ARIGNAR ANNA GOVT ARTS AND SCIENCE COLLEGE

KARAIKAL – 609605

Introduction to C++ - Record

November - 2023

V-Semester

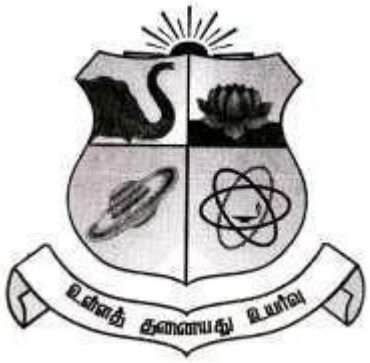
Reg. No. :_____

Name :_____

DEPARTMENT OF COMPUTER SCIENCE

AAGASC - KARAIKAL

ARIGNAR ANNA GOVERNMENT ARTS AND SCIENCE COLLEGE, KARAIKAL-609605



DEPARTMENT OF COMPUTER SCIENCE

Certified that this is the bonafide record of practical work done by Mr. / Miss …….. Reg. No. ..... of Third Year B.Sc. Computer Science during the Vth -Semester in the academic year 2023-24.

HEAD OF THE DEPARTMENT **STAFF INCHARGE**

Submitted for the University Examination held on ……..…..

EXTERNAL EXAMINER **INTERNAL EXAMINER**

CONTENTS

// Ex1. Program to Illustrate Class and object

```
#include<iostream.h>
#include<conio.h>
class person
{
 int a,b;
 public:
  void getdata(void);
  void display(void);
};
void person::getdata(void)
{
        cout<<"\n\tEnter A value :";
        cin>>a;
        cout<<"\n\tEnter B Value :";
        cin>>b;
}
void person::display(void)
{
  cout<<"\n\tAddition Result :"<<a+b;
}
void main()
{
  clrscr();
  person p;
```

```
 cout<<"\t\t Usage of Class And Object\n";
 cout<<"\t\t ************************";
 p.getdata();
 p.display();
 getch();
}
```

OUTPUT:

```
                    Usage of Class And Object
                    **********************
          Enter A Value  : 5
          Enter B Value   : 9
          Addition Result :14
```

// Ex2. Program To Illustrate Inline Functions

```
#include<iostream.h>
#include<conio.h>
inline float mul(float x, float y)
{
 return(x*y);
}
inline float div(float p,float q)
{
 return(p/q);
}
void main()
{
 float a,b;
 clrscr();
 cout<<"\t\t Inline Function";
 cout<<"Enter Values  for a & b:";
 cin>>a>>b;
 cout<<"\n\ta*b = "<<mul(a,b)<<"\n";
 cout<<"\n\ta/b = "<<div(a,b)<<"\n";
 getch();
}
```

OUTPUT:

Inline Function

Enter Values for a& b: 5 8

a*b = 40

a/b = 0.625

// Ex3. Program to Illustrate Static Variables

```
#include<iostream.h>

#include<conio.h>


int count(void)

{

 static int a=0;

 a=a+10;

 return(a);

}

void main()

{

 clrscr();

 cout<<"\n\t\t Static Variable\n";

 cout<<"\n\t\t ***************\n\t";

 cout <<count()<<"\n\t";

 cout <<count()<<"\n\t";

 cout <<count()<<"\n\t";

 getch();

}
```

OUTPUT:

Static Variable

****

10

20

30

// Ex4. Constructor

```
#include <iostream.h>
#include<string.h>
#include<conio.h>
 class Car { // The class
 public: // Access specifier
 char brand[30]; // Attribute
 char model[30]; // Attribute
 int year; // Attribute
 Car(char x[], char y[], int z) { // Constructor with parameters
 strcpy(brand,x);
 strcpy(model,y);
 year = z;
 }
};
void main() {
 // Create Car objects and call the constructor with different values
 clrscr();
 cout<<"\n\t\t Constructor\n";
 cout<<"\n\t\t **********\n";
 Car carObj1("BMW", "X5", 1999);
 Car carObj2("Ford", "Mustang", 1969);
 // Print values
 cout << carObj1.brand << " " << carObj1.model << " " << carObj1.year << "\n";
 cout << carObj2.brand << " " << carObj2.model << " " << carObj2.year << "\n";
```

```
 getch();
}
```

OUTPUT:

```
                Constructor
                **********
        BMW X5 1999
        Ford Mustang 1969
```

/* Ex5. friend function */

```
#include <iostream.h>
#include<conio.h>
class Distance {
   private:
          int meter;
          // friend function
          friend int addFive(Distance);
   public:
          Distance() : meter(0) {}
};
// friend function definition
int addFive(Distance d)
{
   //accessing private members from the friend function
   d.meter += 5;
   return d.meter;
}
void main()
{
   clrscr();
   Distance D;
   cout<<"\n\t\t Friend Function";
   cout<<"\n\t\t ***************\n";
   cout<<"\n\t\tDistance: " << addFive(D);
```

```
    getch();
}
```

OUTPUT:

```
                Friend Function
                **************
            Distance : 5
```

// Ex6. Operator Overloading

```
#include <iostream.h>
#include<conio.h>
class Count {
  private:
   int value;
  public:
   // Constructor to initialize count to 5
   Count() : value(9) {}
   // Overload ++ when used as prefix
   void operator~() {
          ++value;
   }
   void display() {
          cout << "Count: " << value << endl;
   }
};
void main()
 {
   clrscr();
   cout<<"\n\t\t Operator Overloading\n\n\t";
   Count count1;
   // Call the "void operator ++ ()" function
   ~count1;
```

```
    count1.display();
    getch();
}
```

