**Labsheet-1: Function**

**Objectives**

* Recognize the functions
* Explain the inline functions
* Discuss the default arguments
* Explore function overloading
* Explain structures and its use

**Introduction**

Functions are divisions or parts of program which when called have the control transferred to the first statement in the function body. Other statements in the function body are then executed and control returns to the main function (calling function) when the closing braces are encountered. C++ functions are similar to C-functions but they also have quite a few additional features.

1. Inline Function
2. Default Arguments
3. Constant Arguments
4. Function Overloading
5. Call by Reference and Return by Reference

**1. Inline Function**

It is function that is expanded in line when it is invoked with the corresponding function code. **Inline** keyword is used in the header line of function to make the outside function inline. The general syntax of writing inline is as follows:

**inline** return\_type function-name(parameters){

// Function body

}

**Objective:** To save memory (which becomes appreciable when a function is likely to be called many times). These effective enough as long as functions are smaller in size. It reduces function-call overhead, reduces extra space needed for calling function, saves registers etc.

**Problems:**

1. **Write a program using inline function to calculate the square of a number.**

|  |
| --- |
| *// Using inline function to calculate the square of a number.*  *#include<iostream>*  *#include<conio.h>*  *using namespace std;*  *inline int square(int num){*  *return num \* num;*  *}*  *int main()*  *{*  *int number;*  *cout << "Please, Enter a number to find its square: " ;*  *cin >> number;*  *cout << "Square of " << number*  *<< " = " << square(number) ;*  *getch();*  *return 0;*  *}* |

**Possible Output:**

Please, Enter a number to find its square: **5**

Square of 5 = 25

------------------------------------------------------------------

Please, Enter a number to find its square: **-5**

Square of -5 = 25

1. **Write a program using inline function to calculate the multiplication and division of two users input number.**

|  |
| --- |
| #include<iostream>  #include<conio.h>  using namespace std;  inline int multiply(int m, int n){  return m \* n;  }  inline float divide(int m, int n){  return (m \* 1.0) / n;  }  int main(){  int num1, num2 ;  cout << "Please, Enter two number: "<< endl;  cout << "Num1: ";  cin >> num1;  cout << "Num2: ";  cin >> num2;  cout << num1 << " \* " << num2  << " = " << multiply(num1, num2) << endl;  cout << num1 << " / " << num2  << " = " << divide(num1, num2) << endl;  getch();  return 0;  } |

**Possible Output:**

Please, Enter two number:

Num1: **10**

Num2: **13**

10 \* 13 = 130

10 / 13 = 0.769231

**2. Default Argument**

C++ allows programmers to call a function without specifying all its arguments. In such cases, the function assigns a default value to the parameter which does not have a matching arguments in the function call. Default values are specified when the function is declared.

Example:

**float** amount (float principle, int period, float rate=0.15)

Here, default value of rate is 0.15 unless not specified.

1. **Write a Program to calculate SI using default value r=1.5%. Ask the user for principal amount and time [I=PTR/100].**

|  |
| --- |
| #include<iostream>  #include<conio.h>  using namespace std;  inline float simpleInterest(int p, int t, float r = 1.5){  return (p\*t\*r)/100;  }  int main(){  int principal,time;  cout << "Enter Principal amount and Time period:: ";  cin >> principal >> time;  cout << "Simple Interest:: "<< simpleInterest(principal,time) << endl;  getch();  return 0;  } |

**Possible Output:**

Enter Principal amount and Time period:: **1111**

**11**

Simple Interest:: 183.315

1. **Write and test the following ComputeSphere() function that returns the volume v and**

**Surface area s of a sphere with the given radius.**

**void computeSphere(float &v, float &s, float r)**

|  |
| --- |
| #include<iostream>  #include<conio.h>  using namespace std;  void computeSphere(float &v, float &s, float r){  v = (4/3) \* 3.14 \* r \* r \* r;  s = 4 \* 3.14 \* r \* r \* r;  }  int main(){  float radius, volume, area;  cout << "Enter Radius:: ";  cin >> radius;  computeSphere(volume, area, radius);  cout << "Volume of sphere:: "<< volume  <<"\nSurface area of sphere:: " << area << endl;  getch();  return 0;  } |

**Output:**

Enter Radius:: 5

Volume of sphere:: 392.5

Surface area of sphere:: 1570

**3. Constant Arguments**

In C++, an arguments to a function can be declared as constants as shown below:

int strlen(const char\* p);

int length(const string &s);

**4. Function Overloading**

Function overloading means using the same function name to create functions that perform different tasks. Function overloading allows us to design a family of functions with the same function name but having varying number of (and type of) arguments.

