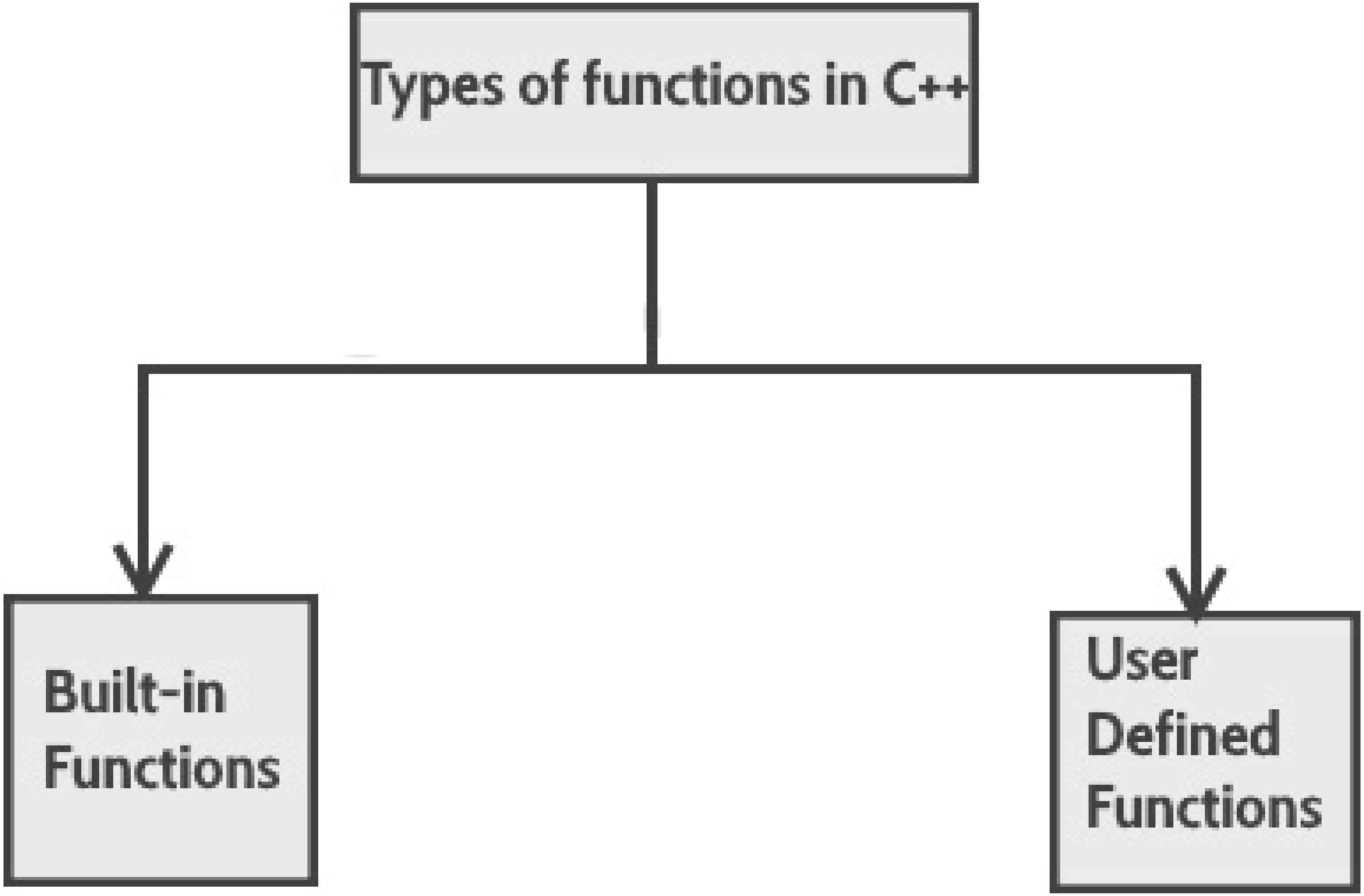
Functions

Functions

- A group of statements that perform a specified operation is called a function.
- In a function we group several program statements into a unit and give that unit a name that is called function name.
- Function can be invoked any where in the program.
- Program statements that appear in the program more than once are suitable for creating a function.
- Function code is stored in only one place in the memory.
- Another reason for creating functions is that a complex or bigger program code is divided into different functions due to which it becomes easy to manage the program.


