout<<"\nThe Return Value of
      isalnum(ch) is :"<<r;
}
```

வெளியீடு-1:

```
      Type a Character :A
      The Return Value of isalnum(ch) is :1
```

வெளியீடு-2:

```
      Type a Character :?
      The Return Value of isalnum(ch) is : 0
```

11.4.2.2 isalpha()

isalpha() செயற்கூறு உள்ளீடு செய்யப்பட்ட குறியுறு ஆங்கில எழுத்தாக உள்ளதா அல்லது இல்லையா என்பதை சரிபார்க்கப் பயன்படுகிறது.

பொது வடிவம்:

isalpha(char c);

இந்த செயற்கூறு உள்ளீடு செய்யப்பட்ட குறியுறு ஆங்கில எழுத்தாக இருந்தால் 1 என்ற மதிப்பைத் திருப்பி அனுப்பும் அல்லது **0** என்ற மதிப்பைத் திருப்பி அனுப்பும். கீழே கொடுக்கப்பட்டுள்ள கூற்றில் குறியுறு எழுத்தாக இல்லையெனில், வெளியீடு **0** ஆக இருக்கும்.

int n = isalpha('3');

கீழே கொடுக்கப்பட்டுள்ள கூற்றில் குறியுறு எழுத்தாக உள்ளது. அதனால் இதன் வெளியீடு 1 – ஆக இருக்கும்.

cout << isalpha('a');

நிரல் 11.4

```
#include<iostream>
#include<stdio.h>
#include<ctype.h>
using namespace std;
int main()
{
      char ch;
      cout << "\n Enter a charater: ";
      ch = getchar();
      cout<<"\n The Return Value of
      isalpha(ch) is :" << isalpha(ch) ;
}
```

வெளியீடு -1:

```
      Enter a charater: A
      The Return Value of isalpha(ch) is :1
```

வெளியீடு - 2:

```
      Enter a charater: 7
      The Return Value of isalpha(ch) is :0
```

11.4.2.3 isdigit()

உள்ளீடு செய்யப்பட்டுள்ள குறியுறு எண்ணாக உள்ளதா அல்லது இல்லையா என்பதைச் சரிபார்க்க இந்த செயற்கூறு பயன்படுகிறது. உள்ளீடு செய்யப்பட்ட குறியுறு எண்ணாக இருந்தால் இந்த செயற்கூறு 1 என்ற மதிப்பைத் திருப்பி அனுப்பும் அல்லது 0 என்ற மதிப்பைத் திருப்பி அனுப்பும்.

பொது வடிவம்:

isdigit(char c);

நிரல் 11.5

```
using namespace std;
#include<iostream>
#include<ctype.h>
int main()
{
    char ch;
    cout << "\n Enter a Character: ";
    cin >> ch;
    cout<<"\n The Return Value of isdigit(ch)
    is :" << isdigit(ch) ;
}
```

வெளியீடு -1

```
    Enter a Character: 3
    The Return Value of isdigit(ch) is :1
```

வெளியீடு -2

```
    Enter a Character: A
    The Return Value of isdigit(ch) is :0
```

Return 0; தற்போது உள்ள நிரல்பெயர்ப்பில் இது கட்டாயம்மில்லை

11.4.2.4. islower()

இந்த செயற்கூறு உள்ளீடு செய்யப்பட்ட குறியுரு எழுத்து ஆங்கில சிறிய எழுத்தாக உள்ளதா அல்லது இல்லையா என்று சரிபார்க்கும். உள்ளீடு செய்யப்பட்ட குறியுரு எழுத்து ஆங்கில சிறிய எழுத்தாக இருந்தால் இந்த செயற்கூறு பூஜ்ஜியம் அல்லாத மதிப்பைத் திருப்பி அனுப்பும், இல்லையேல் 0 (பூஜ்ஜியம்) என்ற மதிப்பைத் திருப்பி அனுப்பும்.

பொது வடிவம்:

islower(char c);

கீழே கொடுக்கப்பட்டுள்ள கூற்றுகளை இயக்கிய பின், உள்ளீடு செய்யப்பட்டுள்ள குறியுரு எழுத்து சிறிய எழுத்தாக உள்ளதால் ‘n’ என்ற மாறியில் 1 என்ற மதிப்பை இருத்தும்.

எடுத்துக்காட்டு **char ch = 'n';**

int n = islower(ch);

ஆனால் கீழே கொடுக்கப்பட்டுள்ள கூற்றில், உள்ளீடு செய்யப்பட்டுள்ள குறியுரு எழுத்து ஆங்கில பெரிய எழுத்தாக உள்ளதால் n என்ற மாறியின் மதிப்பு 0 ஆக இருத்தும்.

int n = islower('P');

11.4.2.5 isupper()

உள்ளீடு செய்யப்பட்டுள்ள குறியுரு எழுத்து ஆங்கில பெரிய எழுத்தாக உள்ளதா என்று சரிபார்க்க இந்த செயற்கூறு பயன்படும். உள்ளீடு செய்யப்பட்ட குறியுரு எழுத்து ஆங்கில பெரிய எழுத்தெனில் இந்த செயற்கூறு 1 என்ற மதிப்பைத் திருப்பி அனுப்பும் அல்லது 0 என்ற மதிப்பைத் திருப்பி அனுப்பும். கீழே கொடுக்கப்பட்டுள்ள எடுத்துக்காட்டுகளில் n என்ற மாறியில் மதிப்பு 1 என்றும் M என்ற மாறியில் மதிப்பு 0 என்றும் இருத்தும்.

பொது வடிவம்:

isupper(char c);

int n=isupper(‘A’);

int m=isupper(‘a’);

11.4.2.6. toupper()

உள்ளீடு செய்யப்பட்டுள்ள குறியுரு எழுத்து ஆங்கில பெரிய எழுத்தாக மாற்ற இந்த செயற்கூறு பயன்படுகிறது. உள்ளீடு செய்யப்பட்ட குறியுரு எழுத்து ஆங்கில பெரிய எழுத்தாகவே இருந்தால், வெளியீடு அதே குறியுருவாக இருக்கும்.

பொது வடிவம்:

char toupper(char c);

கீழே கொடுக்கப்பட்ட கூற்று c என்ற மாறியில் 'K' என்ற மதிப்பிருத்தும்.

char c = toupper('k');

ஆனால், கீழே கொடுக்கப்பட்டுள்ள கூற்றின் வெளியீடு 'B' ஆகவே இருக்கும்.

cout <<toupper('B');

11.4.2.7. tolower()

உள்ளீடு செய்யப்பட்டுள்ள குறியுறு எழுத்தை ஆங்கில சிறிய எழுத்தாக மாற்ற இந்த செயற்கூறு பயன்படுகிறது. உள்ளீடு செய்யப்பட்ட குறியுறு எழுத்தை ஆங்கில சிறிய எழுத்தாக இந்த செயற்கூறு திருப்பி அனுப்பும். உள்ளீடு செய்யப்பட்ட குறியுறு ஆங்கில சிறிய எழுத்தாகவே இருந்தால், வெளியீடு அதே குறியுறுவாக இருக்கும்.

பொது வடிவம்:

char tolower(char c);

கீழே கொடுக்கப்பட்டுள்ள கூற்றில் c என்ற மாறியில் 'k' என்ற மதிப்பை இருத்தும்.

char c = tolower('K');

ஆனால், இந்த கூற்றில் வெளியீடு 'b' ஆகவே இருக்கும்.

cout <<tolower('b');

11.4.3 சரங்களை கையாளுதல் (string.h)

string.h (cstring) என்ற நூலக கோப்பில் குறியுறுகளின் அணியில் உள்ள சரங்களைக் கையாளுவதற்கு என பல்வேறு பொதுவான செயற்கூறுகள் உள்ளன. சரங்களைக் கையாளுவதற்கான செயற்கூறுகளைப் பயன்படுத்துவற்கு முன்பு string.h என்னும் தலைப்பு கோப்பை நிரலில் இணைத்து கொள்ள வேண்டும்.

11.4.3.1 strcpy() :

பொது வடிவம்:

strcpy (target string, source string);

strcpy() என்ற செயற்கூறு இலக்கு (target) மற்றும் மூலம் (source) என்ற இரண்டு செயலுருபுகளை எடுத்துக் கொள்ளும். இந்த செயற்கூறு மூலத்திலுள்ள சரங்களை இலக்கு சரத்தின் நினைவகத்தில் நகல் எடுக்கும். சரத்தின் இறுதியைக் குறிக்கும் வெற்று குறியுறு (\0)-வையும் இந்த செயற்கூறு நகல் எடுக்கும்.

நிரல் 11.6

