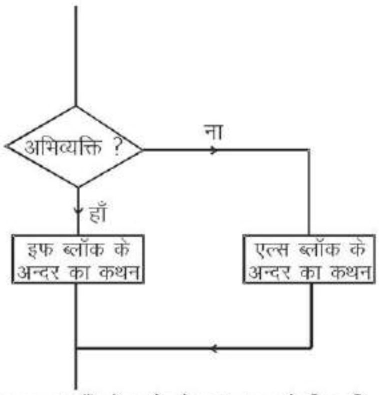
1

नियत्रंण का प्रवाह
(Flow of Control)

1.1 कण्डीशनल अभिव्यक्ति (Conditional expression)

कण्डीशनल अभिव्यक्ति वह होती है जिसका उत्तर हाँ या ना में आता है।

```
if (expression)
        कथन 1;
else
        कथन 2;
```



यदि अभिव्यक्ति का उत्तर हाँ होता है तो कथन 1 को निष्पादित (Execute) करेगा अन्यथा कथन 2 को निष्पादित करेगा।

उदाहरण 1

C++ में एक प्रोग्राम लिखो जो कि सेल्सियस तापमान को फॉरनेहाइट में और फॉरनेहाइट तापमान को सेल्सियस तापमान में परिवर्तित कर दें। इसके लिए प्रोग्राम का चयन वह यूज़र (User) से पूछेगा।

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

```
# include <conio.h>
void main()
{
int choice;
float f,c;
cout<<"\n enter 1 for convert Celsius to fahrenheit and 2 for Fahrenheit to
Celsius";
cin>>choice;
if(choice==1)
      {
       cout<<"\n enter temperature in Celsius";
       cin>>c;
      f= (c-32)/1.8;
      cout<<"\n Fahrenheit="<<f;
      }
else
      {
      cout<<"\n enter temperature in Fahrenheit";
      cin>>f;
      c= (1.8*f)+32;
      cout<<"\n Celsius="<<c;
      }
      getch ( );
}
```

1.2 नेस्टेड इफ (Nested if)

नेस्टेड इफ कथन वह होता है जिसमें एक इफ (if) कथन के अंदर दूसरा इफ कथन होता है। इसको हम निम्न प्रकार परिभाषित करेंगे।

```
if (expression 1)
{
        if (expression 2)
                कथन–1;
        else
                कथन–2;
}
else
{
        if (expression 3)
                कथन–3;
        else
                कथन–4;
}
```

उदाहरण एक प्रोग्राम लिखो जो कि तीन संख्या को यूजर से पूछे और सबसे बडी संख्या को प्रिंट करें।

```
#include<iostream.h>
# include <conio.h>
void main()
{
    int a,b,c;
    cout<<"\n enter the value of a";
    cin>>a;
    cout<<"\n enter the value of b";
    cin>>b;
    cout<<"\n enter the value of c";
    cin>>c;
    if(a>b)
        {
        if(a>c)
        cout<<"\n greatest number is"<<a;
        else
        cout<<"\n greatest number is"<<c;
        }
    else
        {
        if(b>c)
        cout<<"\n greatest number is"<<b;
        else
        cout<<"\n greatest number is"<<c;
        }
    getch ( );
}
```

1.3 स्विच – केस – डिफाल्ट (switch... case... default)

C++ में एक से ज्यादा कथनों में से, एक का चयन करने के लिए स्विच कथन (switch) का प्रयोग करते है।

```
switch (expression)
    {
    case नियतांक 1 : कथन 1;
                    break;
    case नियतांक 2 : कथन 2;
                    break;
    case नियतांक 3 : कथन 3;
                    break;
        default : कथन–4;
    }
```

उदाहरण

C++ भाषा में एक प्रोग्राम लिखो जो कि सप्ताह के दिन का नंबर यूजर से पूछे और फिर सप्ताह का नाम प्रिंट करे।

```
#include<iostream.h>
# include <conio.h>
void main()
{
 int choice;
cout<<"\n enter the number of the weekday";
cin>>choice;
switch (choice)
        {
         case 1: cout<<"\n Sunday";
                 break;
        case 2: cout<<"\n Monday";
                 break;
        case 3: cout<<"\n Tuesday";
                 break;
        case 4: cout<<"\n Wednesday ";
                 break;
        case 5: cout<<"\n Thursday";
                 break;
        case 6: cout<<"\n Friday";
                 break;
        case 7: cout<<"\n Saturday";
                 break;
        default:cout<<"\n you entered wrong choice";
        }
getch ( );
}
```

1.4 नेस्टेड स्विच केस (Nested switch case)

एक स्विच कथन दूसरे स्विच कथन के अंदर हो तो, इस प्रकार के कथन को नेस्टेड स्विच केस कहते हैं।

1.5 ब्रेक कथन (Break Statement)

C++ भाषा में ब्रेक कथन (Break Statement) किसी भी लूप के बीच में से बाहर निकालने का कार्य करता है। जैसे ही ब्रेक (Break Statement) कथन आता है। प्रोग्राम के कन्ट्रोल को लूप से बाहर निकाल देता है।

उदाहरण :

```
#include<iostream.h>
# include <conio.h>
void main()
{
int a=12,i=1;
while (0)
{
if (i==3)
{
break;
}
else
{
        p=a*i;
        i=i+1;
        cout<<"\n"<<p;
        }
}
getch ( );
}
output :12
24
```

उपरोक्त प्रोग्राम में जैसे ही i का मान 3 होगा Break कथन के कारण यह आगे निष्पादित नहीं करेगा।

1.6 लूप (LOOP)

लूप साधारणतया जब प्रयोग करते है, जब किसी कथन को एक से ज्यादा बार दोहराया जाना होता है।

यहां पर तीन प्रकार की लूप (Loops) होती है।

(i) व्हाइल (while)

(ii) डू–व्हाइल (do-while)

(iii) फॉर लूप (for loop)

1.7 व्हाइल (while) लूप

व्हाइल लूप का सिन्टेक्स निम्न प्रकार है।

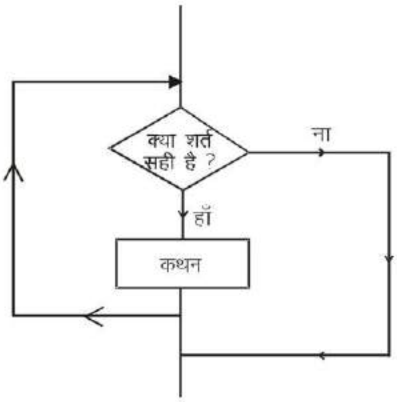
```
while (Condition)
{
        कथन–1;
}
```

यदि कन्डीशन (Condition) सत्य है तो यह व्हाइल लूप के अंदर जो कथन–1 है। उसको निष्पादित (Execute) करेगा। और यदि कन्डीशन सत्य नहीं है तो यह लूप के अंदर लिखे कथन को निष्पादित (Execute) नहीं करेगा। व्हाइल लूप को साधारणतया एन्ट्री कन्ट्रोल लूप (Entry Control loop) कहते है क्योंकि इसके अंदर प्रवेश करने के पहले यह शर्त (Condition) की जांच करता है।

उदाहरण :– निम्नलिखित प्रोग्राम का आउटपुट लिखो।

```
#include<iostream.h>
# include <conio.h>
void main()
{
        int a=7;
        while(a<10)
        {
        cout<<"\n"<<a;
        a=a+1;
        }
getch ( );
}
```

उत्तर : 7
8
9

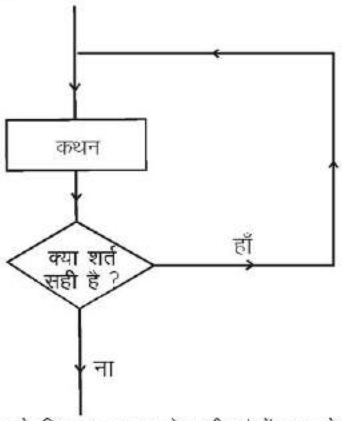


1.8 डू–व्हाइल लूप (do-while loop)

डू व्हाइल लूप को एक्जिक्ट कन्ट्रोल लूप (Exit Control loop) कहते है, क्योंकि यह शर्त की जांच तब करती है, जब वह लूप से बाहर निकलती है। यदि कोई शर्त पहले से ही ना (False) है तो भी यह लूप कम से कम एक बार तो निष्पादित होगी ही जबकि फॉर लूप (for loop) और व्हाइल लूप में यदि शर्त पहले से ही ना (False) है तो यह दोनों लूप एक भी बार निष्पादित नहीं होगी। साधारणतया डू–व्हाइल लूप का प्रयोग तब करते है जब हमें यह नहीं पता है कि यह लूप कितनी बार निष्पादित होगी। डू–व्हाइल लूप का सिन्टेक्स

```
do
{
कथन–1;
```

```
} while (Condition);
```



उदाहरण एक प्रोग्राम लिखो जो कि एक नम्बर के सभी अंकों का जोड कर दे। (यदि यूजर 3427 नंबर देता है तो उसका उत्तर 3+4+2+7=16 आये)

```
#include<iostream.h>
# include <conio.h>
void main()
{
        int n,s=0,d;
        cout<<"\n enter the number";
        cin>>n;
                do
                {
                d=n%10;
                s=s+d;
                n= n/10;
                }while(n!=0);
        cout<<"\n sum="<<s;
getch ( );
}
```

1.9 फॉर लूप (for loop) फॉर लूप का प्रयोग तब किया जाता है जब यूजर को पहले से ही पता हो कि यह लूप एक निश्चित संख्या तक निष्पादित होगी।

फॉर लूप का सिन्टेक्स निम्न है:–

```
for (initialization; condition; increment)
{
        कथन–1;
}
```

उदाहरण :– एक प्रोग्राम लिखो जो कि यूजर द्वारा दिये गये नम्बर तक प्राकृत संख्याओं को प्रिंट करे।

```
#include<iostream.h>
# include <conio.h>
void main()
{
    int n,i;
    cout<<"\n enter the number";
    cin>>n;
    for(i=1;i<=n;i++)
        {
        cout<<i;
        }
getch ( );
}
```

उदाहरण :– एक प्रोग्राम लिखो जो कि किसी भी संख्या का फेक्टोरियल (Factorial) निकाले।

फेक्टोरियल n = n(n-1) 3.2.1

```
#include<iostream.h>
# include <conio.h>
void main()
{
    int n,f=1,i;
    cout<<"\n Enter the number";
    cin>>n;
        for(i=1;i<=n;i++)
        {
        f=f*i;
        }
    cout<<"\n Factorial="<<f;
getch ( );
}
```

1.10 नेस्टेड लूप (Nested loop)

जब एक लूप के अंदर, दूसरी लूप हो तो उसे नेस्टेड लूप कहते है। किन्तु एक नेस्टेड लूप में, अंदर वाली लूप, बाहर वाली लूप से पहले समाप्त (Terminate) होगी।

उदाहरण :–

```
for(int i=1;i<5;i++)
{
for(int j=1;j<=i;j++)
    {
    cout<<j;
    }
cout<<"\n";
}
```

```
1
12
123
1234
```