OUTPUT:

```
        Operator Overloading
    Count : 10
```

/* Ex7. Program To illustrate Function Overloading */

```
#include<iostream.h>
#include<conio.h>
// function with 2 parameters
void display(int var1, double var2) {
 cout << "Integer number: " << var1;
 cout << " and double number: " << var2 << endl;
}
// function with double type single parameter
void display(double var) {
 cout << "Double number: " << var << endl;
}
// function with int type single parameter
void display(int var) {
 cout << "Integer number: " << var << endl;
}
void main() {
 int a = 5;
 double b = 5.5;
 clrscr();
 cout<<"\t\t  Function Overloading\n";
 cout<<"\t\t  ********************\n";
 // call function with int type parameter
 display(a);
```

```
 // call function with double type parameter
 display(b);
 // call function with 2 parameters
 display(a, b);
 getch();
}
```

OUTPUT:

```
                    Function Overloading
                    **********
Integer number : 5
Double number : 5.5
Integer number : 5 and double number: 5.5
```

/* EX8. Multilevel Inheritance */

```
#include<iostream.h>
#include<conio.h>
class sem1
{
private:
public:
sem1(void)
{
 cout<<"\n\tSem1 Papers";
 cout<<"\n\t Programming in C";
 cout<<"\n\t Digital";
}
};
class sem2 : public sem1
{
private:
public:
sem2(void)
{
 cout<<"\n\tSem2 Papers";
 cout<<"\n\t Python";
 cout<<"\n\t Data Structure";
}
};
```

```
class sem3: public sem2
{
private:
public:
sem3(void)
{
cout<<"\n\tSem3 Papers";
cout<<"\n\t Object oriented programming using Java";
cout<<"\n\t Operating System";
cout<<"\n\t Computer Networks";
cout<<"\n\t Software Engineering";
}
};
void main()
{
clrscr();
cout<<"\n\nSem1 Object :\t";
sem1 s1;
cout<<"\n\nSem2 Object :\t";
sem2 s2;
cout<<"\n\nSem3 Object :\t";
sem3 s3;
getch();
}
```

OUTPUT:

Sem1 Object :

Sem1 Papers

Programming in C

Digital

Sem2 Object :

Sem1 Papers

Programming in C

Digital

Sem2 Papers

Python

Data Structure

Sem3 Object :

Sem1 Papers

Programming in C

Digital

Sem2 Papers

Python

Data Structure

Sem3 Papers

Object oriented programming using Java

Operating System

Computer Networks

Software Engineering

/*Ex9. Hierarchical Inheritance */

```
#include<iostream.h>
#include<conio.h>
class dad
{
private:
public:
dad(void)
{
 cout<<"\n\tDad House ";
}
};
class son1 : public dad
{
private:
public:
son1(void)
{
 cout<<"\n\tSon1 House";
}
};
class son2 : public dad
{
private:
public:
```

```
son2(void)
{
cout<<"\n\tSon2 House";
}
};
class son3 : public dad
{
private:
public:
son3(void)
{
cout<<"\n\tSon3 House";
}
};
void main()
{
clrscr();
cout<<"\t\t Hierarchical Inheritance";
cout<<"\n\n Dad class Object :\t";
dad d;
cout<<"\n\nSon1 class Object :\t";
son1 s1;
cout<<"\n\nSon2 class Object :\t";
son2 s2;
cout<<"\n\nSon3 class Object :\t";
```

```
son3 s3;

getch();

}
```

OUTPUT:

Hierarchical Inheritance

Dad class Object :

Dad House

Son1 class Object :

Dad House

Son1 House

Son2 class Object :

Dad House

Son2 House

Son3 class Object :

Dad House

Son3 House

/* Ex10. Pointers to Object in C++ */

```
#include <iostream.h>
#include<conio.h>
class Time
{
  short int hh, mm, ss;
public:
  Time()
  {
    hh = mm = ss = 0;
  }
  void getdata(int i, int j, int k)
  {
    hh = i;
    mm = j;
    ss = k;
  }
  void prndata(void)
  {
    cout << "\nThe time is " << hh << ":" << mm << ":" << ss << ".\n";
  }
};
void main()
{
  Time T1, *tptr;
```

```
  clrscr();
  cout<<"\n\t\t Pointer to Object\n";
  cout<<"\n\t\t *****************\n\n";
  cout << "Initializing data members using the object with values 12, 22, and 11\n";
  T1.getdata(12, 22, 11);
  cout << "\nPrinting members using the object ";
  T1.prndata();
  tptr = &T1;
  cout << "Printing members using the object pointer ";
  tptr->prndata();
  cout << "\n\nInitializing data members using the object pointer, with values 15, 10, and 16\n";
  tptr->getdata(15, 10, 16);
  cout << "\nprinting members using the object ";
  T1.prndata();
  cout << "Printing members using the object pointer ";
  tptr->prndata();
  getch();
}
```

OUTPUT:

Pointer to Object

****

Initializing data members using the object with values 12, 22, and 11

Printing members using the object

The time is 12:22:11.

Printing members using the object pointer

The time is 12:22:11.

Initializing data members using the object pointer,with values 15, 10, and 16

Printing members using the object

The time is 15:10:16.

Printing members using the object pointer

The time is 15:10:16