```
#include <string.h>
#include <iostream>
using namespace std;
int main()
{
   char source[] = "Computer Science";
   char target[20]="target";
   cout<<"\n String in Source Before Copied
   :"<<source;
   cout<<"\n String in Target Before Copied
   :"<<target;
```

நிரல் 11.6

```
      strcpy(target,source);
      cout<<"\n String in Target After strcpy
 function Executed :"<<target;
      return 0;
 }
```

வெளியீடு:

```
    String in Source Before Copied
:Computer Science
    String in Target Before Copied :target
    String in Target After strcpy function
Executed :Computer Science
```

11.4.3.2 strlen()

strlen() என்ற செயற்கூறு மூல சரத்தை அதன் செயலுருபாக எடுத்துக் கொண்டு அதன் நீளத்தை திருப்பி அனுப்பும். வெற்று குறியுறுவை (\0) சரத்தின் நீள கணக்கீட்டில் எடுத்துக்கொள்ளாது.

பொது வடிவம்: strlen (string);

நிரல் 11.7

```
#include <string.h>
#include <iostream>
using namespace std;
int main()
{
    char source[ ] = "Computer Science";
    cout<<"\nGiven String is "<<source<<"
    its Length is "<<strlen(source);
    return 0;
}
```

வெளியீடு:

```
        Given String is Computer Science its
Length is 16
```

11.4.3.3 strcmp()

strcmp() என்ற செயற்கூறு string1 மற்றும் string2 என்ற இரண்டு அளபுருக்களை எடுத்துக் கொள்ளும். இந்த செயற்கூறு string1 மற்றும் string2 உள்ளடக்கத்தை அகர வரிசையில் ஒப்பீடு செய்யும்.

பொது வடிவம்: strcmp (string 1, string 2);

strcmp() செயற்கூறு திருப்பி அனுப்பும் மதிப்புகள்:

- string1-ல் உள்ள முதல் குறியுறுவின் மதிப்பு string2 –ல் உள்ள முதல் குறியுறுவின் மதிப்பை விட அதிகமாக இருந்தால் நேர்ம மதிப்பைத் (Positive value) திருப்பி அனுப்பும். (ASCII மதிப்புகளை ஒப்பிடும்.)
- string1-ல் உள்ள முதல் குறியுறுவின் மதிப்பு string2 –ல் உள்ள முதல் குறியுறுவின் மதிப்பை விட குறைவாக இருந்தால் எதிர்ம மதிப்பைத் (Negative value) திருப்பி அனுப்பும்.
- string1 மற்றும் string2 சமம் எனில் 0 என்ற மதிப்பைத் திருப்பி அனுப்பும்.

நிரல் 11.8

```
#include <string.h>
#include <iostream>
using namespace std;
int main()
{
    char string1[] = "Computer";
    char string2[] = "Science";
    int result;
    result = strcmp(string1,string2);
    if(result==0)
    {
    cout<<"String1 : "<<string1<<" and
    String2 : "<<string2 <<"Are Equal";
    }
    if (result<0)
    {
    cout<<"String1 :"<<string1<<" and
    String2 : "<<string2 <<" Are Not Equal";
    }
}
```

வெளியீடு:

```
    String1 : Computer and String2 : Science
    Are Not Equal
```

11.4.3.4 strcat()

strcat() என்ற செயற்கூறு இலக்கு மற்றும் மூலம் என்ற இரு செயலுருபுகளை எடுத்துக் கொள்ளும். இந்த செயற்கூறு மூல சரத்தின் நகலை இலக்கு சரத்தின் இறுதியில் இணைக்கும்.

பொது வடிவம்: strcat (target, source);

நிரல் 11.9

```
#include <string.h>
#include <iostream>
using namespace std;
int main()
{
    char target[50] = "Learning C++ is fun";
    char source[50] = " , easy and Very useful";
    strcat(target, source);
    cout << target ;
    return 0;
}
```

வெளியீடு:

```
    Learning C++ is fun , easy and Very useful
```

11.4.3.5 strupr()

strupr()-இந்த செயற்கூறு உள்ளீடாக கொடுக்கப்பட்டுள்ள சரத்தை ஆங்கில பெரிய எழுத்துக்களாக மாற்றும்.

பொது வடிவம்: strupr (string);

நிரல் 11.10

```
using namespace std;
#include<iostream>
#include<ctype.h>
#include<string.h>
int main()
{
    char str1[50];
    cout<<"\nType any string in Lower case :";
    gets(str1);
    cout<<"\n Converted the Source
    string “<<str1<<into Upper Case is
    "<<strupr(str1);
    return 0;
}
```

வெளியீடு:

```
    Type any string in Lower case:computer
science
Converted the Source string computer science
into Upper Case is COMPUTER SCIENCE
```

11.4.3.6 strlwr()

strlwr()-இந்த செயற்கூறு உள்ளீடாக கொடுக்கப்பட்டுள்ள சரத்தை ஆங்கில சிறிய எழுத்துக்களாக மாற்றும்.

பொது வடிவம்: strlwr(string);

நிரல் 11.11

```
using namespace std;
#include<iostream>
#include<ctype.h>
#include<string.h>
int main()
{
        char str1[50];
        cout<<"\nType any string in Upper case :";
        gets(str1);
        cout<<"\n Converted the Source string “<<str1<<into Lower Case is "<<strlwr(str1);
}
```

வெளியீடு:

```
        Type any string in Upper case: COMPUTER SCIENCE
Converted the Source string COMPUTER SCIENCE into lower Case is computer science
```

11.4.4 கணித செயற்கூறுகள் (math.h)

math.h என்ற தலைப்பு கோப்பு பெரும்பாலான அடிப்படை கணித செயற்கூறுகளை உள்ளடக்கி உள்ளது.

11.4.4.1 cos() செயற்கூறு

cos() செயற்கூறு ஒரு செயலுருபுவின் மதிப்பு ரேடியன்ஸ் ஆக(Radians) எடுத்துக்கொள்ளும். cos() செயற்கூறு திருப்பி அனுப்பும் மதிப்பின் அளவு [-1, 1] என்ற பரப்பில் இருக்கும். திருப்பி அனுப்பும் மதிப்பு double, float, அல்லது long double என்ற தரவினமாக இருக்கும்.

நிரல் 11.12

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
        double x = 0.5, result;
        result = cos(x);
        cout << "COS("<<x<<")= "<<result;
}
```

வெளியீடு:

```
        COS(0.5)= 0.877583
```

11.4.4.2 sqrt() function

sqrt() செயற்கூறு உள்ளீட்டப்பட்ட செயலுருபின் மதிப்பிற்கான வர்க்க மூலத்தைத் திருப்பி அனுப்பும். sqrt() செயற்கூறு ஒரே ஒரு எதிர்ம எண் அல்லாத செயலுருபை மட்டுமே ஏற்கும். ஒரு எதிர்ம எண்ணை செயலுருபாக கொடுத்தால் sqrt() செயற்கூறு, செயற்களப்பகுதி (domain)பிழையை அறிவிக்கும்.