अभ्यास के लिए प्रश्न

वस्तुनिष्ठ प्रश्न (Obective Type Questions)

निम्नलिखित प्रश्नो में सही विकल्प का चुनाव कीजिए

1. ब्रेक (break) कथन बाहर निकालता है।
   (अ) केवल अंदर वाली लूप से
   (ब) केवल अंदर वाले स्विच(Swith) कथन से
   (स) सभी लूप और स्विच से
   (द) अंदर की लूप या स्विच से
2. जब भी हमें यह पता होता है कि यह लूप एक निश्चित संख्या तक निष्पादित होगी तो हम काम में लेंगे।
   (अ) फॉर लूप (for loop) (ब) डू व्हाइल लूप (Do-while)
   (स) व्हाइल लूप (while loop) (द) उपरोक्त में से कोई नहीं
3. नेस्टेड लूप में पहले लूप निष्पादित होती है।
   (अ) बाहर वाली (ब) अंदर वाली
   (स) दोनों एक साथ (द) कोई भी नहीं
4. कई सारी भिन्न–भिन्न शर्तों में से किसी एक का चयन करने के लिए जो कथन काम में लेंगे।
   (अ) इफ कथन (ब) इफ–इेल्स कथन
   (स) स्विच कथन (द) कोई भी नहीं

लघुत्तरात्मक प्रश्न (Short Answer Type Questions)

1. लूप का प्रयोग कब करते हैं ?
2. लूप कितने प्रकार की होती है ?
3. नेस्टेड लूप क्या होती है ?
4. वह कौनसी लूप है जिसका प्रयोग किसी कथन को निशचित बार तक निष्पादित करने के लिए किया जाता है ?

निबन्धात्मक प्रश्न (Essay Type Questions)

1. किसी संख्या का पहाड़ा (Table) प्रिन्ट करने के लिए C++ भाषा में एक प्रोग्राम लिखिये।
2. दो वेरियेबलों के मान को बिना तीसरे वेरियेबल के उपयोग के आपस में बदलने के लिए प्रोग्राम लिखिए।
3. C++ भाषा में एक प्रोग्राम लिखे जो किसी नम्बर के अंकों का जोडकर प्रिन्ट करें। माना नम्बर 5246 है तो आउटपुट 17 होना चाहिये।
4. डू–व्हाइल लूप और फॉर लूप में अंतर लिखिये। बताइये कि दोनो में से कौनसी लूप एग्जिट कन्ट्रोल लूप कहलाती है?

5. निम्न प्रोग्राम का आउटपुट ज्ञात करो–

```
# include <iostream.h>
# include <conio.h>
void main ()
{
int i, s=0;
for(i=0;i<=3; i++)
{
s=s+i;
}
cout<<"\ns="<<s;
cout<<"\ni="<<i;
getch ( );
}
```

उत्तरमाला

1. (द) 2. (ब), 3. (ब), 4. (स),

2

यूजर डिफाइन्ड फंक्शन
(User defined functions)

2.1 फंक्शन (Function)

किसी भी बड़े प्रोग्राम को हल करते समय, उस प्रोग्राम के बीच–बीच में कई बार समान कोड, एक से ज्यादा बार प्रयोग होते है। इस समान कोड के बार–बार प्रयोग करने के कारण प्रोग्राम भी ज्यादा लंबा हो जाता है। तथा उसको समझने में भी समस्या होती है।

इस समस्या के समाधान के लिए, वह समान कोड जो एक से अधिक बार प्रयोग हो रहा है। उसको अलग से लिख दिया जायेगा तथा उसका एक अलग से नाम दे दिया जायेगा। इस अलग नाम को हम फंक्शन का नाम (Function name) कहते हैं।

उसके बाद मुख्य प्रोग्राम (Main Program) में जहां भी उस समान कोड को लिखने की जरूरत होगी। वहां पर वह समान कोड न लिखकर, उस फंक्शन का नाम लिख देंगे। इस तरह से यह मुख्य प्रोग्राम अपने आप फंक्शन के अंदर लिखे कोड को निष्पादित (Execute) कर देगा। इस प्रकार फंक्शन के प्रयोग से समय, जगह (space) की बचत हो जाती है। और प्रोग्राम भी सरल हो जाता है।

फंक्शन को निम्न प्रकार परिभाषित करेंगे–

```
Data Type Function name (Parameter list)
{
body of the function,
return value;
}
```

फंक्शन का नाम यूजर द्वारा दिया जायेगा।

पैरामीटर लिस्ट वह लिस्ट है जो यूजर मुख्य प्रोग्राम से उप प्रोग्राम में भेजने के लिए वेरियबल काम में लेता है। ये सारे वेरियबल अगर एक से ज्यादा है तो इनके बीच कोमा (,) लगायेंगे। रिटर्न (return) कथन परिणाम को या कोई भी मान (value) को उप प्रोग्राम से मुख्य प्रोग्राम में भेजने के लिए काम आता है।

एक फंक्शन बिना पैरामीटर लिस्ट के भी हो सकता है। उस समय दोनों ब्रेकेटों के बीच खाली स्थान छोड़ देगें जो पैरामीटर लिखने के काम आता है

उदाहरण के लिये एक फंक्शन दिया गया है। वह दो नंबर की तुलना करके जो भी नम्बर छोटा होगा उसको फंक्शन के द्वारा (return) रिटर्न करेगा।

```
int min (int a, int b)
```

```
{
if (a<b)
        return a;
else
        return b;
}
```

2.2 फंक्शन प्रोटोटाइप (Function prototype)

एक फंक्शन प्रोटोटाइप फंक्शन की तरह ही होता है जो पहले से ही कम्पाइलर को बता देता है कि हम इस प्रोग्राम में इस फंक्शन का प्रयोग करेंगे। यह फंक्शन इस टाइप का मान return करेगा। तथा इसके पैरामीटर जो मुख्य प्रोग्राम से उप प्रोग्राम में जाने के लिए है वह इतने है। इस प्रकार फंक्शन का प्रयोग करने से पहले ही हम कम्पाइलर को बता देंगे।

साधारणतया इस प्रकार के फंक्शन का डिक्लेयरेशन मुख्य प्रोग्राम से पहले करते हैं। इस प्रकार के फंक्शन को सार्वभौम प्रोटोटाइप फंक्शन (global prototype function) कहते हैं। और यदि फंक्शन को कॉल करने के अंतर्गत डिक्लेयर करते हैं तो उस फंक्शन को स्थानीय प्रोटोटाइप फंक्शन (Local Prototype function) कहते है।

एक फंक्शन जो किसी भी तरह का मान रिटर्न नहीं करता हैं। उसको हम (void) वॉइड टाइप डिक्लेयर करेंगे।

2.3 फंक्शन को बुलाना (Calling a function)

जब किसी फंक्शन का मुख्य प्रोग्राम में प्रयोग करते हैं तो उसको उप प्रोग्राम से, मुख्य प्रोग्राम में बुलाने के लिए जो कोड लिखे जाते हैं। उसे फंक्शन को बुलाना कहते है।

माना एक फंक्शन निम्न प्रकार परिभाषित है।

int max (int x, int y);

अब हम इस फंक्शन को मुख्य प्रोग्राम में बुलाने के लिए निम्न प्रकार से लिखेंगे–

max (x,y);

फंक्शन को बुलाने का नियम वही है जो फंक्शन को डिक्लेयर करने का है। केवल तरीका भिन्न है तो वह डाटा टाइप को नहीं लिखने का है।

```
#include<iostream.h>
# include <conio.h>
float interest(float,float,float);
void main()
{
float p,r,t,s;
cout<<"\n enter the principle";
cin>>p;
cout<<"\n enter the rate";
cin>>r;
cout<<"\n enter the time";
```

```
cin>>t;
s=interest(p,r,t);
cout<<"\n Simple interest="<<s;
getch ( );
}

float interest(float p, float r, float t)
{
float si;
si=(p*r*t)/100;
return(si);
}
```

2.4 डिफाल्ट आरग्यूमेन्ट (Default argument)

C++ भाषा में डिफाल्ट आरग्यूमेन्ट का प्रयोग होता है। यह साधारणतया जब ज्यादा महत्व रखता है। जब प्रयोग करने वाला, किसी पैरामीटर का मान नहीं भेजता और फंक्शन को बुला लेता है। (calling)

उदाहरण:– एक फंक्शन ब्याज का गणना करने के लिए लिखा गया है।

float interest (float p, float r, float t=3);

इस उपरोक्त फंक्शन में समय का डिफाल्ट मान 3 दिया गया है। इसके बाद जब हम फंक्शन को बुलाते (call) है और माना इस प्रकार लिखते है।

Si = interest (2000, .3)

तो यह फंक्शन मूलधन का मान तो 2000 भेज देगा। ब्याज की दर .3 भेज देगा और क्योंकि समय का मान नहीं भेजा है। तो वह अपने आप ही 3 ले लेगा। जो कि डिफाल्ट पैरामीटर में पहले से ही रखी हुई है।

कोई भी आरग्यूमेन्ट तब तक डिफाल्ट वेल्यू नहीं लेगा, जब तक इसका आगे वाला चर मान नहीं लेगा।

यहां पर कुछ सही और गलत उदाहरण डिफाल्ट आरग्यूमेन्ट के बताये गये है–

float interest (float p, float r = .3, float t = 3) सही

float interest (float p = 3000, float r, float t = 3) गलत

float interest (float p = 4000, float r = .3, float t) गलत

float interest (float p, float r, float t = 3) सही

2.5 निश्चित आरग्यूमेन्ट (constant argument)

निश्चित आरग्यूमेन्ट के अंदर आरग्यूमेण्ट का मान निश्चित रहता है, उस मान को हम परिवर्तित नहीं कर सकते।

उदाहरण के लिए हमने एक फंक्शन area को बुलाया (calling) area (10);

इस फंक्शन में आरग्यूमैन्ट का मान 10 है जो कि निश्चित है। हम इस उपरोक्त फंक्शन को इस तरह से लिख देते है।

area (r);

तो यहां पर r एक वेरियबल है। इसकी कोई भी मान हो सकता है। और अगर इस r के मान को भी निश्चित करना है तो हम नया फंक्शन निम्न प्रकार लिखेंगे।

int multiply (const int a, const int b);

इस फंक्शन में const प्रयोग किया गया है। इसका मतलब यह आरग्यूमैन्ट का मान निश्चित है। अब हम इसके मान को परिवर्तित नहीं कर सकते।

एक फंक्शन दो तरीके से बुला (call) सकते है।

(i) काल वाई वेल्यू (Call by value)

(ii) काल वाई रेफरेन्स (Call by reference)

2.6 काल बाई वेल्यू (Call by value)

इसमें वास्तविक (Actual) पैरामीटर को औपचारिक (formal) पैरामीटर में परिवर्तित कर देते है। परिवर्तित करने के बाद यूजर यदि औपचारिक (formal) पैरामीटर में कोई परिवर्तन करता है तो वह वास्तविक (Actual) पैरामीटर में लागू नहीं होता है। इस प्रकार इस तरह की तकनीक में हम मुख्य प्रोग्राम से उप प्रोग्राम में पैरामीटर के द्वारा मान भेज देते है। और उस उप प्रोग्राम के अंतर्गत यदि पैरामीटर में कोई परिवर्तन हो जाता है, या कर दिया जाता है। तो मुख्य प्रोग्राम में जिस पैरामीटर से उसको भेजा था, उसमें परिवर्तन नहीं होगा।

उदाहरण :–

```
#include<iostream.h>
void change(int a, int b);
void main()
{
 int x=2,y=3;
change(x,y);
cout<<"\n x="<<x<<"\n y="<<y;
}

void change (int a, int b)
{
        a=a+5;
        b=b-2;
        cout<<"\n a="<<a<<"\nb="<<b;

}
```

इस प्रोग्राम का उत्तर निम्न आयेगा–

a=7
b=1
x=2
y=3

इस प्रोग्राम में मुख्य प्रोग्राम मे जो पैरामीटर x और y थे उनका मान उप प्रोग्राम के पैरामीटर a और b में परिवर्तित हो गया, जिससे उप प्रोग्राम में a और b का मान 2 और 3 उसके बाद **a** का मान होगा

7 और b का मान होगा 1 फिर यह a और b के मान को प्रिंट कर देगा। उसके बाद कन्ट्रोल मुख्य प्रोग्राम में आयेगा। मुख्य प्रोग्राम में x और y का मान वही रहेगा जो पहले था, यानि 2 और 3.