Types of functions in C++
Built-in Functions
User Defined Functions

Built in functions

```
pow(2,5);
Sqrt(4);
```

User defined Functions

- There are 3 things important related to a function.

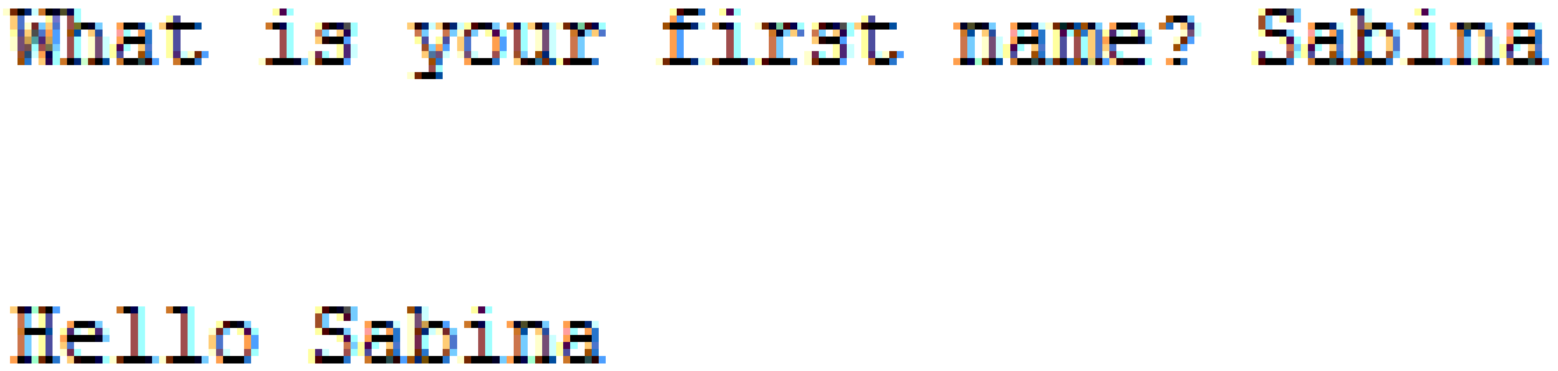
i) **Function Declaration**

ii) **Function Calling**

iii) **Function Definition**

```
#include <iostream>
using namespace std;
void PrintMessage();
//declaration
// Following is the function definition
void PrintMessage()
{
cout << "********************\n";
cout << " Welcome to my\n";
cout << " wonderful program!\n";
cout << "********************\n";
}
int main()
{
char name[10];
cout << "What is your first name? ";
cin >> name;
cout << "\n\nHello " << name << endl << endl;
PrintMessage();
}
```

OUTPUT



```
What is your first name? Sabina


Hello Sabina
```

```
#include <iostream>
using namespace std;
void sum(); //declaration
void sum()//definition
{
  int a=10;
  int b=20;
  int c;
  c=a+b;
  cout<<"Result is "<<c;
}
int main()
{
sum(); //Calling the function
return 0;
}
```

OUTPUT

```
Result is 30
```

```
#include<iostream.h>
#include<conio.h>
void line(void); //Function Declaration
void line(void) //Function Declarator
{       for(int a=1;a<=20;a++)
             cout<<"*";
        cout<<endl;
}
void main(void)
{       clrscr();
        line(); //Function Calling
        cout<<"Hello"<<endl;
        line();
        cout<<"We are studying functions"<<endl;
        line();
        getch();
}
```

Output

```
********************
Hello
********************
We are studying functions
********************
```

Eliminating the Declaration

```
#include<iostream.h>
#include<conio.h>
void line(void) //Function Definition without Declaration
    {
        for(int a=1;a<=20;a++)
            cout<<"*";
        cout<<endl;
    }
void main(void)
    { clrscr();
    line();
    cout<<"Hello"<<endl;
    line();
    cout<<"We are studying functions"<<endl;
    line();
    getch();
}
```

Passing by Value

- An argument is a piece of data, i.e. a value passes from program to function
- These passed values etc can be used by the function according to the requirements.
- There are two ways in Passing by Value, through which arguments can be passed to the functions, i.e.
    1. **Passing Constants to functions**
    2. **Passing Variables to functions**