நிரல் 11.13

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
   double x = 625, result;
   result = sqrt(x);
   cout << "sqrt("<<x<<") = "<<result;
   return 0;
}
```

வெளியீடு:

```
        sqrt(625) = 25
```

11.4.4.3 sin() செயற்கூறு

sin() என்ற செயற்கூறு ஒரே செயலுருபுவின் மதிப்பை ரேடியன்ஸில் ஏற்கும். sin() செயற்கூறு திருப்பி அனுப்பும் மதிப்பின் பரப்பு [-1, 1]. இந்த செயற்கூறு திருப்பி அனுப்பும் மதிப்பு double, float அல்லது long double என்ற தரவுவகையிலிருக்கும் .

11.4.4.4 pow() செயற்கூறு

pow() செயற்கூறு அடித்தள(base) செயலுருபின் மேல் அடுக்குக்குறி(exponent) மதிப்பைத் திருப்பி அனுப்பும். pow() செயற்கூறின் செயலுருபின் தரவுவகை long double- ஆக இருந்தால், திருப்பி அனுப்பும் தரவின் வகை long double ஆக இருக்கும். இல்லையெனில் திருப்பி அனுப்பும் தரவுவகை double – ஆக இருக்கும். pow() செயற்கூறு இரண்டு செயலுருபுகளை ஏற்கும்.

- **அடித்தளம்** – அடித்தள மதிப்பு
- **அடுக்குக்குறி** – அடித்தள மதிப்பின் அடுக்குக்குறி

நிரல் 11.14

```
#include <iostream>
#include <math.h>
using namespace std;
int main ()
{
    double base, exponent, result;
    base = 5;
    exponent = 4;
    result = pow(base, exponent);
    cout<<"pow("<<base<<"^"<<exponent<<")
    = " << result;
    double x = 25;
    result = sin(x);
    cout << "\nsin("<<x<<")= "<<result;
    return 0;
}
```

வெளியீடு:

```
    pow(5^4) = 625
    sin(25)= -0.132352
```

11.4.5 சீரற்ற (Random) எண்களை உருவாக்குதல்

C++ -ல் srand() செயற்கூறு rand() செயற்கூற்றைப் பயன்படுத்தி போலி சீரற்ற எண்களை உருவாக்கலாம். rand() செயற்கூறுக்கு 1 என்ற மதிப்பு கொடாநிலை மதிப்பாகும், இதனால் rand() செயற்கூற்றை srand() என்ற செயற்கூற்றுக்கு முன் அழைத்தால், rand() செயற்கூறு srand(1) என்ற மதிப்புடன் அழைக்கப்பட்டதாக செயல்படும். srand() செயற்கூறு குறியில்லா முழு எண்ணை அதன் அளபுருவாக எடுத்துக்கொள்ளும், இதையே rand() செயற்கூறு அதன் தொடக்க எண்ணாக ஏற்கும். இந்த செயற்கூறு <cstdlib> அல்லது <stdlib.h> என்ற தலைப்பு கோப்பில் வரையறுக்கப்பட்டுள்ளது.

நிரல் 11.15

```
#include<iostream>
#include<cstdlib.h>
using namespace std;
int main()
{
    int random = rand(); /* No srand() calls
before rand(), so seed = 1*/
    cout << "\nSeed = 1, Random number =
" << random;
    srand(10);
    /* Seed = 10 */
    random = rand();
    cout << "\n\nSeed = 10, Random
number = " << random;
    return 0;
}
```

வெளியீடு:

```
    Seed = 1, Random number = 41
    Seed = 10, Random number = 71
```

11.5 பயனர் வரையறுக்கப்பட்ட செயற்கூறுகள்

11.5.1 முன்னுரை

குறிப்பிட்ட பணிக்கான புதிய செயற்கூறுகளை பயனர் அவர் தேவைக்கேற்ப வரையறுக்கலாம். பயனர் உருவாக்கிய செயற்கூறுகளை **பயனர் வரையறுத்த செயற்கூறுகள்** என்றழைக்கப்படுகிறது. ஒரு செயற்கூறினுக்கு உள்ளீட்டு அளபுருக்களை வரையறுப்பது அவசியமில்லை, ஆனால் அவற்றை பயன்படுத்தினால் செயற்கூறை அழைக்கும்

போது செயலுருபுகளுக்கான மதிப்பை அனுப்பலாம். ஒரு செயற்கூறு மதிப்பை வெளியீடாக திருப்பி அனுப்ப வேண்டும் என்பதும் அவசியமில்லை. பொதுவான செயல்பாடுகளை ஒரே மறுபயனாக்க தொகுதியில் உறை பொதியாக்கம் செய்ய செயற்கூறுகள் பயன்படும். மேலும், இந்த செயற்கூற்றுக்கு அதன் செயல்பாட்டிற்கு ஏற்ப பெயரிடுவது மிகவும் ஏற்றதாகும்.

11.5.2 செயற்கூறை வரையறுப்பது

C++-ல், ஒரு செயற்கூறை நிரலில் ஏதேனும் ஒரு பகுதியில் அதைப் பயன்படுத்தும் முன் வரையறுக்கப்பட வேண்டும். செயற்கூறு வரையறுப்பதற்கான தொடரியல்:

```
திருப்பு அனுப்பும் தரவு வகை செயற்கூறின்_பெயர்
(அளபுருக்களின் பட்டியல்)
    {
            செயற்கூறின் உடற்பகுதி
    }
```

குறிப்பு:

1. திருப்பி அனுப்பும் தரவு வகை என்பது **C++** ஏற்கும் ஏதேனும் ஒரு ஏற்கதகு தரவின வகையில் அமைக்கலாம்.

2. செயற்கூறின் பெயர் என்பது பயனர் வரையறுத்த குறிப்பெயர்.

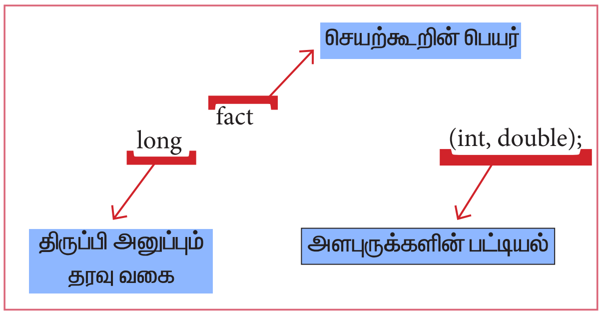
3. அளபுருக்களின் பட்டியல், கொடுப்பது அவசியமில்லை, இது மாறிகளின் பட்டியல். இதில் கொடுக்கப்பட்டுள்ள மாறியின் முன்பாக அதன் தரவினத்தை அறிவிக்கப்பட்டு, மாறிகள் காற்புள்ளியுடன் பிரிக்கப்பட்டிருக்க வேண்டும்.

4. செயற்கூறின் உடற்பகுதி- செயற்கூறின் உடல் பகுதியில் இந்த செயற்கூறு செயற்படுத்த வேண்டிய **C++** கூற்றுகளைக் கொண்டிருக்கும்.

11.5.3 செயற்கூறின் முன்வடிவு

C++ நிரல் பல செயற்கூறுகளை கொண்டிருக்கலாம். ஆனால், நிரல் செயற்பாட்டினைத் தொடங்க கண்டிப்பாக ஒரே ஒரு **main()** செயற்கூறு இருக்க வேண்டும். வரையறுக்கும் போது செயற்கூறுகளை நாம் அவற்றை நம் விருப்பப்படி எந்த வரிசையில் வேண்டுமானாலும் எழுதலாம். பயனர் வரையறுக்கும் செயற்கூறுகள் main() செயற்கூற்றின் முன் அல்லது பின் அமைக்கலாம். மாறியை அறிவிப்பது போன்று, ஒரு செயற்கூறையும் நிரலில் அதை பயன்படுத்தும் முன்னரே அறிவிக்கப்பட வேண்டும். செயற்கூறு அறிவிப்பு கூற்றை main() செயற்கூறின் வெளியே கொடுக்க வேண்டும்.

long fact (int, double)



படம் 11.1

மேலே கொடுக்கப்பட்டுள்ள செயற்கூறின் முன் வடிவம் கீழே கொடுக்கப்பட்டுள்ள தகவல்களை நிரல்பெயர்ப்பிக்கு கொடுக்கும்.

- செயற்கூறு திருப்பி அனுப்பும் மதிப்பு **long**
- செயற்கூறின் பெயர் **fact** என்பதாகும்.
- இந்த செயற்கூறு இரண்டு செயலுருபுக்களை ஏற்கும்.

int தரவினம் இதன் முதல் செயலுருபு.

double தரவினம் இதன் இரண்டாவது செயலுருபு.

int display(int , int) //செயற்கூறின் முன்வடிவம்//

மேலே கொடுக்கப்பட்டுள்ள செயற்கூறு முன்வடிவம் இதன் திருப்பி அனுப்பும் தரவினம், பெயர் மற்றும் முறையான அளபுருக்கள் அல்லது செயலுருபுக்கள் போன்ற தகவல்களை

அளிக்கிறது.

11.5.4 void கட்டளையின் பயன்

void தரவினம் இரண்டு முக்கிய நோக்கங்கள் கொண்டது:

- இந்த செயற்கூறு எந்த மதிப்பையும் திருப்பி அனுப்பாது என்பதைக் குறிக்க.
- பொது இனச் சுட்டியை (generic pointer) அறிவிக்க.

> குறிப்பு
>
> செயற்கூறு எந்த மதிப்பையும் திருப்பி அனுப்பாது என்பதை நிரல்பெயர்ப்பிக்கு **void** தரவினம் குறிக்கும், மேலும் விரிவான சூழலில் **void** எந்த மதிப்பையும் ஏற்காது.

எடுத்துக்காட்டாக:

void fun(void);

மேலே குறிப்பிட்டுள்ள செயற்கூறு முன்வடிவம் நிரல்பெயர்ப்பிக்கு fun() என்ற செயற்கூறு அதை அழைக்கும் நிரலிருந்து எந்த மதிப்பையும் ஏற்றுக் கொள்ளாது. மேலும் இது அதை அழைக்கும் நிரலுக்கு எந்த மதிப்பையும் திருப்பி அனுப்பாது.

11.5.5 செயற்கூற்றை செயல்படுத்துதல்

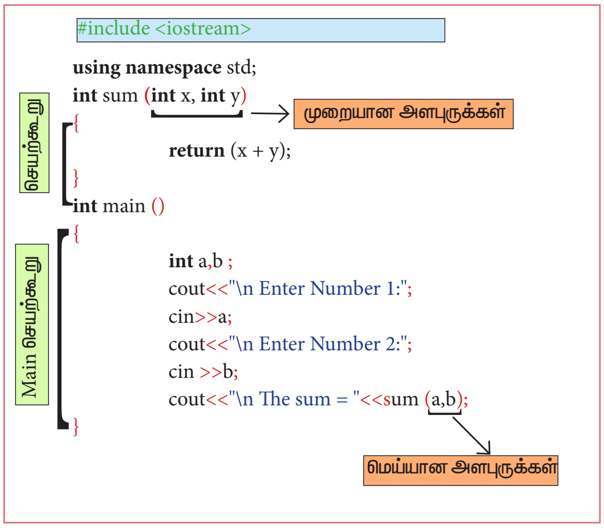
பயனர் வரையறுத்த செயற்கூறை செயல்படுத்த அதன் பெயர் மற்றும் தேவையான செயலுருபுக்கள் கொண்டு அழைக்க வேண்டும். செயற்கூறை அழைக்கப்படும் போது நிரல்பெயர்ப்பி செயற்கூறின் முன் வடிவத்தை பயன்படுத்தி செயற்கூறு சரியாக அழைக்கப்பட்டுள்ளதா என்று சரிபார்க்கும். முன்வடிவில் உள்ள செயலுருபின் தரவு வகையும் அழைப்புக் கூற்றில் உள்ள செயலுருபுக்களின் தரவு வகைகள் பொருத்தமாக இல்லையெனில், தரவு வகை மாற்றம் தானாகவே செய்ய முடியும் என்றால் நிரல்பெயர்ப்பி அதை செய்யும், இல்லையெனில் நிரல்பெயர்ப்பி இதற்கு பிழை அறிக்கை அறிவிக்கும்.

எடுத்துக்காட்டாக:

1	display()	செயலுருபு ஏற்காது மற்றும் எந்த மதிப்பையும் திருப்பி அனுப்பாத அழைப்பு செயற்கூறு.
2	display (x, y)	செயலுருபுக்களை ஏற்கும் மற்றும் எந்த மதிப்பையும் திருப்பி அனுப்பாத அழைப்பு செயற்கூறு.
3	x = display()	செயலுருபு ஏற்காது மற்றும் மதிப்பைத் திருப்பி அனுப்பும் அழைப்பு செயற்கூறு.
4	x = display (x, y)	செயலுருபுக்களை ஏற்கும் மற்றும் மதிப்பைத் திருப்பி அனுப்பும் அழைக்கும் செயற்கூறு.

14.5.5.1 முறையான அளபுருக்கள் (Formal Parameters) மற்றும் மெய்யான அளபுருக்கள் (Actual Parameters) அல்லது செயலுருபுக்கள் (Arguments)

செயலுருபுக்கள் அல்லது அளபுருக்கள் மூலமாக அழைக்கும் செயற்கூறிலிருந்து அழைக்கப்படும் செயற்கூறுக்கு மதிப்புகள் பரிமாற்றம் செய்யப்படும். வரையறுக்கப்பட்ட செயற்கூறில் மாறிகளாக பயன்படுத்தப்பட்டுள்ள அளபுருக்களை முறையான அளபுருக்கள் என்றழைக்கப்படும். அழைப்பு செயற்கூறில் உள்ள மாறிலிகள் அல்லது மாறிகள் அல்லது கோவைகளை மெய்யான அளபுருக்கள் என்றழைக்கப்படும்.



```
#include <iostream>
using namespace std;
int sum (int x, int y)
{
        return (x + y);
}
int main ()
{
        int a,b ;
        cout<<"\n Enter Number 1:";
        cin>>a;
        cout<<"\n Enter Number 2:";
        cin >>b;
        cout<<"\n The sum = "<<sum (a,b);
}
```

படம் 11.2 முறையான மற்றும் மெய்யான அளபுருக்கள்

11.5.5.2 முன்னியல்புச் செயலுருபுக்கள் (Default Argument)

C++ மொழியில் ஒரு செயற்கூற்றின் முன்வடிவில் உள்ள முறையான அளபுருக்களில் முன்னியல்பு மதிப்புகளை இருத்தி வைக்க முடியும். செயற்கூற்றை அழைக்கும் போது முன்னியல்பு செயலுருபு சில மதிப்புகளைத் தவிர்க்க வழிவகுக்கும். செயற்கூற்றை அழைக்கும் போது ஏதேனும் செயலுருபுகளுக்கு மதிப்பு கொடுக்காவிடில் நிரல்பெயர்ப்பி முன்னியல்பு செயலுருபுக்களின் மதிப்பைகளை அழைக்கப்பட்ட செயற்கூற்றிக்கு ஏற்கும்.

- மாறியில் தொடக்க மதிப்பிருந்தும் வடிவில் முன்னியல்பு மதிப்பு தரப்பட்டுள்ளது.

எடுத்துக்காட்டு :

void defaultvalue(int n1=10, n2=100);

- ஒரு செயற்கூறின் அழைப்புக் கூற்றில் சில செயலுருபுக்களை விட்டுவிடவோ அல்லது செயலுருபுக்கள் இல்லாமலே அழைக்கவோ முன்னியல்பு செயலுருபுக்கள் வழி வகுக்கின்றன.

எடுத்துக்காட்டு : **defaultvalue(x,y);**
defaultvalue(200,150);
defaultvalue(150);
defaultvalue(x,150);

- முன்னியல்பு மதிப்புகள் செயற்கூறு முன்வடிவில் செயலுருபுப் பட்டியலில் இறுதியாகவே இடம் பெற வேண்டும். பட்டியலில் தொடக்கத்திலோ, நடுவிலோ இடம் பெறக் கூடாது.

எடுத்துக்காட்டு :

- **void defaultvalue(int n1=10, n2);**
//தவறான முன்வடிவம்
- **void defaultvalue(int n1, n2 = 10);**
//சரியான முன்வடிவம்

11.5.5.3 மாறிலி செயலுருபுக்கள்

const சிறப்புச் சொல்லை பயன்படுத்தி மாறிலிகளை அறிவிக்கலாம். **const** சிறப்புச் சொல் ஒரு மாறியின் மதிப்பை மாறாத தன்மை கொண்டதாக ஆக்கும். மாறிலியின் மாறிகள் அறிவிக்கும் போதே அதனுடைய தொடக்க மதிப்புடன் அறிவிக்க வேண்டும். **const** என்னும் பண்புணர்த்தி **(modifier)** மூலம் ஒரு மாறியில் இருத்தப்படும் மதிப்பை செயற்கூற்றின் உடல் பகுதியில் மாற்றப்பட முடியாது.

தொடரியல் :

<return type> <function name> (const <datatype variable = value>)

எடுத்துக்காட்டு :

- int minimum(const int a=10);
- float area(const float pi=3.14, int r=5);

நிரல் 11.16