**Problems:**

1. **Write a program to find the volume of 3 objects: namely cube, cylinder and rectangular box using the same function name, volume().**

|  |
| --- |
| #include<iostream>  #include<conio.h>  using namespace std;  inline int volume(int l){  return l \* l \* l;  }  inline float volume(int h, float r){  return 3.14 \* r \* r \* h;  }  inline int volume(int l, int b , int h){  return l \* b \* h;  }  int main(){  float radius;  int l\_cube, h\_cyl, l\_rec, b\_rec, h\_rec;  cout << "Enter side of a Cube:: ";  cin >> l\_cube;  cout << "Volume of cube is " << volume(l\_cube) << endl;  cout << "Enter Radius and Height for Cylinder:: ";  cin >> radius >> h\_cyl;  cout << "Volume of cylinder is " << volume(h\_cyl, radius) << endl;  cout << "Enter length, breadth and height for Rectangle:: ";  cin >> l\_rec >> b\_rec >> h\_rec;  cout << "Volume Rectangle" << volume(l\_rec, b\_rec, h\_rec) << endl;  getch();  return 0;  } |

**Output:**

Enter side of a cube:: **10**

Volume of a cube is 1000

Enter radius and height for cylinder:: **2 5**

Volume of cylinder is 62.8

Enter length, breadth and height for rectangle:: **5 10 15**

Volume of Rectangle is 750.

1. **Similarly write a program to find the Area of cube, cylinder and rectangle using the concept of function overloading.**

|  |
| --- |
| #include<iostream>  #include<conio.h>  using namespace std;  inline float area(float l){  return 6 \* l \* l;  }  inline float area(float h, float r){  return 2 \* 3.14 \* r \* (r+h);  }  inline int area(int l, int b){  return l \* b;  }  int main(){  float radius, l\_cube, hc;  int l\_rec, b\_rec;  cout << "Enter side of a cube:: ";  cin >> l\_cube;  cout << "Area Cube: " << area(l\_cube) << endl;  cout << "Radius, Height For Cylinder: ";  cin >> radius >> hc;  cout << "Area Cylinder = " << area(hc, radius) << endl;  cout << "Length, breadth for rectangle:: ";  cin >> l\_rec >> b\_rec;  cout << "Area Rectangle = " << area(l\_rec, b\_rec) << endl;  getch();  return 0;  } |

**Possible Output:**

Enter radius of a cube:: **5**

Area of cube is 150

Enter Radius and height for cylinder:: **5 4**

Area of cylinder is 282.6

Enter length, breadth for rectangle:: **9 8**

Area of rectangle is 72

**STRUCTURES**

Structure is a collection of simple variables and the variables can be of different data types. In C, structure was confined to holding data only but in C++, structures can hold both data and functions. Structures can be nested i.e. structure can be used within another structure too.

**Problems:**

1. **A phone no. such as (212) 767-8900 can be thought as having three parts:**

**The area code (212), the exchange (767), and the number (8900).**

**Write a Program that uses a structure to store the three parts of a phone number separately, call the structure phone. Create two structure variable of type phone. Initialize one, and have the user input a number for the other one. Then display both number.**

|  |
| --- |
| #include<iostream #include<conio.h> using namespace std;  struct Phone{  int area\_code;  int exchange;  int number; };  int main(){  Phone me, y;  me.area\_code = 212;  me.exchange = 767;  me.number = 8900;  cout << "Enter your area code, exchange and number:: ";  cin >> y.area\_code >> y.exchange >> y.number;  cout << "My number is " << "(" << me.area\_code  <<")" << me.exchange << "-" << me.number << endl;  cout << "Your number is " << "(" << y.area\_code  << ")" << y.exchange << "-" << y.number << endl;  getch();  return 0;  } |