Passing Constants to functions

- As the name represents, In passing constants to functions, a character, integer or float constant is actually passed as argument to the function, i.e.
  - **line('*');**
  - **square(5);**

```
#include<iostream.h>
#include<conio.h>
void line(char); //Function Declaration
void line(char ch) //Function Declarator
{       for(int a=1;a<=20;a++)
             cout<<ch;
        cout<<endl;
}
void main(void)
{       clrscr();
        line('*'); //Function Calling
        cout<<"Hello"<<endl;
        line('-');
        cout<<"We are studying functions"<<endl;
        line('*');
        getch();
}
```

Output

```
********************
Hello
--------------------
We are studying functions
********************
```

```
#include <iostream>
using namespace std;
int sum(int num1, int num2)
{
    int num3 = num1+num2;
    cout<<num3;
}
int main()
{
    sum(1,99);
    return 0;
}
```

Practice

- Write a program to check whether the entered number is even or not.
- Write a program to check the grade of students.
- Write a program to calculate the smallest number. Between (1,10)(31,10)(11,8)

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

void line(char='*', int=20);

void main(void)
{ clrscr();
  line();
  line('=');
  line('-',10);
  getch();
}

void line(char ch, int n)
{ for(int a=1;a<=n;a++)
   cout<<ch;
  cout<<endl;
}
```

```
#include <iostream>
using namespace std;
int main() {
int i, number=0, factorial=1;
while(number<1 || number>10)
{
    cout << "Enter integer number (1-10) = ";
    cin >> number;
}
for(i=1; i<=number; i++)
 { factorial = factorial*i; }
cout << "Factorial = " << factorial << endl;
return 0; }
```

Find Smallest number

```
#include<iostream>
using namespace std;
int compare(int a, int b) {
    return (a < b) ? a : b; }
int main()
 {cout << "\nSmallest Number :" << compare(1, 10);
 cout << "\nSmallest Number :" << compare(31, 10);
 cout << "\nSmallest Number :" << compare(11, 8);
  return 0;
}
```

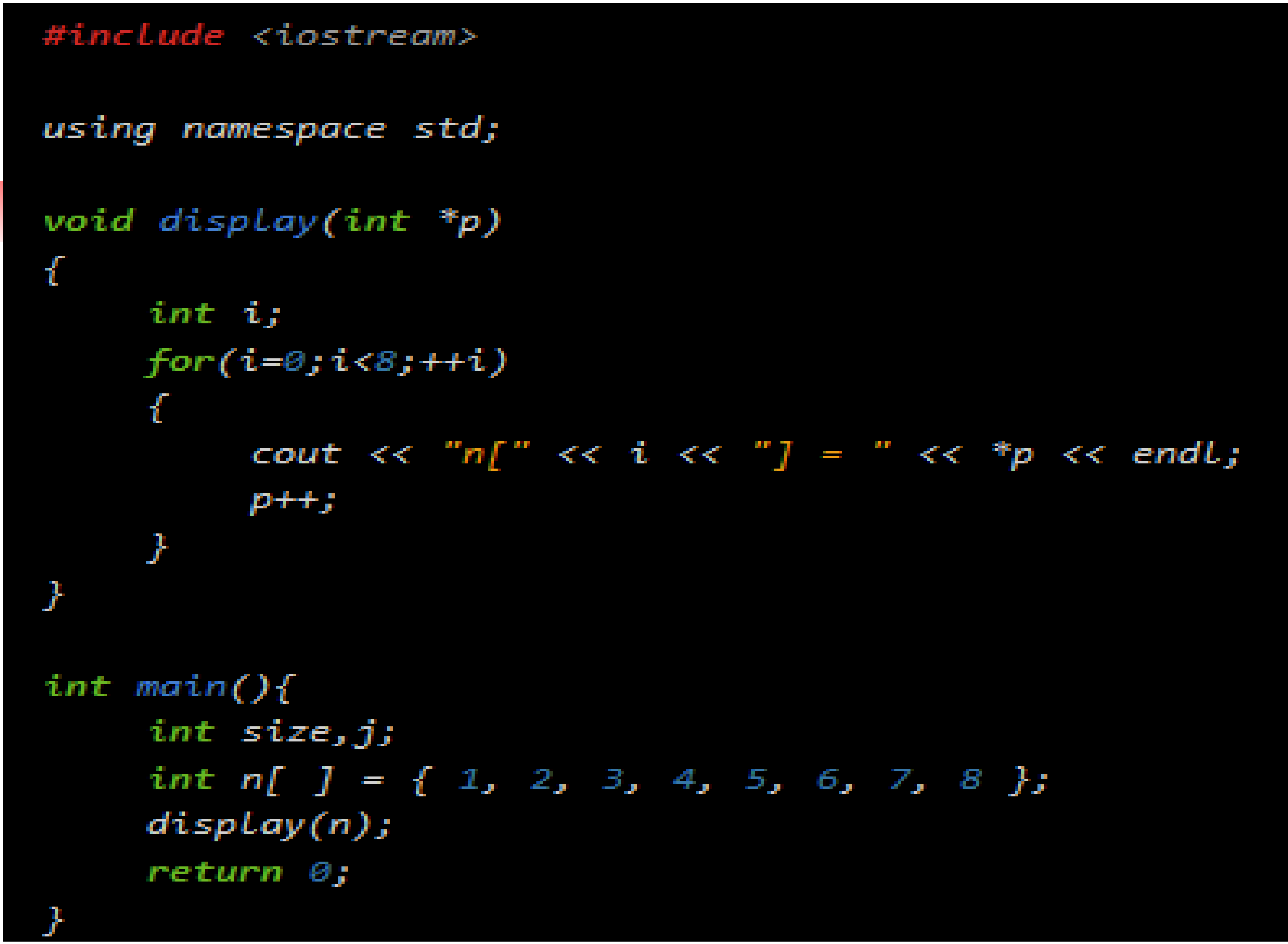
Passing Variables to functions