2.7 काल बाई रेफरेन्स (Call by reference)

जब हम फंक्शन को बुलाते है तो काल बाई रेफरेन्स में वास्तविक (Actual) पैरामीटर को औपचारिक (formal) पैरामीटर में परिवर्तित कर देते है। परिवर्तित करने के बाद यदि उप प्रोग्राम में औपचारिक (formal) पैरामीटर में कोई परिवर्तन करते है तो उसका प्रभाव वास्तविक (Actual) पैरामीटर पर भी पड़ता है। जिससे वास्तविक (Actual) पैरामीटर भी परिवर्तित हो जाते है।

इस तरह की रेफरेन्सिंग में वास्तविक (Actual) पैरामीटर और औपचारिक (formal) पैरामीटर का नाम मेमोरी के अंदर एक ही होता है। जिससे यदि हम वास्तविक (Actual) पैरामीटर को परिवर्तित करेंगे तो वह औपचारिक (formal) पैरामीटर में भी परिवर्तन कर देगा। और यदि हम औपचारिक (formal) पैरामीटर में परिवर्तन करते है तो वह वास्तविक (Actual) पैरामीटर में भी परिवर्तन कर देगा।

काल बाई रेफरेन्स बनाने के लिए हम वेरियबल के पहले & (एम्परसन्ड) का चिन्ह लगा देंगे।

उदाहरण :–

```
void change (int & a, int b)
```

यहां वेरियबल a कॉल बाई रेफरेन्स है, जबकि वेरियबल b कॉल बाय वेल्यू है।

उदाहरण :–

```
#include<iostream.h>
void change(int& a, int& b);
void main()
{
 int x=2,y=3;
change(x,y);
cout<<"\n x="<<x<<"\n y="<<y;
}

void change (int & a, int & b)
{
a=a+5;
b=b-2;
cout<<"\n a="<<a<<"\nb="<<b;
}
```

इस प्रोग्राम का परिणाम निम्न आयेगा–

```
a=7
b=1
x=7
y=1
```

इस प्रोग्राम में (Actual) पैरामीटर का मान मुख्य प्रोग्राम में 2 और 3 है जो कि परिवर्तित होकर उप प्रोग्राम के पैरामीटर (formal) में परिवर्तित हो गया। अब उप प्रोग्राम के अंतर्गत पैरामीटर a और b

का मान 7 और 1 हो गया। इसके प्रभाव से मुख्य प्रोग्राम में भी वेरियबल x और y का मान 7 और 1 हो गया।

उदाहरण :— C++ भाषा में एक प्रोग्राम लिखो जो समीकरण $ax^2+bx+c=0$ के मूल निकाले।

```
#include<iostream.h>
#include<math.h>
# include <conio.h>
void roots(int,int,int,float& x3,float& x4);
void main()
{
int a,b,c;
float root1,root2;
cout<<"\n enter the coeff. of x2";
cin>>a;
cout<<"\n enter the coeff of x";
cin>>b;
cout<<"\n enter constant term";
cin>>c;
roots(a,b,c,root1,root2);
cout<<"\n root1="<<root1;
cout<<"\n root2="<<root2;
getch ( );
}
void roots(int a, int b, int c, float& x1,float& x2)
{
int d;
d= b*b-4*a*c;
if(d>=0)
        {
        x1=(-b+sqrt(d))/(2*a);
        x2=(-b-sqrt(d))/(2*a);
        }
else
        cout<<"\n roots are not real";
}
```

2.8 फंक्शन के द्वारा भेजा गया मान (Returning Value from a function)

एक फंक्शन हमेशा एक ही प्रकार का मान भेजता है। लेकिन अगर फंक्शन वाइड (void) टाइप का डिक्लेयर है तो यह जरूरी नहीं है कि वह एक भी प्रकार का मान भेजे। यानि फंक्शन कोई भी मान नहीं भेजता है।

फंक्शन के द्वारा जो भी मान भेजा जाता है। वह निम्न प्रकार के हो सकते है।

(i) वाइड (void)

(ii) int

(iii) float

(iv) char

void का उदाहरण–

```
void change ( )
    {
    _____;
    _____;
    }
```

2.9 फंक्शन के द्वारा एरे को बुलाना (Calling function with arrays)

जब भी हम एरे को फंक्शन के द्वारा मुख्य प्रोग्राम से उप प्रोग्राम में भेजते हैं तो हम एरे के प्रथम अवयव का पता (Address), पाइन्टर (Pointer) के द्वारा भेजते हैं न कि सारा एरे। जिससे कम्पाइलर अपने आप एरे के बाकी अवयवों के पते की गणना स्वयं कर लेता है, और मान भी पता कर लेता है।

उदाहरण :– कई विद्यार्थियों की औसत आयु निकालने का प्रोग्राम लिखो?

उत्तर :–

```
#include<iostream.h>
# include <conio.h>
float average( int marks[20], int n);
void main()
{
 int marks[20],i,n;
float av;
cout<<"\n how many students";
cin>>n;
for(i=0;i<n;i++)
        {
        cout<<"\n enter the marks of student"<<(i+1);
        cin>>marks[i];
        }
av=average(marks,n);
cout<<"\n average marks"<<av;
getch ( );
}
float average(int marks[20], int n)
{
float s=0,a;
for(int i=0;i<n;i++)
        {
        s= s+marks[i];
        }
a=s/n;
return a;
```

}

2.10 फंक्शन और वेरियबल का परिक्षेत्र (Scope rule of function & variable)

चार तरह का परिक्षेत्र (Scope) C++ में फंक्शन का आता है। वह है स्थानीय (Local), Function, फाइल (File) और क्लास (Class)

2.10.1 स्थानीय परिक्षेत्र (Local Scope)

जो भी कोई वेरियबल किसी ब्लॉक में डिक्लेयर करेंगे। तब वह वेरियबल और फंक्शन उस ब्लाक का स्थानीय फंक्शन और वेरियबल कहलायेगा।

2.10.2 फंक्शन परिक्षेत्र (Function Scope)

अगर कोई वेरियबल किसी फंक्शन के अंदर डिक्लेयर करेगा तो उसका जो दायरा होगा वह उस फंक्शन के अंदर ही होगा। और उसको उस फंक्शन के अंदर ही प्रयोग कर सकते है।

2.10.3 फाइल परिक्षेत्र (File Scope)

अगर कोई वेरियबल फंक्शन और ब्लांक के बाहर डिक्लेयर किया जाये और इसका प्रयोग ब्लॉक और फंक्शन में कहीं भी किया जाये

2.10.4 क्लास परिक्षेत्र (Class Scope)

अगर कोई वेरियबल क्लास के अंदर डिक्लेयर किया जाता है तो उसका प्रयोग उसी क्लास के अन्तर्गत होता है। अतः उसे क्लास परिक्षेत्र (Class Scope) कहेंगे।

```
# include <iostream.h>
int p,q;
char ch;                फाइल परिक्षेत्र
void main ( )
{
int a, b;          स्थानीय परिक्षेत्र
        void print ( )
        {
        int x;          फंक्शन परिक्षेत्र
        }
        void print ( )
        {
                int q;
        }
}
```

2.11 स्थानीय और सार्वभौम वेरियबल (Local & global variable)

2.11.1 स्थानीय वेरियबल (Local)– स्थानीय वेरियबल वह है जिसका प्रयोग उसी ब्लॉक या फंक्शन में किया जाता है। इस वेरियबल को हम उस उदाहरण

```
#include<iostream.h>
void main()
{
 int a=12;
cout<<a;
}
```

उपरोक्त उदाहरण में वेरियबल a उस ब्लॉक के लिए स्थानीय है। जिसको कि हम केवल Main प्रोग्राम में प्रयोग कर सकते है।

2.11.2 सार्वभौम वेरियबल (Global) सार्वभौम वेरियबल वह है, जिसको हम प्रोग्राम के अंदर कही भी प्रयोग कर सकते है। यानि इस वेरियबल को हम फंक्शन, ब्लॉक या कहीं भी प्रयोग कर सकते है। इसलिए इस तरह के वेरियबल को सार्वभौम वेरियबल (Global) कहते है।

उदाहरण :–

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

उपरोक्त उदाहरण में जो वेरियबल a दर्शाया गया है। वह सार्वभौम वेरियबल है। जिसको कि मुख्य प्रोग्राम में भी प्रयोग कर सकते है और फंक्शन के अंदर भी प्रयोग कर सकते है।

कई बार जब स्थानीय (Local) और सार्वभौम (Global) वेरियबल हमेशा फंक्शन के अंदर छुपा होता है। जिससे कि उसका प्रयोग फंक्शन के अन्दर करने पर स्थानीय वेरियबल ही प्रिंट होगा न कि सार्वभौम वेरियबल। निम्न उदाहरण दिया गया है।

```
#include<iostream.h>
# include <conio.h>
int x=35;
void main()
{
 int x=40;
cout<<x;
getch ( );
}
```

उपरोक्त प्रोग्राम का परिणाम कम्प्यूटर पर 40 मिलेगा।

```
#include<iostream.h>
# include <conio.h>
int x=35;
void main()
{
        int x=40;
        cout<<"\n Global value of x="<<::x;
```

```
        cout<<"\n Local value of x="<<x;
getch ( );
}
```

It will give the out put
Global value of x=35
Local value of x=40

अभ्यास प्रश्न

वस्तुनिष्ठ प्रश्न (Obective Type Questions)

निम्नलिखित प्रश्नों में सही विकल्प का चुनाव कीजिए

1. एक फंक्शन डेटा टाइप जो किसी भी प्रकार का मान रिटर्न नहीं करता है।
   (अ) इण्ट टाइप (ब) void
   (स) फ्लोट (द) डबल
2. फंक्शन को बनाने से फायदा है।
   (अ) समय की बचत (ब) जगह की बचत
   (स) समय और जगह की बचत (द) उपरोक्त में से कोई भी नहीं
3. किसी भी वेरियबल को काल वाई रेफरेन्स बनाने के लिए वेरियबल के नाम से पूर्व चिन्ह जोडते हैं।
   (अ) # (ब) &
   (स) ? (द) उपरोक्त में से कोई नहीं
4. जब लोकल वेरियबल व सार्वभौम वेरियबल का नाम एक हो तो सार्वभौम वेरियबल को लोकल वेरियबल से अलग दियाने के लिऐ चिन्ह प्रयुक्त करेंगे।
   (अ) :: (ब) ;
   (स) : (द) उपरोक्त में से कोई नहीं

लघुत्तरात्मक प्रश्न (Short Answer Type Questions)

1. फंक्शन क्या होता है ?
2. स्थानीय व सार्वभौमिक वेरियवल में से किसका परिक्षेत्र ज्यादा होता है ?
3. किसी भी स्ट्रिंग को रिटर्न करने के लिए फंक्शन का कौनसा टाइप प्रयुक्त किया जाता है ?
4. फंक्शन के टाइप कितने प्रकार के होते है ?
5. फंक्शन प्रोटोटाइप क्या होता है ?

