 GALGOTIAS UNIVERSITY

**Plot No.2, Sector -17 A, Yamuna Expressway,**

**Greater Noida, Gautam Buddha Nagar, U.P., India**

**SCHOOL OF COMPUTING SCIENCE & ENGINEERING**

**“LAB PRACTICAL FILE”**

##### Course Name: Object Oriented Programming

**Course Code:  BCS01T1006**

**School: SCSE**

**Program: B. Tech**

**Year: 1st Semester: 2**

**Session: 2021-2022**

|  |  |
| --- | --- |
| **Submitted By:** | **Submitted To:** |
| **NEERAJ SINGH**  **21SCSE1011675**  **SEC-24,P2** | Miss Ragini Kumari |

|  |  |  |  |
| --- | --- | --- | --- |
| **S. NO.** | **Program name** | **date** | **sign** |
| 1 | Write a Simple C++ programs to implement various control structures.  a)Switch Statement b)if statement  c)For loop d)While loop | 4/4/2022 |  |
| 2 | Write a program explaining various access modifiers | 4/4/2022 |  |
| 3 | Write a program to explain the concept of function overloading | 4/4/2022 |  |
| 4 | Write program on Constructors & destructors. | 6/5/2022 |  |
| 5 | Write a Program to understand friend function & friend Class | 6/5/2022 |  |
| 6 | Write a program to explain the concept of Constructor overloading | 6/5/2022 |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |
| 10 |  |  |  |
| 11 |  |  |  |
| 12 |  |  |  |
| 13 |  |  |  |
| 14 |  |  |  |
| 15 |  |  |  |
| 16 |  |  |  |
| 17 |  |  |  |
| 18 |  |  |  |
| 19 |  |  |  |
| 20 |  |  |  |

|  |  |
| --- | --- |
| **Ex. No.: 1** | **Date: 04**/04/2022 |
| **implement various control structures** | |
| **Aim:**  **Write a Simple C++ programs to implement various control structures. a) Switch Statement b) if statement c)For loop d) While loop** | |
| **Program:**  #include <bits/stdc++.h>  using namespace std;  int main() {      //while      while(1){          cout<<"1.one"<<endl;          cout<<"2.two"<<endl;          cout<<"3.three"<<endl;          int n;          cin>>n;          //switch          switch (n)          {          case 1:          cout<<"one\n";              break;          case 2:          cout<<"two\n";              break;          case 4:          cout<<"three\n";              break;          default:              break;          }          //for          for(int i=0;i<1;i++){              if(n==1){                  cout<<"hello one "<<endl;              }              else if(n==2){                  cout<<"hello two"<<endl;              }              else if(n==3){                  cout<<"hello three"<<endl;              }              else{                  cout<<"hello world"<<endl;              }          }          char ch;          cout<<"do you want to continue(y/n):";          cin>>ch;          //if          if(ch=='n' || ch=='N'){              break;          }      }      return 0;  } | |
| **Output:**  1.one  2.two  3.three  1  one  hello one  do you want to continue(y/n):n | |
| **Result:** The above experiment successfully completed. | |

|  |  |
| --- | --- |
| **Ex. No.: 2** | **Date: 04**/04/2022 |
| **access modifiers** | |
| **Aim:**  Write a program explaining various access modifiers | |
| **Program**  #include <bits/stdc++.h>  using namespace std;  class Animal{      private:          int age;      protected:          int id;      public:          string name;          int getAge(){              return age;          }          void Setage(int a){              age=a;          }          int getId(){              return id;          }          void setId(int i){              id=i;;          }};  int main() {      Animal a1;      a1.name="neeraj";      a1.setId(1234);      a1.setId(4);      cout<<"name is :"<<a1.name<<endl;      cout<<"age is : "<<a1.getAge()<<endl;      cout<<"id is : "<<a1.getId()<<endl;      return 0;} | |
| **Output:**  name is :neeraj  age is : 0  id is : 4 | |
| **Result:** The above experiment successfully completed. | |

|  |  |
| --- | --- |
| **Ex. No.: 3** | **Date: 04**/04/2022 |
| **function overloading** | |
| **Aim:**  Write a program to explain the concept of function overloading | |
| **Program**  #include<bits/stdc++.h>  using namespace std;  class Circle{      public:      void area(int radius){          int areaa=3.14\*radius\*radius;          cout<<"the area of circle is: "<<areaa<<endl;      }  };  class Triangle{      public:      void area(int base,int height){        int areaa=0.5\*base\*height;        cout<<"area of trianle is :"<<areaa<<endl;      }    };  class Square{      public:      void area(int side){          int areaa=side\*side;          cout<<"area of square is :"<<areaa<<endl;      }  };  int main(){      Circle c;      Triangle t;      Square s;      c.area(4);      t.area(4,6);      s.area(7);      return 0;  } | |
| **Output:**  the area of circle is: 50  area of trianle is :12  area of square is :49 | |
| **Result:** The above experiment successfully completed. | |

|  |  |
| --- | --- |
| **Ex. No.: 4** | **Date: 06**/05/2022 |
| **Constructors & destructors** | |
| **Aim:**  Write program on Constructors & destructors. | |
| **Program**  #include<bits/stdc++.h>  using namespace std;  class Hero {      private:      int health;      public:      char \*name;      //constructor      Hero() {          cout << "neeru bhai call constructor" << endl;          name = new char[100];      }      //Destructor -> free memmory      ~Hero() {          cout << "neeru bhai call Destructor " << endl;      }  };  int main() {      Hero h;      return 0;  } | |
| **Output:**  neeru bhai call constructor  neeru bhai call Destructor | |
| **Result:** The above experiment successfully completed. | |

|  |  |
| --- | --- |
| **Ex. No.: 5** | **Date: 06**/05/2022 |
| friend function & friend Class | |
| **Aim:**  Write a Program to understand friend function & friend Class | |
| **Program**  // C++ program to demonstrate the working of friend function  #include <iostream>  using namespace std;  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;  }  int main() {  Distance D;  cout << "Distance: " << addFive(D);  return 0;  }  Part 2-friend class  // Add members of two different classes using friend functions  #include <iostream>  using namespace std;  // forward declaration  class ClassB;  class ClassA {    public:  // constructor to initialize numA to 12  ClassA() : numA(12) {}    private:  int numA;    // friend function declaration  friend int add(ClassA, ClassB);  };  class ClassB {  public:  // constructor to initialize numB to 1  ClassB() : numB(1) {}    private:  int numB;    // friend function declaration  friend int add(ClassA, ClassB);  };  // access members of both classes  int add(ClassA objectA, ClassB objectB) {  return (objectA.numA + objectB.numB);  } | |
| **Output:**  Output-1-friend function  Distance: 5  Output-2-friend class  Sum: 13 | |
| **Result:** The above experiment successfully completed. | |

|  |  |
| --- | --- |
| **Ex. No.: 4** | **Date: 06**/05/2022 |
| **Constructor overloading** | |
| **Aim:**  Write a program to explain the concept of Constructor overloading | |
| **Program**  #include <iostream>  using namespace std;  class construc{  public:  float area  // Constructor with no parameters  construct(){  area = 0;  }  // Constructor with two parameters  construct(int a, int b){  area = a \* b;  }    void disp(){  cout<< area<< endl;  }  };  int main(){  construct o;  construct o2( 10, 20);    o.disp();  o2.disp();  return 1;  } | |
| **Output:**  0  200 | |
| **Result:** The above experiment successfully completed. | |