```
#include<iostream.h>
#include<conio.h>
void chline(char, int);
void chline(char ch, int n)
{
        for(int a=1;a<=n;a++)
                cout<<ch;
        cout<<endl;
}
void main(void)
{        clrscr();
        char ch;
         int n;
        cout<<"Enter a character ";
        cin>>ch;
        cout<<"Enter a value ";
        cin>>n;
        chline(ch, n);  //Character and Integer variables passed
        cout<<"Hello"<<endl;
        chline(ch, n);
        cout<<"We are studying functions"<<endl;
        chline(ch, n);
        getch();
}
```

Output

```
Enter a character +
Enter a value 10
++++++++++
Hello
++++++++++
We are studying
functions
++++++++++
```

- Arrays an functions



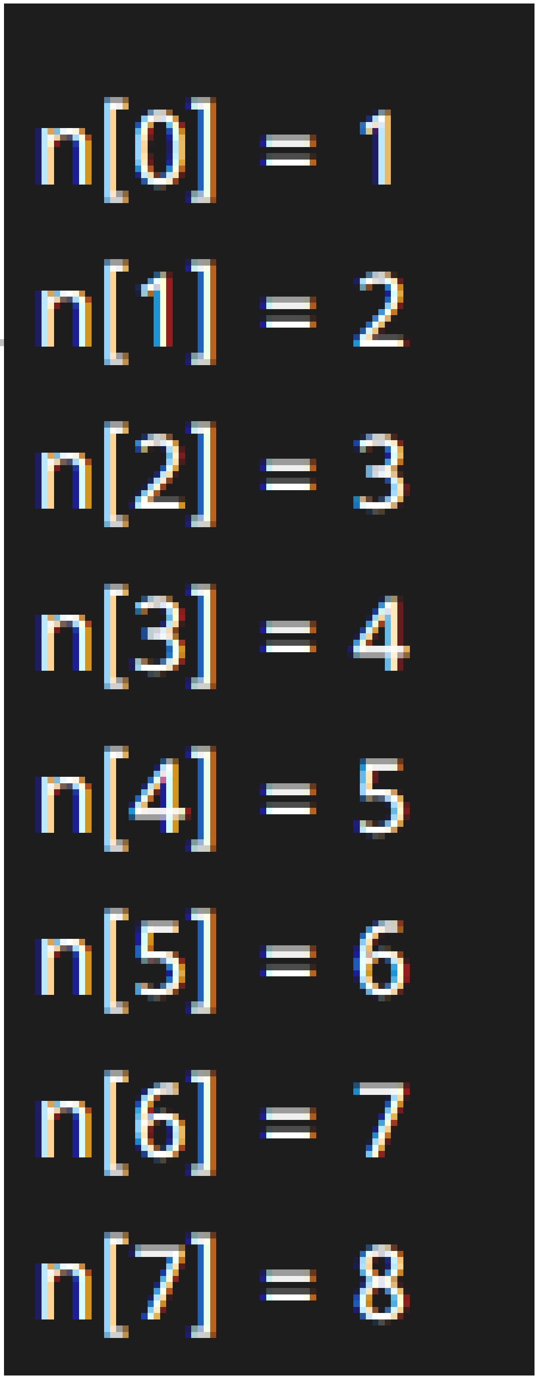
```cpp
#include <iostream>