निबन्धात्मक प्रश्न (Essay Type Questions)

1. काल बाई वेल्यू व काल बाई रेफरेन्स में अन्तर लिखिये।
2. C++ भाषा में साधारण ब्याज की गणना करने का प्रोग्राम फंक्शन के द्वारा बनाओ।
3. सामान्य और सार्वभौम वेरियबलों में अन्तर लिखिये।
4. C++ भाषा में एक प्रोग्राम लिखो जो कि किसी द्विघात समीकरण के मूल निकाले।
5. C++ भाषा में एक प्रोग्राम लिखिये जो NCr का मान निकाले।
   (NCr=n का फेक्टोरियल / r का फेक्टोरियल (n-r) का फेक्टोरियल)

उत्तरमाला

1. (ब) 2. (स), 3. (ब), 4. (अ)

3

एरे एवम् स्ट्रकचर
(Array and Structure)

एरे एक ही प्रकार के डाटा समूह को कहते हैं। ये डाटा प्रकार किसी भी प्राथमिक प्रकार जैसे int, float, char आदि के अथवा यूजर द्वारा डिफाइन प्रकार के हो सकते है। इन का प्रयोग structure व object के अन्दर भी किया जा सकता है।

3.1 एरे की आवश्यकता (Need for array)

हम उस स्थिति के बारे में सोचे, जब हमें कहा जाये कि एक प्रोग्राम लिखो जो कि 30 विद्यार्थियों की उम्र को यूजर से पूछे और फिर उनकी औसत आयु निकाले। अगर हम इस प्रोग्राम को बिना एरे के हल करे तो हमें 30 वेरियबल तो 30 विद्यार्थियों की उम्र को रखने के लिए व एक वेरियबल उनकी औसत आयु यानि कुल 31 वेरियबल की आवश्यकता है। इन 31 वेरियबलों का नाम भी अलग–अलग होगा। अब इन 31 वेरियबलों (variable) को प्रोग्राम के अंदर डिकलेयर (Declare) करना बहुत बड़ी समस्या हो जावेगी। तथा समय भी ज्यादा लगेगा। लेकिन यदि हम एरे का प्रयोग करेंगे, जैसे age[30], तब यह 30 वेरियबल अपने आप ही age[0]से age [29] नाम से जाने जायेंगे। और प्रोग्राम को भी सरल बना देंगे।

3.2 एरे के प्रकार (Types of array)

एरे निम्न प्रकार के होते हैं।

(i) एक विमीय एरे (one dimensional array)

(ii) द्वि विमीय एरे (Two dimensional array)

3.3 एरे डिक्लेरेशन (Array Declaration)

एरे को एक नाम से डिक्लेयर करते हैं। और इसके अवयवों को उसके सबस्क्रिप्ट (Subscript) द्वारा प्रयोग करते हैं।

साधारणतया एरे को निम्न प्रकार परिभाषित करते हैं।

Data Type array name [size];

यहां पर Data Type का मतलब एरे का डाटा टाइप,

array name - एरे नाम यूजर द्वारा दिया जायेगा।

Size - संख्या जो कि उसके अवयवों की संख्या दर्शाता है। और अवयवों की संख्या हम integer में देंगे न कि float में।

उदाहरण :–

```
int age [30];
float Marks [50];
```

3.4 एक विमीय एरे में प्रारंभिक मान रखना
(Array Initialization of one dimensional array)

एक विमीय एरे की प्रारंभिक मान उसी तरह रखी जाएगी, जिस तरह दूसरे वेरियबल का मान रखा जाता है। लेकिन एरे का मान { } ब्रेकेट के अंदर रखा जाता हैं।

माना कि आप 5 विद्यार्थियों की प्रारंभिक आयु एरे में रखना चाहते है, तो उसके लिए कम्प्यूटर में लिखेंगे

```
int age [5] = {16, 17, 19, 18, 15};
```

यह निम्न प्रकार अपने आप एरे से संबंधित वेरियबल में मान रख देगा।

age [0] = 16

age [1] = 17

age [2] = 19

age [3] = 18

age [4] = 15

3.5 एरे अवयवों का परिचालन (Manipulations of array element)

3.5.1 जोड़ना :– पहले से ही व्यवस्थित एरे के अंदर नया डाटा को व्यवस्थित जगह पर जोड़ने को एरे के अंदर जोड़ना कहते है।

निम्नलिखित चित्र में एक एरे दर्शाया गया है। अब हम इसमें एक डाटा 35 जो कि अपने व्यवस्थित स्थान पर जोड़ना चाहते है। उसके लिए हम पहले वह स्थिति का पता करेंगे, जहां पर यह नया डाटा जोड़ना है। उसके बाद उसके आगे के सारे अवयव एक स्थिति आगे खिसकेंगे, जिससे वह

स्थान जहां नया डाटा रिक्त हो जायेगा और उसके बाद उस रिक्त स्थान पर नया डाटा जुड़ जाएगा।

0	10	0	10	0	10
1	12	1	12	1	12
2	21	2	21	2	21
3	25	3	25	3	25
4	31	4	31	4	31
5	39	5		5	35
6	43	6	39	6	39
7	45	7	43	7	43
8	50	8	45	8	45
9		9	50	9	50

उदाहरण :–

किसी एरे के अन्दर डाटा को जोड़ने का प्रोग्राम C++ भाषा में लिखें।

```
#include<iostream.h>
# include <conio.h>
int findpos(int [],int,int);
void main()
{
        int a[10],data,n,p;
        cout<<"\n how many number";
        cin>>n;
                for(int i=0;i<n;i++)
                {
                cout<<"\n enter a"<<(i+1);
                cin>>a[i];
                }
        cout<<"\n enter the data which to be inserted";
        cin>>data;
        p=findpos(a,n,data);
                for(i=n;i>p;i++)
```

```
            {
            a[i]=a[i-1];
            }
        a[p]=data;
        n= n+1;
        cout<<"\n The array now is as:\n";
        for(i=0;i<n;i++)
        cout<<a[i]<<" ";
        cout<<"\n";
}
int findpos(int a[], int n, int data)
{
int pos;
if(data < a[0])
    pos=0;
else
    {
    for (int i=0;i<n-1;i++)
        {
        if(a[i] <= data && data < a[i+1])
            {
            pos=i+1;
            break;
            }
        }
        if(i==n-1) pos=n;
    }
return pos;
}
```

3.5.2 विलोपन (Deletion) :–

एक एरे अवयव को पहले से सुव्यवस्थित एरे में से हटाने को एरे अवयव का विलोपन कहते है। निम्नलिखित चित्र में एक एरे के सारे अवयव दर्शाये गये है। उसमें से एरे अवयव 22 को हटाना है। 22 को हटाने के बाद उसके दो अवयव जिनका मान 25 और 30 है। एक–एक अवयव ऊपर खिसकेंगे, जिससे वह रिक्त स्थान को ग्रहण कर लेगा।

0	5	0	5	0	5
1	7	1	7	1	7
2	12	2	12	2	12
3	13	3	13	3	13
4	18	4	18	4	18
5	19	5	19	5	19
6	21	6	21	6	21
7	22	7		7	25
8	25	8	25	8	30
9	30	9	30	9	

उदाहरण :– एक एरे में से एक अवयव को हटाने का प्रोग्राम लिखो?

```
#include<iostream.h>
void main()
{
    int a[10],data,n,p;
    cout<<"\n how many number";
    cin>>n;
    for(int i=0;i<n;i++)
        {
         cout<<"\n enter a"<<(i+1);
        cin>>a[i];
        }
    cout<<"\n enter the data which to be deleted";
    cin>>data;
    for(i=0;i<n;i++)
        {
            if (a[i]==data)
            {
            p=i;
            break;
```

```
                }
            }
    for(i=p;i<n;i++)
            {
            a[i]=a[i+1];
            }
    for(i=0;i<n-1;i++)
            { cout<<"\n"<<a[i]; }
    cout<<"\n The array now is as:\n";
                for(i=0;i<n;i++)
                cout<<a[i]<<" ";
                cout<<"\n";
  getch ( );
}
```

3.5.3 औसत (Average of array elements)

औसत निकालने के लिए सारे एरे अवयवों के मान को जोड़कर उसे उतने ही अवयवों से भाग देने पर जो संख्या आयेगी उसे एरे का औसत कहते हैं।

```
#include<iostream.h>
# include <conio.h>
void main()
{
 int age[30];
int i,n;
float av, s=0;
cout<<"\n how many students but not more than 30";
cin>>n;
for(i=0;i<n;i++)
    {
    cout<<"\nenter the age of student  "<<(i+1);
    cin>>age[i];
    s= s + ageIi];
    }
av= s/n;
cout<<"\n average age="<<av;
getch ( );
}
```

आउटपुट का नमूना :

```
How many students but not more than 30
5
enter the age of student 1 11
enter the age of student 2 12
enter the age of student 3 16
enter the age of student 4 17
enter the age of student 5 18
average age=14.8
```

3.5.4 रेखीय सर्च (Linear Searching) :–

किसी एक दिये हुए एरे में, किसी अवयव का पता करने को सर्चिंग कहते हैं। यदि डाटा उस एरे की सारणी में मिल जाता है, तो कम्प्यूटर उत्तर देगा कि डाटा मिल गया, अन्यथा डाटा नहीं मिला। इस प्रोग्राम के लिए हम एक वूलियन वेरियबल (Boolean Variable) लेंगे। उसके प्रारंभिक मान को हम 0 कर देंगे। इसका मतलब वह अवयव उस सारणी में नहीं है, यह हम मानेंगे। और जैसे ही डाटा उस सारणी में मिल जायेगा। वूलियन वेरियबल का मान 1 हो जायेगा इसका मतलब वह डाटा इस सारणी में उपलब्ध है।

```
#include<iostream.h>
# include <conio.h>
void main()
{
 int age[30],i,data,n, flag=0;
cout<<"\n how many data";
cin>>n;
for(i=0;i<n;i++)
     {
     cout<<"\n enter the age of student"<<(i+1);
     cin>>age[i];
     }
cout<<"\n enter the age which to be found in the list";
cin>>data;
for(i=0;i<n;i++)
     {
     if(age[i]==data)
            {
            flag=1;
            break;
            }
     }
if(flag==1)
     cout<<"\n the given age in the list";
else
     cout<<"\n the given age is not in the list";
getch ( );
}
```