```
#include <iostream>
using namespace std;
double area(const double r,const double pi=3.14)
{
        return(pi*r*r);
}
int main ()
{
        double rad,res;
        cout<<"\nEnter Radius :";
        cin>>rad;
        res=area(rad);
        cout << "\nThe Area of Circle ="<<res;
        return 0;
}
```

வெளியீடு :

```
        Enter Radius :5
        The Area of Circle =78.5
```

“r” என்ற மாறியின் மதிப்பை r=25 என்று “area” செயற்கூறின் உடல் பகுதியில் மாற்றியமைத்தால், நிரல்பெயர்ப்பி “**assignment of read-only parameter 'r'**” என்ற பிழையை அறிவிக்கும்.

```
{
        r=25;
        return(pi*r*r);
}
```

11.6 செயற்கூற்றை அழைப்பதற்கான வழிமுறைகள்

C++ மொழியில் செயலுருபுக்களை செயற்கூறுக்கு இரு வழிகளில் அனுப்பலாம். செயலுருபுக்களை அனுப்பும் முறையைப் பொருத்து, செயற்கூற்றை அழைக்கும் வழிமுறைகள் மதிப்பு மூலம் அழைத்தல் (Call by Value)மற்றும் குறிப்பு மூலம் அல்லது முகவரி மூலம் அழைத்தல் (Call by Reference or Address) என்று இரு வகைப்படுத்தப்படும்.

11.6.1 மதிப்பு மூலம் அழைத்தல் முறை

இந்த முறையில் மெய்யான அளபுருவின் மதிப்பை முறையான அளபுருவில் நகலெடுக்கும். இந்த முறையில் முறையான அளபுருவின் மதிப்பில் ஏதேனும் மாற்றங்கள் செய்தால் அது மெய்யான அளபுருவின் மதிப்பில் பிரதிபலிப்பதில்லை.

நிரல் 11.17

```
#include<iostream>
using namespace std;
void display(int x)
{
        int a=x*x;
        cout<<"\n\nThe Value inside display function (a * a):"<<a;
}
int main()
{
          int a,b;
        cout<<”\nExample : Function call by value:”;
          cout<<"\n\nEnter the Value for A :";
          cin>>a;
        display(a);
        cout<<"\n\nThe Value inside main function "<<a;
        return(0);
}
```

வெளியீடு :

```
        Example : Function call by value
        Enter the Value for A : 5
        The Value inside display function (a * a) : 25
        The Value inside main function 5
```

11.6.2 குறிப்பு மூலம் அழைத்தல் முறை

இந்த முறையில் மெய்யான அளபுருவின் குறிப்பை அல்லது முகவரியை முறையான அளபுருவில் நகலெடுக்கும். முகவரியின் மூலம் அழைப்பதால் முறையான அளபுருவின் மதிப்பில் ஏதேனும் மாற்றம் செய்தால் மெய்யான அளபுருவில் அந்த மாற்றம் பிரதிபலிக்கும்.

நிரல் 11.18

```
#include<iostream>
using namespace std;
void display(int &x) //passing address of a//
{
         x=x*x;
         cout<<"\n\nThe Value inside display function (n1 x n1) :"<<x ;
 }
int main()
{
int n1;
cout<<"\nEnter the Value for N1 :";
cin>>n1;
cout<<"\nThe Value of N1 is inside main function Before passing : "<< n1;
display(n1);cout<<"\nThe Value of N1 is inside main function After passing (n1 x n1)  : "<< n1;
return(0);
}
```

வெளியீடு :

```
         Enter the Value for N1 :45
         The Value of N1 is inside main function Before passing : 45
         The Value inside display function (n1 x n1) :2025
         The Value of N1 is inside main function After passing (n1 x n1)  : 2025
```

display() செயற்கூற்றின் தலைப்புப் பகுதியில் உள்ள மாற்றத்தை கவனிக்கவும். & என்ற குறியுடன் x என்ற அளபுருவை அறிவிக்கப்பட்டு, இதை குறிப்பு மாறி என்பதை உணர்த்தும் இதனால் இந்த செயற்கூறை அழைக்கப்படும்போது குறிப்பு மூலம் அழைக்கப்பட வேண்டும். இதனால் **display()** செயற்கூறை அழைக்கும் போது num1 என்ற மாறியின் முகவரியை x மாறி பகிர்ந்து கொள்ளும். **main()** செயற்கூறில் num1 என்ற மாறியின் பெயரும் **display()** செயற்கூறில் x என்ற மாறியின் பெயரும் ஒரே சேமிப்பு பகுதியைக் குறிக்கிறது. இதனால் x மாறியின் மதிப்பை மாற்றியமைத்தால் அந்த மாற்றம் num1 மாறியின் மதிப்பை உண்மையில் மாற்றும்.

11.6.3 Inline செயற்கூறு

ஒரு செயற்கூறை அழைக்கும் கூற்று நிரல்பெயர்ப்பியை செயற்கூறுக்கு (வரையறுக்கப்பட்ட செயற்கூறுகள்

அடுக்கங்களில் (STACKS) சேமிக்கப்படும்) தாவச் செய்து மற்றும் செயற்கூறு செயல்பாட்டின் பின் அழைப்புக்கூறுக்கு அடுத்திருக்கும் கட்டளைக்குத் தாவச்செய்கிறது. இதனால் நிரலின் இயக்க நேரத்தின் வேகம் குறையும். சிறிய செயற்கூறின் வரையறுப்புக்கு அடுக்கங்களைப் பயன்படுத்துவதைக் குறைக்க Inline செயற்கூறுகள் பயன்படுத்தலாம்.

inline செயற்கூறு மூல நிரலில் சாதாரணச் செயற்கூறு போன்றே தோற்றமளிக்கும். ஆனால் செயற்கூறின் கட்டளைகள் முழுமையும் அழைப்புக் கூற்றுக்குப் பதிலாக அப்படியே நிரலில் செருகப்பட்டுவிடும். **Inline** சிறப்புச் சொல்லை செயற்கூறின் தலைப்பில் இணைத்து அந்த செயற்கூறை **inline** செயற்கூற்றாக மாற்ற முடியும்.

தொடரியல் :

inline returntype functionname (datatype parametername1,... datatype parameternameN)

inline செயற்கூற்றுகளின் நன்மைகள்

- **inline** செயற்கூறுகள் வேகமாக செயல்படும். ஆனால் அதிக நினைவக இடத்தை எடுத்துக் கொள்ளும்.
- அடுக்கங்களை பயன்படுத்தும் போது உள்ள சிக்கல்பாட்டினைக் குறைக்கிறது.

நிரல் 11.19

```
#include <iostream>
using namespace std;
inline float simpleinterest(float p1,float n1, float
r1)
{
    float si1=(p1*n1*r1)/100;
    return(si1);
}
int main ()
{
    float  si,p,n,r;
    cout<<"\nEnter the Principle Amount Rs. :";
    cin>>p;
    cout<<"\nEnter the Number of Years      :";
    cin>>n;
```

நிரல் 11.19