**Output:**

Enter your area code, exchange and number:: **145 555 1212**

My number is (212)767-8900

Your number is (145)555-1212

1. **Create a structure called volume that uses three variables of type distance to model the volume of a room. Initialize a variable of type volume of room. Initialize a variable of type volume to specify dimensions, then calculate the volume it represents and printout the results. To calculate the volume convert each dimension from a distance variable to a variable of type float representing feet and fraction of a foot and then multiply the resulting three number.**

|  |
| --- |
| #include<iostream>  #include<conio.h>  using namespace std;  struct Dim{  int feet;  float inch;  };  struct Volume{  Dim length;  Dim breadth;  Dim height;  };  int main(){  Volume V;  V.length.feet = 12; V.length.inch = 5;  V.breadth.feet = 7; V.breadth.inch = 6;  V.height.feet = 8; V.height.inch = 8;  float l = V.length.feet + V.length.inch/12;  float b = V.breadth.feet + V.breadth.inch/12;  float h = V.height.feet + V.height.inch/12;  cout << "Volume(sq. feet) = " << l \* b \* h << endl;  getch();  return 0;  } |

**Possible Output:**

Volume(sq. Feet) = 807.083

1. **Write a Program using structure to create a database for a hospital giving details about the patient admitted.**

|  |
| --- |
| #include<iostream>  #include<conio.h>  using namespace std;  struct Detail{  char name[100];  int age, bed\_no  int dd, mm, yy;  char nature[100], gender[10];  int ward\_no, dd1, mm1, yy1;  };  void getDetail(Detail &D)  {  cout << "Enter patient name:: ";  cin >> D.name;  cout << "Enter age:: ";  cin >> D.age;  cout << "Enter bed number:: ";  cin >> D.bed\_no;  cout << "Enter Nature of Illness:: ";  cin >> D.nature;  cout << "Date Of Birth(DD-MM-YY): ";  cin >> D.dd >> D.mm >> D.yy;  cout << "Enter gender:: ";  cin >> D.gender;  cout << "Enter ward number:: ";  cin >> D.ward\_no;  cout << "Enter date of admission in dd mm yy format:: ";  cin >> D.dd1 >> D.mm1 >> D.yy1;  }  int main()  {  Detail D1;  getDetail(D1);  cout << "\*\*-----Patient Detail-----\*\*" <<endl;  cout << "Patient Name:: " << D1.name << endl;  cout << "Age:: " << D1.age << endl;  cout << "Bed No. " << D1.bed\_no << endl;  cout << "Nature of illness: " << D1.nature <<endl;  cout << "DOB:" << D1.dd << "/" << D1.mm << "/" << D1.yy <<endl;  cout << "Gender: " << D1.gender << endl;  cout << "Ward No: " << D1.ward\_no <<endl;  cout << "Admit" << D1.dd1<< "/" << D1.mm1 << "/" << D1.yy1 <<endl;  /getch();  return 0;  } |

**Output:**

Enter Patient Name: **Trump**

Enter age:: **76**

Enter bed number:: **12**

Enter Nature of Illness:: **Unknown**

Date Of Birth(DD-MM-YY): **11 09 2017**

Enter gender:: M**ale**

Enter ward number:: 4

Admitted On: **12 11 2017**

\*\*-----Patient Detail-----\*\*

Patient Name: Trump

Age: 76

Bed No: 12

Nature of illness: Unknown

Date of Birth: 11/09/2017

Gender: Male

Ward No. : 4

Admit: 12/11/2017

**Conclusion**

**Function** is a group of statements that together perform a task. C++ program has at least one function, which is **main()**, and all the most trivial programs can define additional functions. The C++ compiler need to know about a function before you call it, and you can let the compiler know about it in two ways.

1. By defining before you call it.  
2. By specifying function prototype before you call it.

Statement that declares a function’s name, return type, number and type of arguments is **Function** **Declaration**. In C++, it is possible to define several functions with the same name performing different actions but with different parameter, thanks to **Function** **Overloading**. A function definitions such that each call to function is, in effect, replaced by the statement that define that function is **Inline** **Function. Structure** is a collection of simple variables and the variables can be of different data types. The C++ keyword **return** is one of the several way to exit a function. When **return** **0** statement is used at the end of the **main(),** the value 0 indicate that the program has run successfully.

***The End.***