using namespace std;

void display(int *p)
{
        int i;
        for(i=0;i<8;++i)
        {
                cout << "n[" << i << "] = " << *p << endl;
                p++;
        }
}

int main(){
        int size,j;
        int n[ ] = { 1, 2, 3, 4, 5, 6, 7, 8 };
        display(n);
        return 0;
}
```

OUTPUT



```
n[0] = 1
n[1] = 2
n[2] = 3
n[3] = 4
n[4] = 5
n[5] = 6
n[6] = 7
n[7] = 8
```

Functions with Structures

```
#include <iostream>
 using namespace std;
struct Student{
char stuName[30];
int stuRollNo;
int stuAge; };
void printStudentInfo(Student);
int main(){
Student s;
cout<<"Enter Student Name: "; cin.getline(s.stuName, 30);
cout<<"Enter Student Roll No: ";
 cin>>s.stuRollNo;
 cout<<"Enter Student Age: "; cin>>s.stuAge;
printStudentInfo(s); return 0; }
void printStudentInfo(Student s)
{ cout<<"Student Record:"<<endl;
cout<<"Name: "<<s.stuName<<endl;
cout<<"Roll No: "<<s.stuRollNo<<endl;
cout<<"Age: "<<s.stuAge; }
```

Practice

(3) Write a function reverse(int n) which reverses the digits in its paramter and returns the result. For example, if n is 927, it would return 729. Apply the function in a program that asks the user to enter 10 numbers and reverses them.

```
#include<iostream.h>
struct Distance
{     int feet;
      float inches;
};
void showDistance(Distance);
void main(void)
{     Distance d1, d2;
      cout<<"Enter feet for 1st Distance ";
      cin>>d1.feet;
      cout<<"Enter inches for 1st Distance ";
      cin>>d1.inches;
      cout<<"Enter feet for 2nd Distance ";
      cin>>d2.feet;
      cout<<"Enter inches for 2nd Distance ";
      cin>>d2.inches;
      cout<<"\nFirst Distance is ";
      showDistance(d1);
      cout<<"\nSecond Distance is ";
      showDistance(d2);
      }
```

```
void showDistance(Distance dd)
{   cout<<dd.feet<<"\'-"<<dd.inches<<"\"";
```

OUTPUT

```
Output
Enter feet for 1st Distance 5
Enter inches for 1st Distance 6.5
Enter feet for 2nd Distance 7
Enter inches for 2nd Distance 8.5
First Distance is 5'-6.5"
Second Distance is 7'-8.5"
```

Returning Structure Variables

```
#include<iostream.h>
#include<conio.h>
struct Distance
{ int feet;
float inches;
};
Distance sumDistance(Distance, Distance);
void showDistance(Distance);
void main(void)
{ clrscr();
    Distance d1, d2, d3;
    cout<<"Enter feet for 1st Distance ";
    cin>>d1.feet;
    cout<<"Enter inches for 1st Distance
   ";
    cin>>d1.inches;
    cout<<"Enter feet for 2nd Distance ";
    cin>>d2.feet;
    cout<<"Enter inches for 2nd
   Distance";
    cin>>d2.inches;
d3 = sumDistance(d1, d2);
cout<<"\nFirst Distance is ";
showDistance(d1);
cout<<"\nSecond Distance is ;
showDistance(d2);
cout<<"\nSum of Distance is”;
showDistance(d3);
getch();
}
```

Returning Structure Variables

```
Distance sumDistance(Distance dd1, Distance dd2)
{    Distance dd3;
     dd3.inches = dd1.inches + dd2.inches;
     dd3.feet = 0;
          if(dd3.inches >= 12.0)
          {
               dd3.inches -=12.0;
               dd3.feet++;
          }
     dd3.feet += dd1.feet + dd2.feet;
     return dd3;
}
void showDistance(Distance dd)
{
     cout<<dd.feet<<"'-"<<dd.inches<<"\"";
     cout<<endl;
}
```

Output