```
    cout<<"\nEnter the Rate of Interest     :";
    cin>>r;
    si=simpleinterest(p,n,r);
    cout << "\nThe Simple Interest = Rs."<<si;
    return 0;
}
```

வெளியீடு:

```
    Enter the Principle Amount Rs. :60000
    Enter the Number of Years      :10
    Enter the Rate of Interest     :5
    The Simple Interest = Rs.30000
```

மேலே கொடுக்கப்பட்டுள்ள நிரலில் உள்ள செயற்கூறின் வரையறுப்பு பொதுவான வடிவில் இருந்தாலும் நிரல்பெயர்ப்பிக்கு பின் செயற்கூறை இயக்கும் போது (p1*n1*r1)/100 என்ற செயற்கூறின் கூற்று si=simpleinterest(p,n,r); என்று அழைக்கும் கூற்றை si= (p1*n1*r1)/100; என்று மாற்றியமைக்கும்.

11.7 பயனர் வரையறுத்த செயற்கூற்றுகளை அறிவிக்கும் போது பயன்படுத்தப்படும் பல்வேறு வடிவங்கள்

11.7.1 மதிப்பை திருப்பி அனுப்பாத மற்றும் அளபுருக்களை ஏற்காத செயற்கூறு

கீழே கொடுக்கப்பட்டுள்ள நிரல் மதிப்பைத் திருப்பி அனுப்பாத மற்றும் அளபுருக்களை ஏற்காத செயற்கூறிற்கான ஒரு எடுத்துக்காட்டு.

display() என்பது செயற்கூறின் பெயர், இதன் திருப்பி அனுப்பும் தரவினம் void மற்றும் இந்த செயற்கூறு எந்த அளபுருவையும் ஏற்காது.

நிரல் 11.20

```
#include<iostream>
using namespace std;
void display()
{
    cout<<"First C++ Program with Function";
}
int main()
{
    display(); // Function calling statement//
    return(0);
}
```

வெளியீடு :

```
    First C++ Program with Function
```

11.7.2 திருப்பி அனுப்பும் மதிப்பு மற்றும் அளபுருக்களை ஏற்காத செயற்கூறு

display() என்ற செயற்கூறின் திருப்பி அனுப்பும் தரவினம் int மற்றும் இந்த செயற்கூறு அளபுருவையும் ஏற்காது. return செயற்கூறு அழைப்பு செயற்கூறுக்கு மதிப்பைத் திருப்பி அனுப்பும் மற்றும் நிரலின் கட்டுப்பாட்டை மீண்டும் அழைப்புக் கூற்றுக்கு திருப்பி அனுப்பும்.

நிரல் 11.21

```
#include<iostream>
using namespace std;
int display()
{
        int a, b, s;
        cout<<"Enter 2 numbers: ";
        cin>>a>>b;
        s=a+b;
        return s;
        }
int main()
{
         int m=display();
        cout<<"\nThe Sum="<<m;
        return(0);
}
```

வெளியீடு :

```
        Enter 2 numbers: 10 30
        The Sum=40
```

11.7.3 மதிப்பை திருப்பி அனுப்பாத மற்றும் அளபுருக்களை ஏற்கும் செயற்கூறு

display() என்ற செயற்கூறின் திருப்பி அனுப்பும் தரவினம் void, மேலும் இது x மற்றும் y என்ற இரண்டு அளபுருக்கள் அல்லது செயலுருபுக்களின் மதிப்புகளை ஏற்கும். **return** கூற்று கட்டுப்பாட்டை அழைப்பு கூற்றுக்குத் திருப்பி அனுப்பும்.

நிரல் 11.22

```
#include<iostream>
using namespace std;
void display(int x, int y)
{
        int s=x+y;
        cout<<"The Sum of Passed Values: "<<s;
}
int main()
{
        int a,b;
        cout<<"\nEnter the First Number    :";
        cin>>a;
        cout<<"\nEnter the Second Number  :";
         cin>>b;
         display(a,b);
        return(0);
```

11.7.4 மதிப்பை திருப்பி அனுப்பும் மற்றும் அளபுருவை ஏற்கும் செயற்கூறு

display(), என்ற செயற்கூறு **int** என்ற மதிப்பைத் திருப்பி அனுப்பும். மேலும் **x** மற்றும் **y** என்ற இரண்டு அளபுருக்கள் அல்லது செயலுருபுக்களில் மதிப்புகளை ஏற்கும். **return** கூற்று கட்டுப்பாட்டை அழைப்பு கூற்றிக்குத் திருப்பி அனுப்பும்.

நிரல் 11.23

```
#include<iostream>
using namespace std;
int display(int x, int y)
{
        int s=x+y;
        return s;
}
int main()
{
         int a,b;
         cout<<"\nEnter the First Number    :";
        cin>>a;
        cout<<"\nEnter the Second Number  :";
        cin>>b;
        int s=display(a,b);
        cout<<”\nExample:Function with Return Value and  with Arguments”;
        cout<<"\nThe Sum of Passed Values: "<<s;
        return(0);
}
```

வெளியீடு :

```
        Enter the First Number    :45
        Enter the Second Number  :67
        Example: Function with Return Value and  with Arguments
        The Sum of Passed Values: 112
```

11.8 கட்டுப்பாட்டை செயற்கூறிலிருந்து திருப்பி அனுப்புதல்

return கூற்றை பயன்படுத்தி கட்டுப்பாட்டை செயற்கூறிலிருந்து திரும்பப் பெறலாம்.

return கூற்று செயற்கூறின் இயக்கத்தை நிறுத்தி கட்டுப்பாட்டை அழைத்த செயற்கூறுக்கு திருப்பி அனுப்பும். செயற்கூற்றின் பகுதியில் எங்கு return கூற்று இயக்கப்படுகிறதோ, உடனே அந்த இடத்தில் செயற்கூறின் செயல்பாடு நிறுத்தப்பட்டு, கட்டுப்பாடு அழைப்பு கூற்றுக்குத் திரும்பும்.

11.8.1 return கூற்று

செயற்கூறிலிருந்து கட்டுப்பாட்டைத் திரும்பப் பெற **return** கூற்று பயன்படுகிறது. இதை தாவும் கூற்றாக வகைப்படுத்தலாம். ஏனென்றால் இந்த கூற்றை இயக்கியவுடன் செயற்கூறின் இயக்கம் நிறுத்தப்பட்டு, கட்டுப்பாடு அழைக்கும் கூற்றுக்கு திரும்ப அனுப்பலாம். **return** கூற்று அதனோடு தொடர்புடைய மதிப்பு உடன் அல்லது மதிப்பு இல்லாமலோ இருக்கலாம். மதிப்பு உடன் உள்ள **return** கூற்று அழைப்பு கூற்றுக்கு அந்த மதிப்பைத் திருப்பி அனுப்பும். **void** செயற்கூறிற்கும் அளபுரு ஏற்காத **return** கூற்றைப் பயன்படுத்தி அந்த செயற்கூற்றின் இயக்கத்தை நிறுத்தலாம்.

தொடரியல்:

```
return expression/variable;
```

எடுத்துக்காட்டு :

```
return(a+b);   return(a);

return; // to terminate the function
```

11.8.2 திருப்பி அனுப்பும் மதிப்புகள்:

மதிப்பினைத் திருப்பி அனுப்பாத செயற்கூற்றினை **void** என்று அறிவிக்கலாம். எந்த தரவினத்தையும் வெளிப்படையாக அறிவிக்கவில்லையெனில் செயற்கூறின் திருப்பி அனுப்பும் தரவினம் **int** என்று கருதும்.

எடுத்துக்காட்டு :

```
int add (int, int);

add (int, int);
```

இவ்விரு முன்வடிவங்களும், int என்ற மதிப்பை திருப்பி அனுப்பும், ஏனெனில் C++ மொழியில் செயற்கூறு திருப்பி அனுப்பும் மதிப்பு தானமைவாகவே int ஆக இருக்கும். கீழ்காணும் எடுத்துக்காட்டுகளை காண்க.

வரிசை எண்	செயற்கூறு முன்வடிவு	திருப்பியனுப்பும் தரவினம்
1	int sum (int, float)	int
2	float area(float, float)	float
3	char result()	char
4	double fact(int n)	double

முழு எண் அல்லாத மதிப்புகளை திருப்பி அனுப்புதல்

அழைப்பு கூற்றிற்கு சரத்தை கூட திருப்பி அனுப்ப முடியும்.

நிரல் 11.24