3.5.5 अधिकतम और न्यूनतम मान निकालना (Finding maximum/minimum value)

उदाहरण :– किसी एरे के अन्दर अधिकतम और न्यूनतम मान निकालने का प्रोग्राम लिखिये।

```
#include<iostream.h>
#include<stdio.h>
# include <conio.h>
void main()
{
int n,max,min,i,marks[30];
```

```
cout<<"\n how many marks you enter";
cin>>n;
for(i=0;i<n;i++)
    {
     cout<<"\n enter the marks of student"<<(i+1);
    cin>>marks[i];
    }
max=marks[0];
min=marks[0];
for(i=1;i<n;i++)
    {
    if(marks[i] > max ) max=marks[i];
    if(marks[i] < min ) min=marks[i];
    }
cout<<"\n Maximum marks"<<max;
cout<<"\n Minimum marks"<<min;
getch ( );
}
```

3.6 स्ट्रिंग को डिक्लेयर करना (Declaration of String)

C++ भाषा में कोई भी स्ट्रिंग डाटा टाइप नहीं है। जिसका की सीधा प्रयोग किया जा सके। इसलिए स्ट्रिंग के डिक्लेयर करने के लिए हम करैक्टर का एरे डिक्लेयर करेंगे जिसका कि अंतिम करैक्टर '\0' खाली करैक्टर (Null character) हो। जब भी हम स्ट्रिंग डिक्लेयर करेंगे तो उस करैक्टर के एरे की संख्या हमेशा जितने स्ट्रिंग के अंदर करैक्टर हैं। उससे एक ज्यादा डिक्लेयर करेंगे, जिससे कि उसका अंतिम करैक्टर उस एरे के अंतिम अवयव में स्टोर हो जाये।

यदि हमें एक स्ट्रिंग डिक्लेयर करनी है। जिसके करैक्टर की संख्या 20 है तो उसे निम्न प्रकार से डिक्लेयर करेंगे।

char Name [21];

उदाहरण :– C++ भाषा में एक प्रोग्राम लिखो जो कि एक स्ट्रिंग उलटा और सीधा पढ़ने पर, एक ही स्ट्रिंग पढ़ी जाये। (उदाहरण मलयालम, सरस आदि)

```
#include<iostream.h>
# include <conio.h>
#include<string.h>
#include<stdio.h>
void main()
{
 char s[20];
 int i,j,n,flag=1;
 cout<<"\n enter string";
 gets(s);
 n=strlen(s);
for(i=0,j= n-1; ((i<= n/2) && (j>= n/2));i++,j—)
```

```
{
    if (s[i]!=s[j])   flag=0;
    break;
}
if(flag==1)
 cout<<"\n string is palindrome";
 else
     cout<<"\n string is not palindrome";
getch ( );
 }
```

3.7 स्ट्रिंग का प्रारंभिक मान (Initialization of string)

कई बार जब हम स्ट्रिंग का मान यूजर से नहीं पूछना चाहते हैं और सीधे ही प्रोग्राम में हम उसका प्रारंभिक मान रखना चाहते हैं। उसके लिये हम प्रोग्राम में निम्न प्रकार लिखेंगे।

char name [20] = "prateek"

उपरोक्त डिक्लरेशन अपने आप ही एरे का मान निम्न प्रकार रख देगा।

```
name [0] = 'p'
name [1] = 'r'
name [2] = 'a'
name [3] = 't'
name [4] = 'e'
name [5] = 'e'
name [6] = 'k'
name [7] = '\0'
```

कम्पाइलर अपने आप ही अंतिम करैक्टर को नल करैक्टर (Null character) से जोड़ देता है।

3.8 स्ट्रिंग में स्वर व व्यंजनों की गणना करना (Counting Vowels/Consonants/digit/special characters)

एक स्ट्रिंग दी हुई है। उसमें आप स्वरों की गणना करना चाहते है। तो उसके लिए हम स्ट्रिंग एरे के प्रत्येक अवयव की तुलना स्वरों (a, e, i, o, u, A, E, I, O, U) से करेंगे। और अगर स्वर मिल जाते है तो हम पहले से ही डिक्लेयर काउण्टर वेरियबल (Counter variable) का मान बढ़ा देंगे जो कि स्वरों की गणना करता है। यदि उस स्ट्रिंग में स्वर नहीं मिलता है तो, दूसरा काउण्टर वेरियबल (Counter variable) जो व्यंजन की गणना करता है। उसका मान बढ़ जाता है। इस प्रकार हम एक प्रोग्राम के द्वारा स्ट्रिंग में स्वर व व्यंजन की गणना करते हैं।

उदाहरण :– एक स्ट्रिंग के अन्दर स्वर व व्यंजनों की गणना करने का प्रोग्राम लिखिये।

```
#include<iostream.h>
#include<stdio.h>
#include<string.h>
# include <conio.h>
void main()
{
```

```
char st[80];
int i,n,v=0,c=0;
cout<<"\n enter the string";
gets(st);
n=strlen(st);
for(i=0;i<n;i++)
    {
    switch(st[i])
    {
    case 'a' :
    case 'A' :
    case 'e' :
    case 'E' :
    case 'i' :
    case 'I' :
    case 'o' :
    case 'O' :
    case 'u' :
    case 'U' :  ++v;
                break;
    default: ++c;
    }
  }
  cout<<"\n number of vowels"<<v;
  cout<<"\n number of consonants"<<c;
getch ( );
  }
```

3.9 इजएलनम (isalnum)

इजएलनम(isalnum) फंक्शन ctype.h फाइल में उपलब्ध है। इस फंक्शन (Function) का कार्य एक करैक्टर की यह जांच करता है कि वह करैक्टर एल्फाबेट और नंबर है या नहीं है। एल्फाबेट का मान A से Z या a से z तक तथा नंबर का मान 0 से 9 तक हो सकता है।

```
#include<iostream.h>
#include<ctype.h>
#include<stdio.h>
# include <conio.h>
void main()
{
 int ch;
 cout<<"\n enter a character";
 ch=getchar();
 if (isalnum(ch))
         cout<<"\n the character is alphanumeric";
 else
         cout<<"\n no alphanumeric";
```

```
}
```

3.10 इजएल्फा (isalpha)

यह फंक्शन भी isalnum की तरह है लेकिन यह फंक्शन किसी दिए हुए करैक्टर की जांच करता है कि वह एल्फाबेट (alphabet) है या नहीं।

```
#include<iostream.h>
#include<ctype.h>
#include<stdio.h>
# include <conio.h>
void main()
{
 int ch;
 cout<<"\n enter a character";
 ch=getchar();
 if (isalpha(ch))
         cout<<"\n the character is alphabet";
 else
         cout<<"\n the character is not alphabet";
getch ( );
 }
```

3.11 इजडिजीट (isdigit)

यह फंक्शन भी ctype.h हैडर फाइल में उपलब्ध है। इस फंक्शन का काम किसी करैक्टर को यह पता करना है कि वह करैक्टर नंबर है या नहीं। नंबर का मान 0 से 9 तक होता है।

```
#include<iostream.h>
#include<ctype.h>
#include<stdio.h>
# include <conio.h>
void main()
{
 int ch;
 cout<<"\n enter a character";
 ch=getchar();
 if (isdigit(ch))
         cout<<"\n the character is digit";
 else
         cout<<"\n the character is not digit";
getch ( );
 }
```

3.12 इजलोवर (islower)

यह फंक्शन Ctype.h हैडर फाइल में उपलब्ध है। इसका काम एक करैक्टर की यह जांच करना है कि वह करैक्टर छोटे करैक्टर (a-z) में है या नहीं।

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

```
#include<stdio.h>
# include <conio.h>
void main()
{
 int ch;
 cout<<"\n enter a character";
 ch=getchar();
 if (islower(ch))
         cout<<"\n the character is in lower case";
 else
         cout<<"\n the character is not in lower case";
getch ( );
 }
```

3.13 इजअपर (isupper)

यह फंक्शन भी इजलोवर की तरह है। लेकिन यह करैक्टर की जांच करता है कि करैक्टर बडे करैक्टर (A-Z) में है या नहीं।

```
#include<iostream.h>
#include<ctype.h>
#include<stdio.h>
# include <conio.h>
void main()
{
 int ch;
 cout<<"\n enter a character";
 ch=getchar();
 if (isupper(ch))
         cout<<"\n the character is in upper case";
 else
         cout<<"\n the character is not in upper case";
getch ( );
 }
```

3.14 टू लोवर (tolower)

इस फंक्शन का काम बडे करैक्टरों को छोटे करैक्टरों (small character) में परिवर्तित करना है।

```
#include<iostream.h>
#include<ctype.h>
#include<stdio.h>
# include <conio.h>
void main()
{
 int ch;
 cout<<"\n enter an upper case character";
 ch=getchar();
 if (isalpha(ch))
         if(isupper(ch))
```

```
        {
        cout<<"\n the character is in lower case is";
        putchar(tolower(ch));
        }
        else
        cout<<"\n The character is not an upper case";
  else
  cout<<"\n the character is not alphabet";
getch ( );
  }
```

3.15 टू अपर (toupper)

यह फंक्शन भी ctype.h हैडर फाइल में उपलब्ध है तथा इसका काम छोटे करैक्टरों को बड़े करैक्टरों में परिवर्तित करना है।

```
#include<iostream.h>
#include<ctype.h>
#include<stdio.h>
# include <conio.h>
void main()
{
 int ch;
 cout<<"\n enter a lower case character";
 ch=getchar();
 if (isalpha(ch))
        if(islower(ch))
        {
        cout<<"\n the character is in upper case is";
        putchar(toupper(ch));
        }
        else
        cout<<"\n The character is not a lower case";
  else
        cout<<"\n the character is not alphabet";
getch ( );
  }
```

3.16 एसटीआर कॉपी (strcpy)

इस फंक्शन का काम एक स्ट्रिंग को दूसरे स्ट्रिंग में कॉपी करना है। इसका प्रारूप इस प्रकार है।

```
strcpy (destination, source)
```

यहां पर destination वह स्ट्रिंग है जिसके अंदर कॉपी करना है। तथा source वह स्ट्रिंग है जिसको कि कॉपी करना है।

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

```
#include<stdio.h>
#include<string.h>
# include <conio.h>
void main()
{
 char st[20],dt[20];
 cout<<"\n enter string : ";
 gets(st);
 strcpy(dt,st);
 cout<<"\n orignal string as: "<<st;
 cout<<"\n copied string as: "<<dt;
getch ( );
 }
```