```
Enter feet for 1st Distance 5
Enter inches for 1st Distance 6.5
Enter feet for 2nd Distance 8
Enter inches for 2nd Distance 9.5
First Distance is 5'-6.5"
Second Distance is 8'-9.5"
Sum of Distance is 14'-4"
```

Passing by Reference

In passing arguments by reference, instead of passing a value to the function, its reference that is the address of that variable is passed.

Passing by reference has two main advantages, i.e.

1. Function can access the actual variables of the calling function.
2. Provides a mechanism for returning more than one value from the called function to its calling function.

Overloaded Functions
Or
Function Overloading

- Overloaded function or Function overloading means that more than one function with the same name exists in the program but differing in the number of arguments.
- When the function will be called, then number of arguments will decide that which function will be actually called, i.e.

Function Overloading Cont…

```
void line();
void line(int);
void line(char);
void line(int, char);
void line(char, int);
```

- We can see the above mentioned declarations that all five functions have the same name, i.e. **line**, but every functions prototype is different from one another.
- Similarly, when we'll call the function line than its number of arguments will decide, which function to execute.

Function Overloading Cont…

- Function definition doesn’t need to be in sequence the way functions are declared,
- but only requirement is that the number of function definitions should be equal to the number of function declarations.
- In overloaded functions, the compiler can distinguish even if we provide different types of arguments in the functions.

Function Overloading

```
#include<iostream.h>
#include<conio.h>
void line(void);
void line(int);
void line(char);
void line(int, char);
void line(char, int);

void main(void)
{     clrscr();
      line(10);
      line();
      line('=',15);
      line('*');
      line(20,'-');
      getch();
}
void line(void)
{ for(int a=1;a<=10;a++)
cout<<"*";
cout<<endl;
}
```

```
void line(int n)
{ for(int a=1;a<=n;a++)
      cout<<"*";
      cout<<endl;
}
void line(char c)
{ for(int a=1;a<=10;a++)
      cout<<c;
      cout<<endl;
}
void line(int n, char c)
{ for(int a=1;a<=n;a++)
      cout<<c;
      cout<<endl;
}
void line(char c, int n)
{ for(int a=1;a<=n;a++)
      cout<<c;
      cout<<endl;
}
```

Default Arguments in Function Declaration and Calling

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

void line(char='*', int=20);

void main(void)
{
        clrscr();
        line();
        line('=');
        line('-',10);
        getch();
}
void line(char ch, int n)
{
        for(int a=1;a<=n;a++)
        cout<<ch;
        cout<<endl;
}
```

Inline Functions

```
#include<iostream.h>
#include<conio.h>
inline float p2k(float pounds) //inline function
{ return 0.453592 * pounds;
}
void main(void)
{
         clrscr();
        int pounds;
        cout<<"Enter weight in pounds ";
        cin>>pounds;
        cout<<pounds<<" Pounds = "<<p2k(pounds);
        cout<<" Kilograms";
        getch();
}
```

Output

```
Enter weight in pounds 180
180 Pounds = 81.646561 Kilograms
```

Storage Classes of Variables

- There are three storage classes of variables
  1. **Automatic Variables**
  2. **External Variables**
  3. **Static Variables**

Static Variables

```
#include<iostream.h>
#include<conio.h>
float getAverage(float);
void main(void)
{ clrscr();
float data=1, average;
while(data!=0)
{ cout<<"Enter a number ";
cin>>data;
average = getAverage(data);
cout<<"New Average is "<<average<<endl;
}
getch();
}
float getAverage(float newdata)
{ static float total = 0;
static int count = 0;
count++;
total += newdata;
return total / count;
}
```

Output

```
Enter a number 5
New Average is 5
Enter a number 15
New Average is 10
Enter a number 25
New Average is 15
Enter a number 35
New Average is 20
Enter a number 10
New Average is 18
Enter a number 20
New Average is 18.333334
Enter a number 30
New Average is 20
Enter a number 0
New Average is 17.5
```

Practice tasks

1. Write a value returning function that receives three integers and returns the largest of
the three. Assume the integers are not equal to one another.
2. Write a value returning function that receives two floating point numbers and returns
true if the first formal parameter is greater than the second.
3. Write a value returning function that receives a character and returns true if the character is a vowel and false otherwise. For this example, vowels include the characters 'a', 'e', 'i', 'o', and 'u'.