```
#include<iostream>
#include<string.h>
using namespace std;
char *display()
{
    return ("chennai");
}
int main()
{
    char s[50];
    strcpy(s,display());
    cout<<"\nExample:Function with Non
    Integer Return"<<s;
    return(0);}
```

வெளியீடு :

```
Example: Function with Non Integer Return
Chennai
```

11.9 தற்சுழற்சி செயற்கூறு (Recursive Function)

ஒரு செயற்கூறு தன்னைத் தானே அழைத்துக் கொண்டால் அதை தற்சுழற்சி செயற்கூறு என்று அறியப்படும். இந்த நுட்பம் தற்சுழற்சி முறை என்றழைக்கப்படும்.

எடுத்துக்காட்டு 1: **தற்சுழற்சி முறையில் ஒரு எண்ணின் காரணியை கணக்கிடுக.**

நிரல் 11.25

```
#include <iostream>
using namespace std;
int factorial(int); // Function prototype //
int main()
{
        int no;
        cout<<"\nEnter a number to find its factorial: ";
        cin >> no;
        cout << "\nFactorial of Number " << no <<" = " << factorial(no);
        return 0;
}
int factorial(int m)
{
```

நிரல் 11.25

```
          if (m > 1)
          {
          return m*factorial(m-1);
          }
          else
          {
          return 1;
}
}
```

வெளியீடு :

```
          Enter a number to find its factorial: 5
          Factorial of Number 5 = 120
```

குறிப்பு: main() செயற்கூறின் பின்பு factorial() செயற்கூறை எழுதியிருப்பதால் இந்த செயற்கூறின் முன் வடிவம் கொடுப்பது அவசியமானது.

11.10 மாறிகளின் வரையெல்லை விதிமுறைகள் (Scope rules of Variables)

வரையெல்லை என்பது ஒரு மாறியின் அணுகியல்பைக் குறிக்கிறது. C++ மொழியில் நான்கு வகையான வரையெல்லைகள் உள்ளன. அவை உள்ளமை வரையெல்லை (Local scope), செயற்கூறு வரையெல்லை (Function scope), கோப்பு வரையெல்லை (File scope) மற்றும் இனக்குழு வரையெல்லை (Class scope).

11.10.1 முன்னுரை

வரையெல்லை என்பது ஒரு மாறி செயல்படும் வரம்பெல்லை அல்லது அதன் வாழ்நாள் வரையாகும். மேலும் இதை விவரிக்கும் போது மாறிகளை நான்கு இடங்களில் அறிவிக்கலாம்.

- ஒரு தொகுதிக்குள் அறிவிக்கும்போது அவற்றை உள்ளமை மாறிகள் என்றழைக்கப்படும்.
- செயல்கூறின் உள்ளே அறிவித்தால் அவற்றை செயல்கூறு மாறிகள் என்றழைக்கப்படும்.
- எல்லா செயற்கூறுக்கும் வெளியே அறிவித்தால், அவற்றை பொதுமையான / முழுதாளவிய (Global) மாறிகள் என்றழைக்கப்படும்.
- இனக்குழுவில் உள்ளே அறிவித்தால் அவற்றை இனக்குழு மாறிகள் அல்லது தரவு உறுப்புகள் (data members) என்று அழைக்கப்படும்.

11.10.2 உள்ளமை வரையெல்லை:

- உள்ளமை மாறி, ஒரு தொகுதிக்குள் **(Block)** வரையறுக்கப்படுகிறது. ஒரு தொகுதியில் உள்ள நிரல் { } என்ற அடைப்புக்குறிக்குள் இருக்கும்.
- ஒரு உள்ளமை மாறியின் வரையெல்லை அது வரையறுக்கப்பட்டுள்ள தொகுதிக்குள்

மட்டுமே இருக்கும்.

- ஓர் உள்ளமை மாறியை அது அறிவிக்கப்பட்டுள்ள தொகுதிக்கு வெளியிலிருந்து அணுக முடியாது.
- நிரலின் கட்டுப்பாடு ஒரு கட்டளைத் தொகுதிக்குள் நுழையும் போது, அதன் உள்ளமை மாறிகள் உருவாக்கப்படுகின்றன. வெளியேறும் போது அவை அழிக்கப்படுகின்றன.

11.10.3 கோப்பு வரையெல்லை:

- செயற்கூறினுள் அறிவிக்கப்பட்ட மாறியின் வரையெல்லை அந்த செயற்கூறின் தொகுதி மற்றும் துணை தொகுதி வரை உள்ளது.
- மாறியின் வாழ்நாள் செயற்கூறு தொகுதியின் வாழ்நாள்வரைக்கும் இருக்கும் முறையான அளபுருக்களின் வரையெல்லை செயற்கூறின் வரையெல்லை ஆகும்.
- கோப்பு வரையெல்லை மாறியை பொதுமை மாறிகள் என்றழைக்கப்படும்.

11.10.4 கோப்பு வரையெல்லை:

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

நிரல் 11.26

```
//Demo to test all Scopes//
#include<iostream>
using namespace std;
int file_var=20;            //Declared within File - file scope variable
void add(int x)
{
        int m;              //Declaration of  variable m in add () - Function scope variable
        m=x+30+file_var;
        cout<<"\n The Sum = "<<m;
}
int main ( )
{
int a ;
a = 10;
if(a>b)
{
        int t;              // local to this if block - Local variable
        t=a+20;
 }
```

நிரல் 11.26

```
cout<<t;

add(a);

cout<<m;

cout<<”\nThe File Variable = “<<file_var;

return(0);

}
```

Error

குறிப்பு In function 'int main()':

[Error] 't' was not declared in this scope

மேலே கொடுக்கப்பட்டுள்ள நிரலைச் செயல்படுத்தும் பொழுது, நிரல் பெயர்ப்பி ஒரு பிழை செய்தியைக் கொடுக்கிறது. என்னும் மாறியை அணுகமுடியாது. ஏனெனில் அதன் வாழ்நாள் இயங்கு நிலையில் அதன் தொகுதியில் வாழ்நாள் வரைக்கும் இருக்கும் தொகுதி நிறைவேற்றப்பட்டபின் உள்ள t- ன் வாழ்நாள் முடிந்து விடும்.

[Error] 'm' was not declared in this scope

மாறியின் வாழ்நாள் செயற்கூறு தொகுதியின் வாழ்நாள்வரைக்கும் இருக்கும்.

11.10.5 இனக்குழு வரையெல்லை:

- பயனர்கள் புதிய தரவினங்களை உருவாக்கவும், நடைமுறைப் படுத்தவும் ஒரு புதிய வழியை இனக்குழு திறக்கிறது. வேறுபட்ட இனத்தரவுகளை ஒன்றாகச் சேர்த்து வைக்க இனக்குழுக்கள் ஒரு புதிய வழிமுறையை வழங்குகின்றன.
- தரவு உறுப்புகள் தரவு மாறிகள் என்று அழைக்கப்படும், இவை இனக்குழுவின் பண்புக்கூறுகளை உணர்த்தும்.