3.17 एसटीआर कैट (strcat)

यह फंक्शन string.h हैडर फाइल में उपलब्ध है। Concatenation का अर्थ है जोडना, strcat में केट (cat) शब्द Concatenation से लिया गया है। इसलिए strcat का मतलब हुआ दो स्ट्रिंग को जोड़ना और एक नयी स्ट्रिंग बनाना। यह फंक्शन पहले से दी हुई स्ट्रिंग के आगे, जिस भी स्ट्रिंग को जोड़ना है। उसको पहले वाली स्ट्रिंग के आगे जोड़ देता है। इसका प्रारूप इस प्रकार है।

strcat (S1, S2);

यहां पर S1 व S2 दो अलग–अलग स्ट्रिंग है और परिणाम S1 स्ट्रिंग में जुड़ जाता है। और स्टोर हो जाता है।

```
#include<iostream.h>
#include<ctype.h>
#include<stdio.h>
#include<string.h>
# include <conio.h>
void main()
{
char st[20]="Rajasthan ";
char dt[20]="Board";
strcat(st,dt);
cout<<"\n Orignal string as: "<<st;
cout<<"\n Copied string as: "<<dt;
getch ( );
}
```

3.18 एस टी आर लेन (strlen)

यह फंक्शन string.h हैडर फाइल में उपलब्ध है। तथा इसका कार्य किसी दी हुई स्ट्रिंग की लम्बाई यानि स्ट्रिंग के अंदर करैक्टरों की संख्या की गणना करने के काम आती है।

```
#include<iostream.h>
#include<string.h>
# include <conio.h>
void main()
{
char s1[20]="Computer Science";
int n;
```

```
n=strlen(s1);
cout<<"\n length is:"<<n;
getch ( );
}
```

यह निम्नलिखित आउटपुट देगा।

length is : 16(including the blank space)

3.19 एस टी आर सी एम पी (strcmp)

यह फंक्शन भी string.h हैडर फाइल में उपलब्ध है। इसका काम दो स्ट्रिंग S1 व S2 की तुलना करना है। यदि S1 स्ट्रिंग, S2 स्ट्रिंग से छोटी है तो इसका मान 0 से कम अर्थात् ऋणात्मक मान आयेगा। और यदि स्ट्रिंग S1 व स्ट्रिंग S2 दोनों बराबर है तो इसका मान शून्य आयेगा। और यदि स्ट्रिंग S1 स्ट्रिंग S2 से बड़ी है तो फंक्शन का मान शून्य से ज्यादा अर्थात् धनात्मक आयेगा।

```
#include<iostream.h>
#include<ctype.h>
#include<stdio.h>
#include<string.h>
# include <conio.h>
void main()
{
char s1[20]="Raja";
char s2[20]="Raja";
if(strcmp(s1,s2)==0)
        cout<<"\nThe string are equal";
else
        cout<<"\n the string are not equal";
getch ( );
}
```

3.20 एसटीआर सीएमपीआई (strcmpi)

यह string.h हैडर फाइल का फंक्शन है। तथा एसटीआरसीएमपी से मिलता जुलता है। लेकिन यह फंक्शन जब दो स्ट्रिंगों की तुलना करता है तो यह नहीं देखता है कि जो करेक्टर लिखे गये है। वह बड़े अक्षरों में लिखे है, या छोटे अक्षरों में लिखे है।

उदाहरण के लिये हम दो स्ट्रिंग S1 = "Raja" और दूसरी स्ट्रिंग S2 = "raja" ले और फिर हम strcmpi फंक्शन का प्रयोग करे तो हम पायेंगे कि दोनों स्ट्रिंग S1 व S2 समान है। जबकि इन्हीं दो स्ट्रिंगों को लेकर हम strcmp( ) फंक्शन का प्रयोग करे तो हम पायेंगे कि उसका उत्तर आयेगा कि ये दोनों स्ट्रिंग S1 व S2 समान नहीं है। (क्योंकि उसने बड़े R व छोटे r को अलग माना है)

3.21 मैथ हैडर फाइल (math.h header file)

3.21.1 एफ एबसल्यूट fabs( ) यह फंक्शन किसी (float) फ्लोट नम्बर का मान निकालने के काम आता है। इस फंक्शन का प्रयोग करते समय हमें math.h हैडर फाइल को प्रोग्राम में जोड़ना होता है।

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

```
# include <conio.h>
void main ( )
{
  float number = -1234.00;
  cout<<"absolute value: "<<fabs(number);
   return (0);
getch ( );
}
```

3.21.2 लॉग (log ( )) :–

यह फंक्शन किसी धनात्मक नम्बर का लॉग निकालने के काम आता है। जिसका कि आधार e हो।

उदाहरण :–

```
#include<iostream.h>
#include<math.h>
# include <conio.h>
void main()
{
 int x=100,y;
y=log(x);
cout<<y;
getch ( );
}
output
2.302585
```

3.21.3 पाव (pow()) :–

यह फंक्शन किसी नम्बर की घात (Power) निकालने के काम आता है।

उदाहरण के लिए हमें x^y का मान निकालना हो तो हम इस फंक्शन को C++ भाषा में लिखेंगे pow (x,y)

3.21.4 स्क्वायर रूट (sqrt ( )) :–

यह फंक्शन किसी नम्बर का वर्गमूल निकालने के काम आता है।

उदाहरण के लिए यदि हमें 16 का वर्गमूल निकालना है तो हम C++ भाषा में इस प्रकार लिखेंगे।

```
x=16,
y=sqrt (x);
cout <<y;
```

3.21.5 साइन (sin( )) :–

साइन फंक्शन किसी कोण का साइन (sine) मान निकालने के काम आता है। लेकिन उसके पहले हम दिए हुए कोण को रेडियन में बदलने के लिए 3.14 से गुणा करके जो भी संख्या आयेगी, उसको 180 से भाग देंगे।

उदाहरण :– 90^0 कोण के साइन का मान ज्ञात करो।

```
#include<iostream.h>
#include<math.h>
# include <conio.h>
void main()
{
 float x;
 x=sin(90*3.14/180);
 cout<<x;
getch ( );
}
```

3.21.6 कॉस (cos ( )) :–

कॉस फंक्शन, साइन फंक्शन की तरह ही है। केवल अंतर इतना ही है कि यह कोण का cosine मान निकालता है।

3.21.7 एवस्ल्यूट (abs) :–

एवस्लूयट फंक्शन किसी मान का परिमाण निकालने के लिए प्रयोग लिया जाता है। और किसी भी मान का परिमाण हमेशा एक धनात्मक नम्बर आयेगा।

x=-3; का एवस्ल्यूट मान निकालने के लिए निम्न प्रकार लिखेंगें।

Y=abs(x);

3.22 स्टेडिलिव हैडर फाइल (stdlib.h) :-

कई बार प्रोग्राम के द्वारा हम ऐसा नम्बर निकालना चाहते है, जो जब जब भी प्रोग्राम को निष्पादित (Execute) करे तो हर बार अलग–अलग उत्तर दे। जो कि साधारण लाटरी या किसी खेल के प्रोग्राम में काम आता है। इस प्रकार के प्रयोग में हम रेण्डमाईज (randomize) और रेण्ड (rand) फंक्शन का प्रयोग करते हैं। इन फंक्शनों का प्रयोग करते समय हम stdlib.h हैडर फाइल प्रोग्राम के अंदर जोड़ते है।

उदाहरण :–

```
#include<iostream.h>
#include<stdlib.h>
# include <conio.h>
void main()
{
 int n;
 randomize();
 n=(int)(rand()*10);
 cout<<n;
getch ( );
}
```

उपरोक्त प्रोग्राम को जब भी हम कम्प्यूटर पर रन करेंगे तो हमेशा हमको जो भी नम्बर मिलेगा, वह 0 से 9 तक ही मिलेगा। और अगर हम इस प्रकार का प्रोग्राम बनाना चाहे जिसमें कि वह जो नम्बर दे वह 0 से 99 के बीच हो तो हम उपरोक्त प्रोग्राम में 10 की जगह 100 लिख देंगे।

व खोलते समय स्ट्रीम में डिक्लेयर करेंगे

3.23 उपयोकर्ता द्वारा परिमाषित डाटा टाइप (User Defined Data Types)

3.23.1 टाइपडैफ (Typedef) :–

C++ में मौजुदा डाटा टाइप के नाम परिवर्तन की सुविधा प्रदान की गई है। कीवर्ड (Keyword) typedef की मदद से किसी टाइप विशेष को एक नया नाम (पर्यायवाची या उपनाम) दिया जा सकता है। एक typedef स्टेटमेंट एक नया डाटा टाइप डिफाइन नहीं करता है, अपितु यह मौजुद डाटा टाइप को एक पर्यायवाची या उपनाम देता है।

सिंटेक्स निम्न प्रकार है :–

```
typedef type alias;
```

यहाँ type दी गई टाइप है तथा alias उसका नया नाम है।

उदाहरण :–

```
typedef long integer;
typedef double Real;
integer K=45;
const real PI=3.14159
```

3.23.2 # डिफाइन डाइरेकटिव (# define directive) :–

C++ में प्रीप्रोसेसर डाइरेकटिव **# define** का उपयोग कर कोन्सटेंट (constant) डिफाइन किये जा सकते हैं।यह डाइरेकटिव एक टेक्सट फरेज (text pharse) तथा आइडेंटिफायर (identifier) के मध्य एक समानता (equivalence) उत्पन्न करता है।

```
#define PI 3.14159
```

प्रोग्राम के प्रारंभ में उपस्थित यह स्टेइमेंट यह इंगित करता है कि पुरे प्रोग्राम में जहाँ भी आइडेंटिफायर (identifier) PI उपस्थित है, उसे टेक्सट फैरज (text pharse) 3.14159 से प्रतिस्थपित किया जाना है। #define का उपयोग कर डाटा टाइप डिफाइन नहीं किया जा सकता है।

3.24 द्विविमीय एरे (Two dimensional array)

वह एरे जिसमें दो अवयव काम में लिए जाते हों एक अवयव तो लाइन को तथा दूसरा अवयव कॉलम को दर्शाता है। द्विविमीय एरे का प्रयोग किसी टेबल की तरह डाटा को दर्शाने या मैट्रिक्स संबंधित गणना के लिए करते हैं।

उदाहरण के लिए एक कम्पनी तीन तरह के बल्ब जिनमें 15 वॉट, 60 वॉट तथा 100 वॉट के बल्ब का निर्माण करती है। और इन भिन्न भिन्न प्रकार के बल्बों की मूल्य अलग–अलग राज्य में अलग

अलग है। इस तरह के डाटा को दर्शाने के लिए द्विविमीय एरे की आवश्यकता होगी जो इस तरह दर्शायी जावेगी।

	15W	60W	100W
राजस्थान	12	15	17
हरियाणा	11	14	18
यू०पी०	13	16	17

3.25 द्विविमीय एरे को डिक्लेयर करना (Declaration of two dimension array)

द्विविमीय एरे को डिक्लेयर करने का सामान्य तरीका निम्न है।

Data type arrayname[rows] [columns];

Data type	-	एरे के नाम का डाटा टाइप को दर्शाता है।
array name	-	जो भी एरे का नाम प्रयोगकर्ता देना चाहे।
row	-	लाइनों की संख्या को दर्शाता है।
Column	-	कॉलम संख्या को बताता है।

int price [4] [5];

यह द्विविमीय एरे को दर्शा रहा है जो कि integer Type का है तथा इसका वेरियबल का नाम price है। तथा लाइनों की संख्या 4 तथा कॉलमों की संख्या 3 है।

उदाहरण :– एक प्रोग्राम लिखो जो तीन तरह के बल्बों को चार राज्यों में अलग–अलग मूल्यों को स्टोर करता हो।

```
#include<iostream.h>
#include<iomanip.h>     /* it is used for setw() */
# include <conio.h>
void main()
{
int price[3][4];
int i,j;
for( i =0;i<3;i++)
        {
        for(j=0;j<4;j++)
                {
                cout<<"\n enter the price of "<<(i+1)<<"bulbs"<(j+1)<<"states";
                cin>>price[i][j];
                }
        }
cout <<"\n the price are as \n";
for(i=0;i<3;i++)
        {
        for(j=0;j<4;j++)
```

```
            {
            cout<<setw(8)<<price[I][j];
            }
        cout<<"\n";
        }
getch ( );
}
```

3.26 द्विविमीय एरे का प्रारंभिक मान रखना (Initialization of two Dimensional array)

एक विमीय एरे की तरह ही द्विविमीय एरे का प्रारंभिक मान रखा जायेगा। द्विविमीय एरे में प्रारंभिक मान रखते समय प्रथम अवयव को लिखना जरूरी नहीं है। उसे खाली छोड़ सकते है। लेकिन द्वितीय अवयव को लिखना जरूरी है।

int price[ ] [3]= {4, 5, 6, 7, 4, 3};

यह द्विविमीय एरे निम्न प्रकार मान रखेगा।

price [0] [0]=4
price [0] [1]=5
price [0] [2]=6
price [1] [0]=7
price [1] [1]=4
price [1] [2]=3

3.27 एरे अवयवों को इनपुट करना

(Inputting array elements)

Ex. Write a program of addition of two matrix.

```
#include<iostream.h>
#include<iomanip.h>
# include <conio.h>
void main()
{
int a[10[10],b[10[10],c[10[10];
int m,n,i,j;
cout<<"\n enter the number of rows";
cin>>m;
cout<<"\n enter the number of cols";
cin>>n;
for(i=0;i<m;i++)
{
        for(j=0;j<n;j++)
        {
        cout<<"\n enter the value of a "<<(i+1)<<(j+1);
        cin>>a[i][j];
        }
```

```
}
for(i=0;i<m;i++)
        {
        for(j=0;j<n;j++)
        {
        cout<<"\n enter the value of b "<<(i+1)<<(j+1);
        cin>>b[i][j];
        }
        }
for(i=0;i<m;i++)
{
        for(j=0;j<n;j++)
        {
        c[i][j]= a[i][j] + b[i][j];
        }
}
for(i=0;i<m;i++)
        {
        for(j=0;j<n;j++)
        {
        cout<<setw(8)<<c[i][j];
        }
        cout<<"\n";
        }
getch ( );
}
```

3.28 विकर्ण अवयव (Diagonal element)

एक मैट्रिक्स दिया हुआ है। और हमें उसके विकर्णो के अवयवों का जोड़ निकालना है। तो सबसे पहले तो यह तब ही संभव है जबकि वह मैट्रिक्स वर्ग मैट्रिक्स (Square matrix) हो। अर्थात् वह मैट्रिक्स जिसमें लाइनों और कॉलमों की संख्या समान हो।

वर्ग मैट्रिक्स दो तरह के विकर्ण रखता है। एक तो मुख्य (Main) विकर्ण तथा दूसरा उप विकर्ण (sub digonal)

```
#include<iostream.h>
# include <conio.h>
void main()
{
int a[10[10];
int m,n,i,j,s=0;
cout<<"\n enter the number of rows";
cin>>m;
cout<<"\n enter the number of cols";
cin>>n;
```

```
for(i=0;i<m;i++)
    {
    for(j=0;j<n;j++)
        {
        cout<<"\n enter the value of a"<<(i+1)<<(j+1);
        cin>>a[i][j];
        }
    }
for(i=0;i<m;i++)
    {
    for(j=0;j<n;j++)
        {
        if((i==j) || (i+j==m-1))
        s= s+ a[i][j];
        }
    }
cout<<"\n sum of diagonal is = "<<s;
getch ( );
}
```

3.29 अधिकतम और न्यूनतम मान निकालना (Finding Maximum/Minimum value)

उदाहरण :– किसी द्विविमीय एरे में से न्यूनतम और अधिकतम मान निकालने का प्रोग्राम लिखें।

```
#include<iostream.h>
# include <conio.h>
void main()
{
int a[10[10];
int m,n,i,j,max,min;
cout<<"\n enter the number of rows";
cin>>m;
cout<<"\n enter the number of cols";
cin>>n;
for(i=0;i<m;i++)
    {
    for(j=0;j<n;j++)
        {
        cout<<"\n enter the value of a "<<(i+1)<<(j+1);
        cin>>a[i][j];
        }
    }
max=a[0][0];
```

```
min=a[0][0];
for(i=0;i<m;i++)
    {
    for(j=0;j<n;j++)
        {
        if(a[i][j] > max)  max=a[i][j];
        if(a[i][j] < min)  min=a[i][j];
        }
    }
cout<<"\n  Maximum value in a matrix is = "<<max;
cout<<"\n  Minimum value in a matrix is = "<<min;
getch ( );
}
```

3.29 स्ट्रकचर (Structure)

स्ट्रकचर (Structure) सामान्य चरों का एक संग्रह (Collection) है। एक स्ट्रकचर में विभिन्न प्रकार (type) के चर उपस्थित हो सकते हैं। यह स्थिति एरे से भिन्न है क्योंकि एरे समान प्रकार के चरों का संग्रह (Collection) है।

C भाषा से अनुरुपता रखने के लिए C++ भाषा में struct कीवर्ड (keyword) रखा गया है। जो C++ में स्ट्रकचर (Structure) डिफाइन करने की सुविधा प्रदान करता है।तथापि C++ में स्ट्रकचर (Structure), C++ की क्लास (Class) के समान है।C++ में स्ट्रकचर (Structure) एवम् C++ की क्लास (Class) में मुख्य अंतर यह है कि C++ स्ट्रकचर (Structure), C++ की ऐसी क्लास (Class) है जिसके सभी सदस्य पब्लिक (Public) है तथा फंक्शन (Function) जिसके सदस्य नहीं हो सकते है।

3.29.1 स्ट्रकचर डिफाइन व डिकलेयर करना (Defining and Declaring Structure)

सिंटेक्स (Syntax) :-

स्ट्रकचर डिफाइन करने के लिए

```
struct structrue_name
{
 data type variable1;
 data type variable2;
 .
 .
data type variablen;
};
```

स्ट्रकचर डिकलेयर करने के लिए

```
structure_name structure_variable1,structure_variable1;
```

उदाहरण (Example) :-

```
struct Employee
{
 char name[20];
 int age;
 float salary;
};
Employee e1;
```

उपरोक्त उदाहरण में हमने Employee नाम का एक स्ट्रकचर बनाया है। name[20] एक char टाइप का चर है, age int टाइप का चर है तथा salary float टाइप का चर है। जो स्ट्रकचर employee के सदस्य (member) है।

Employee e1;

उपरोक्त स्टेटमेंट में हमने स्ट्रकचर टाइप का एक चर e1 डिफाइन किया है। e1 चर की यह परिभाषा मैमोरी में 28 बाइट स्थान संरक्षित करता है।

एक char टाइप का चर को मैमोरी में संरक्षित करने के लिए एक बाइट मैमोरी स्पेस (Memory Space) की आवश्यकता होती है। चूंकि चर name एक char टाइप का ऐसा एरे है जिसकी विमा (dimension) 20 है।अतः name चर के लिए मैमोरी में 1 बाइट X 20 = 28 बाइट स्थान संरक्षित होगा। age int टाइप का चर है जिसे मैमोरी में संरक्षित करने के लिए चार बाइट मैमोरी स्पेस की आवश्यकता होती है तथा salary float टाइप का चर है,इसे भी मैमोरी में संरक्षित करने के लिए चार बाइट मैमोरी स्पेस की आवश्यकता होती है।

चर e1 Employee टाइप का स्ट्रकचर है अतः डिकलेरेशन (Employee e1) के द्वारा मैमोरी में कुल 20+4+4=28 बाइट स्थान संरक्षित होगा। यह गणना यह मान कर की गई है कि कम्प्यूटर एक 32 बिट मशीन है।

3.29.2 स्ट्रकचर के सदस्यों को एक्सेस करना (Accesing Structure Members)

एकबार स्ट्रकचर चर डिफाइन करने के पश्चात डोट ऑपरेटर (Dot Operator) का उपयोग कर उस स्ट्रकचर के सदस्यों को एक्सेस (access) किया जा सकता है जैसे –

```
e1.name= "SHAILENDRA";
e1.Age=42;
e1.salary=40000.00;
```

डोट आपरेटर (Dot Operator) का वास्तविक नाम मेम्बर एक्सेस ऑपरेटर (Member Access Operator) है।

उदाहरण (Example) :-

```
cout << "\n Name" << e1.Name;
```

उपरोक्त स्टेटमेंट employee स्ट्रकचर के सदस्य Name का मान आउटपुट के रुप में प्रदान करेगा।

3.29.3 स्ट्रकचर की डेफिनेशन व डिकलेयरेशन को मिलाना (Combining Definition and Declaration of Structure)

सिंटेक्स (Syntax) :-

```
struct
{
data type variable1;
data type variable2;
.
.
data type variablen;
}structure_variable1;
```

उदाहरण (Example) :-

```
struct
{
char name[20];
int age;
float salary;
}e1,e2;
```

उपरोक्त उदाहरण में स्ट्रकचर की परिभाषा (definition) के लिए अलग से स्टेटमेंट नहीं दिया गया है।

3.29.4 स्ट्रकचर के सदस्यों को इनीशीयलाइज करना (Initializing Structure Members)

सिंटेक्स (Syntax) :-

```
struct structrue_name
{
data type variable1;
data type variable2;
```

```
.
.
data type variablen;
};
structure_name structure_variable_name={value1,value2...valuen};
```

उदाहरण (Example) :-

```
struct Employee
{
 char name[20];
 int age;
 float salary;
};
Employee e1={"SHAILENDRA",42,40000.00};
Employee e2={"VINEET",41,80000.00};
```

3.29.5 नेस्टेड स्ट्रकचर (Nested Structure)