```
class  student
{
   private :
int mark1, mark2,
total;
};
```

student என்ற இனக்குழுவில் mark1, mark2 மற்றும் total என்பவை தரவு மாறிகள். இந்த தரவு மாறிகளின் வரையெல்லை student இனக்குழுவின் வரையெல்லையாகும்.

குறிப்பு

“Classes and Object” என்ற அதிகாரத்தில் இனக்குழு வரையெல்லை பற்றி விரிவாக தெரிந்து கொள்ளலாம்.

11.10.6 வரையெல்லை தெளிவுபடுத்தும் செயற்குறி (Scope resolution operator)

வரையெல்லை தெளிவுபடுத்தும் செயற்குறி ஒரு மாறியின் மறைந்து கிடக்கும் வரையெல்லையை வெளிக்கொணரும். வரையெல்லை தெளிவுபடுத்தும் செயற்குறி :: இதற்கு பயன்படுகிறது.

- பொதுமையான மாறியும் உள்ளமை மாறியும் ஒரே பெயரை கொண்டிருப்பதால், பொதுமையான மாறியை இயக்க வரையெல்லை தெளிவுபடுத்தும் செயற்குறி பயன்படுத்தும் ஒரு எடுத்துக்காட்டு.

நிரல் 12.27

```
// Program to show that we can access a global variable
// using scope resolution operator :: when there is a local
// variable with same name //
#include<iostream>
using namespace std;
int x=45;  // Global Variable x
int main()
{
 int x = 10; // Local Variable x
cout << "\nValue of global x is " << ::x;
 cout << "\n\nValue of local x is " << x;
 return 0;
}
```

வெளியீடு:

```
      Value of global x is 45
      Value of local x is 10
```

ஆய்வு அறிக்கை :

1. வர்க்க மூலம் (square root) , அடுக்கின் மதிப்பு (power values), tan, கன மூலம் (cube root) போன்றவற்றைக் கண்டறிய செயற்கூறுகளைப் பயன்படுத்தி நிரலை எழுதுக.

2. ஐந்து மாணவர்களின் பெயர்களை அவர்களின் தலைப்பு எழுத்தை இறுதியில் அமையுமாறு உள்ளீடாக செய்க, பெயரை ஆங்கில சிறிய மற்றும் பெரிய எழுத்துக்களாக வெளியீடாக செய்யவும். மேலும் ஒவ்வொரு பெயருக்கும் உள்ள எழுத்துக்களின் எண்ணிக்கை வெளியீடாக பெற உரிய நிரலை எழுதுக.

3. காரணிப்படுத்துதல் (factorial), பகா எண்(prime number), ஆம்ஸ்டார்ங் எண்கள் (Armstrong numbers) போன்றவை கண்டறிய செயற்கூறுகளை பயன்படுத்தி நிரலை எழுதுக.

4. ஒருவரின் பெயர் மற்றும் பாலினம் உள்ளீடாக பெற்று திரு/ திருமதி என்ற சொல்லை பெயருடன் இணைத்து வெளியிடுவதற்கு உரிய நிரலை எழுதுக.

நினைவில் கொள்க

- ஒரு பெரிய நிரலை சிறு சிறு பகுதிகளாக பிரிப்பதையே செயற்கூறுகளாகும்.
- செயற்கூறுகள் முன் வரையறுக்கப்பட்ட அல்லது உள்ளிணைந்த அல்லது நூலக செயற்கூறுகள் மற்றும் பயனர் வரையறுத்த செயற்கூறுகள் என வகைப்படுத்தலாம்.
- பயனர் வரையறுத்த செயற்கூறுகள் பயனரால் உருவாக்கப்படும்.
- void செயற்கூறு எந்த மதிப்பையும் செயற்கூறு திருப்பி அனுப்பாது என்று நிரல்பெயர்ப்பிக்கு தெரிவிக்கும்.
- return கூற்று அழைப்பு செயற்கூற்றுக்கு ஒரு மதிப்பை திருப்பி அனுப்பியும், மேலும் அழைக்கும் செயற்கூற்றிற்கு கட்டுப்பாட்டை திருப்பி அனுப்பும்.
- C++ மொழியில் தானமைவாக திருப்பி அனுப்பும் தரவு வகை int தரவினமாகும்.
- தன்னைத் தானே அழைத்துக் கொள்ளும் செயற்கூறுகளை தற்சுழற்சி செயற்கூறு என்றழைக்கப்படும்.
- வரையெல்லை என்பது ஒரு மாறியின் அணுகியல்பைக் குறிக்கும்.
- உள்ளமை வரையெல்லை, செயற்கூறு வரையெல்லை, கோப்பு வரையெல்லை மற்றும் இனக்குழு வரையெல்லை என நான்கு வகையான வரையெல்லைகள் உள்ளன.
- வரையெல்லை செயற்குறி (::) ஒரு மாறியின் மறைக்கப்பட்ட வரையெல்லையை தெரிவிக்கும்.

மதிப்பீடு

N1H9CG

பகுதி அ

I சரியான விடை யை தேர்வு செய்யவும் :

1. இவற்றுள் எந்த தலைப்பு கோப்பு நிலையான I/O விற்கான முன்வரையறுக்கப்பட்ட செயற்கூறுகளை வரையறுக்கும் ?

அ) stdio.h ஆ) math.h

இ) string.h ஈ) ctype.h

2. ஒரு குறியுறுவை எழுத்து மற்றும் எண் வகையா அல்லது இல்லையா என்பதை சரிபார்க்க உதவும் செயற்கூறு எது?

அ) isalpha() ஆ) isdigit()

இ) isalnum() ஈ) islower()

3. நிரலின் செயலாக்கம் எந்த செயற்கூறிலிருந்து தொடங்கும் ?

அ) isalpha() ஆ) isdigit()

இ) main() ஈ) islower()

4. இவற்றுள் எந்த செயற்கூறு ஒரு மதிப்பைத் திருப்பி அனுப்பி மற்றும் செயலுருபுகளை ஏற்காத செயற்கூறு ஆகும்?

அ) x=display(int, int) ஆ) x=display()

இ) y=display(float) ஈ) display(int)

5. add(int, int); என்ற செயற்கூற்றின் முன்வடிவின் திருப்பி அனுப்பும் தரவினத்தின் வகை யாது?

அ) int ஆ) float

இ) char ஈ) double

6. இவற்றுள் எது வரையெல்லை செயற்குறியாகும் ?

அ) > ஆ) &

இ) % ஈ) ::

பகுதி - ஆ

குறு வினாக்கள் (2 மதிப்பெண்கள்) :

1. செயற்கூறுகள் - வரையறை
2. strlen() செயற்கூறை பற்றி எழுதுக
3. void தரவு வகையின் முக்கியத்துவங்கள் என்ன?
4. அளபுரு என்றால் என்ன ? அதன் வகைகளை பட்டியலிடுக
5. உள்ளமை வரையெல்லை பற்றி சிறுகுறிப்பு வரைக.

பகுதி - இ

சிறு வினாக்கள் (3 மதிப்பெண்கள்) :

1. உள்ளிணைந்த செயற்கூறுகள் என்றால் என்ன ?
2. isuppr() மற்றும் toupper() செயற்கூறுகளின் வேறுபாடுகள் யாவை?
3. strcmp() செயற்கூறு பற்றி குறிப்பு வரைக.
4. C++ மொழியில் உள்ள pow() செயற்கூறு பற்றி சிறுகுறிப்பு வரைக.
5. செயற்கூறு முன்வடிவம் நிரல்பெயர்ப்பிக்கு எந்த தகவலை வழங்கும் ?
6. முன்னிலைப்பு செயலுருபுக்கள் என்றால் என்ன ? எடுத்துக்காட்டு தருக.

பகுதி - ஈ

பெரு வினாக்கள் (5 மதிப்பெண்கள்) :

1. மதிப்பு மூலம் அழைத்தல் முறையை தகுந்த எடுத்துக்காட்டுடன் விளக்குக.
2. தற்சுழற்சி என்றால் என்ன ? தற்சுழற்சி முறையில் ஒரு எண்ணிற்கான மிகப்பெரிய பொதுவான காரணியை கணக்கிட ஒரு நிரலை எழுதுக.
3. செயற்கூறு மதிப்பை திருப்பி அனுப்பும் பல்வேறு வடிவங்களை எடுத்துக்காட்டுடன் விளக்குக.
4. மாறியின் வரையெல்லை விதிமுறைகளை எடுத்துக்காட்டுடன் விளக்குக.
5. ஒரு முழு எண்ணை உள்ளீட்டு அதை தலைகீழாக மாற்றம் செய்யும் நிரலை எழுதுக.