सिंटेक्स (Syntax) :-

```
struct structrue_name1
{
 data type variable1;
 data type variable2;
 .
 .
data type variablen;
};
struct structrue_name2
{
 data type variable1;
 data type variable2;
structure_Name1 Structure_variable1;
};
```

उदाहरण (Example) :-

```
struct Employee
{
 char name[20];
 int age;
 float salary;
};
struct Company
{
char Company_Name[20];
Employee e1,e2;
};
```

3.29.6 स्ट्रकचरो का एरे (Array of Structure)

सिंटेक्स (Syntax) :-

```
struct structrue_name
{
data type variable1;
 data type variable2;
 .
.
data type variablen;
};
structure_name structure_variable_name[dimension];
```

उदाहरण (Example) :-

```
struct Employee
{
 char name[20];
 int age;
 float salary;
};
```

Employee e[100];

उपरोक्त उदाहरण के स्टेटमेंट –

Employee e[100];

में हमने Employee स्ट्रकचर टाइप का चर डिकलेयर किया है, जिसकी विमा 100 है।

3.29.7 स्ट्रकचर को वेल्यु/आरग्युमेंट के रुप में एक फंक्शन को भेजना (Passing Structure to Function as Value/argument)

एक पूरा स्ट्रकचर (Structure) आरग्युमेंट (argument) के रुप में एक फंक्शन को भेजा जा सकता है।

<u>उदाहरण (Example)</u> :-

```
#include<iostream>
# include <conio.h>
# include <stdio.h>
using namespace std;
struct distance
{
int meter;
float cm;
};
void display( distance d ) ;
void main( )
{
distance d1,d2;
cout << "Enter meter:";
cin >> d1.meter;
cout << "Enter cm:";
cin >> d1.cm;
cout << "\n Enter meter:";
cin >> d2.meter;
cout << "Enter cm:";
cin >> d2.cm;
cout << "\n d1=";
display(d1);
cout << "\n d2=";
```

```
display(d2);
cout << endl;
return 0;
getch ( );
}
void display( distance dd)
{
cout << dd.meter << " m " << dd.cm << " cm";
}
```

उपरोक्त प्रोग्राम में main( ) पार्ट दो दूरियों meter तथा cm फॉमेट नें उपयोगकर्ता से प्राप्त करता है और उन्हें दो स्ट्रकचर d1 तथा d2 में रखता है। फिर यह फंक्शन display( ) को कॉल करता है जो स्ट्रकचर distance को स्ट्रकचर के रुप में स्वीकार करता है। यह फंक्शन दूरियों को सामान्य फॉमेट में डिस्प्ले करता है।

Output :-

```
Enter meter:8
enter cm:2.5

Enter meter:9
Enter cm:4.5

d1=8 m 2.5 cm
d2=9 m 4.5 cm
```

3.29.8 फंक्शन रिर्टनिंग स्ट्रकचर (Function returning Structure)

पिछले खण्ड में हम पढ चुके है कि कैसे एक पूरा स्ट्रकचर (Structure) फंक्शन के आरग्युमेंट (argument) के रुप में भेजा जा सकता है। इसी प्रकार स्ट्रकचर को फंक्शन की रिर्टन वेल्यु के रुप में भी चुन सकते है।

उदाहरण (Example) :-

```
#include <iostream>
# include <conio.h>
# include <stdio.h>
using namespace std;
struct distance
```

```
{
int meter;
float cm;
};
distance add_distance( distance& d1 , distance& d2 );
void display( distance d);
int main( )
{
distance d1,d2,d3;
cout << "Enter meter:";
cin >> d1.meter;
cout << "Enter cm:";
cin >> d1.cm;
cout << "\n Enter meter:";
cin >> d2.meter;
cout << "Enter cm:";
cin >> d2.cm;
d3 = add_distance(d1, d2);
cout << endl;
display(d1);
cout << " + ";
display(d2);
cout << " = ";
display(d3);
cout << endl;
return 0;
}
void display( distance dd)
{
cout << dd.meter << " m " << dd.cm << " cm";
}
void add_distance( distance dd1 , distance dd2)
```

```
{
dd3.cm = dd1.cm + dd2.cm;
dd3.meter=0;
if ( dd3.cm >= 10.0)
        {
        dd3.cm -= 10.0 ;
        dd3.meter ++ ;
        }
dd3.meter += dd1.meter + d d2.meter
return dd3;
getch ( );
}
```

Output :-
Enter meter:8
enter cm:2.5

Enter meter:9
Enter cm:4.5

8 m 2.5 cm + 9 m 4.5 cm = 17 m 7 cm

उपरोक्त प्रोग्राम में main( ) पार्ट दो दूरियों meter तथा cm फारमेट नें उपयोगकर्ता से प्राप्त करता है और उन्हें दो स्ट्रकचर d1 तथा d2 में रखता है। फिर यह फंक्शन add_distance( ) को कॉल करता है जो d1 तथा d2 को स्ट्रकचर के रुप में स्वीकार करता है तथा उन्हें जोड़ कर तीसरे स्ट्रकचर d3 के रुप में main फंक्शन को लौटा देता है।

3.29.9 स्ट्रकचर को रेफरेंस के रुप में एक फंक्शन को भेजना (Passing Structure to Function as reference)

जिस प्रकार सामान्य डाटा टाइप को एक फंक्शन को रेफरेंस द्वारा भेजा जा सकता है। उसी प्रकार एक स्ट्रकचर एक फंक्शन को रेफरेंस द्वारा भेजा जा सकता है। पिदले खण्ड में दिये गए प्रोग्राम में हमने फंक्शन add_distance को स्ट्रकचर d1 तथा d2 वेल्यु /आरग्युमेंट के रुप में भेजा था। इस खण्ड में, स्ट्रकचर d1 तथा d2 को फंक्शन add_distance को रेफरेंस के रुप में भेजेंगे ।

उदाहरण (Example) :-

```
#include <iostream>
# include <conio.h>
# include <stdio.h>
using namespace std;
structure distance
{
int meter;
float cm;
};
distance add_distance( distance& d1, distance& d2 );
void display( distance d )
void main( )
{
distance d1,d2,d3;
cout << "Enter meter:";
cin >> d1.meter;
cout << "Enter cm:";
cin >> d1.cm;
cout << "\n Enter meter:";
cin >> d2.meter;
cout << "Enter cm:";
cin >> d2.cm;
d3 = add_distance(d1, d2);
cout << endl;
display(d1);
cout << " + ";
display(d2);
cout << " = ";
display(d3);
cout << endl;
return 0;
getch ( );
}
void display( distance dd)
```

```
{
cout << dd.meter << " m " << dd.cm << " cm";
}
void add_distance( distance& dd1 , distance& dd2)
{
dd3.cm = dd1.cm + dd2.cm;
dd3.meter=0;
if ( dd3.cm >= 10.0)
        {
        dd3.cm -= 10.0 ;
        dd3.meter ++ ;
        }
dd3.meter += dd1.meter + d d2.meter
}
```

Output :-
Enter meter:8
enter cm:2.5

Enter meter:9
Enter cm:4.5

8 m 2.5 cm + 9 m 4.5 cm = 17 m 7 cm

अभ्यास प्रश्न

वस्तुनिष्ठ प्रश्न

निम्नलिखित प्रश्नो में सही विकल्प का चुनाव कीजिए

1. रेण्डम फंक्शन का प्रयोग करते समय निम्न हैडर फाइल को प्रोग्राम में जोड़ेगें।
   (अ) स्ट्रिंग डाट एच (ब) एस टी डीलिब डाट एच
   (स) सी टाइप डाट एच (द) उपरोक्त में से कोई नहीं

2. निम्न में से मेथ डाट एच हैडर फाइल का फंक्शन है।
   (अ) इज लोवर (ब) लाग
   (स) रेण्डमाइज (द) एग्जिट
3. एरे के अन्दर सारे अवयव होते हैं।
   (अ) एक ही तरह के (ब) अलग–अलग तरह के
   (स) मिक्स तरह के (द) कोई भी नहीं
4. 20 विद्यार्थियों की आयु को एक विमीय एरे में स्टोर करने पर उसके एरे के अवयव के नम्बर होंगे।
   (अ) 0 से 19 (ब) 0 से 20
   (स) 1 से 21 (द) कोई भी नहीं
5. एक स्ट्रिंग वेरियेबल char name [20]; में वेरियेबल के अंदर अक्षर आ सकते है।
   (अ) 20 (ब) 19
   (स) 21 (द) 22
6. एस टी आर सी पी वाई (strcpy) फंक्शन को प्रयोग करते समय प्रोग्राम के अंदर हैडर फाइल जोड़ेगें।
   (अ) मैथ डाट एच (math.h) (ब) स्ट्रिंग डाट एच (string.h)
   (स) सी टाइप डाट एच (ctype.h) (द) उपरोक्त में से कोई नहीं
7. किसी भी अक्षर को बडे अक्षरों में परिवर्तित करने का फंक्शन है।
   (अ) टू लोवर (tolower) (ब) टू अपर (toupper)
   (स) इज लोवर (islower) (द) इज अपर (isupper)

लघुत्तरात्मक प्रश्न (Short Answer Type Questions)

1. किसी भी स्ट्रिंग को एरे में स्टोर करने पर उसका अंतिम करैक्टर क्या आता है?
2. एरे कितने प्रकार के होते हैं ?
3. दो स्ट्रिंग की तुलना करने वाले फंक्शन का नाम बताइये ?
4. किस प्रकार के एरे में लाइन व कॉलम का प्रयोग होता हैं ?
5. टाइपडैफ (Typedef) क्या करता है ?
6. स्ट्रकचर को डिफाइन व डिकलेयर करने के लिए सिंटेक्स (Syntax) लिखिए।
7. स्ट्रकचर को इनीशीयलाइज करने के लिए सिंटेक्स (Syntax) लिखिए।

निबन्धात्मक प्रश्न (Essay Type Questions)

1. C++ भाषा में एक प्रोग्राम लिखिये जो यूजर द्वारा दिये गये संख्याओं का औसत मान निकाले।

2. एक विमीय एरे और द्विविमीय एरे में अन्तर लिखिये ?
3. एक प्रोग्राम लिखिये जो दो मैट्रिक्स का जोड़ निकाले।
4. C++ भाषा में एक प्रोग्राम लिखिये जो एक स्ट्रिंग को पढ़े व स्ट्रिंग के अन्दर जितने भी स्वर और व्यंजन है। उनकी गणना बताये।
5. C++ भाषा में एक प्रोग्राम लिखिये जो कि एक मैट्रिक्स के अंदर मुख्य एवं उप विकर्णों के अवयव का जोड़ निकाले।
6. C++ भाषा में स्ट्रकचर क्या होते है। स्ट्रकचर को किस प्रकार एक फंक्शन को भेजा जा सकता है? प्रत्येक का एक उदाहरण देकर समझाइए।
7. C++ भाषा में एक फंक्शन को किस प्रकार रिर्टन करता है ?एक उदाहरण देकर समझाइए।

उत्तरमाला

1. (अ) 2. (अ) 3. (अ) 4. (अ), 5. (ब)
6. (ब), 7. (ब)