Objectives:

* Learn how to create and use classes and objects in C++.
* Use constructors to set up objects when they are created.
* Understand destructors and how they clean up when objects are destroyed.
* Write simple programs using classes, constructors, and destructors.

Tools and Libraries Used :

* Programming Language: C++
* IDE: Visual Studio Code
* Libraries: #include<iostream>, #include<string>

Theory

**Classes and Objects in C++**

In C++, a **class** is a user-defined data type that acts as a blueprint for creating **objects**. A class can contain **data members** (variables) and **member functions** (methods) which define the behavior of the class.

An **object** is an instance of a class. When you create an object, you’re creating a variable that has its own copy of the class’s data and access to the class’s functions.

Syntax of a Class:

|  |
| --- |
| 1. class ClassName { 2. public:   1. // data members 2. // member functions 3. }; |

By default, members of a class are **private**, but using public:, we make them accessible from outside the class.

Example:

|  |
| --- |
| 1. #include <iostream> 2. using namespace std; 3.   4. class Car { 5. public:  6. string brand; 7. int year; 8.   1. void display() { 2. cout << "Brand: " << brand << ", Year: " << year << endl; 3. } 12. }; 13. 4. int main() { 5. Car myCar; // Creating an object of Car 6. myCar.brand = "Toyota"; 7. myCar.year = 2021; 8. myCar.display(); // Accessing member function 9. return 0; 10. } |

* Car is a class with two variables (brand and year) and a function display().
* myCar is an object created from the Car class.
* We assign values to the object and call its method.

**Constructors in C++**

A **constructor** is a special member function that is automatically called when an object is created. Its main job is to initialize the object.

**Rules and Features:**

* The constructor has the **same name** as the class.
* It has **no return type**, not even void.
* You can have **multiple constructors** (constructor overloading).

**Syntax:**

|  |
| --- |
| 1. class ClassName { 2. public:   1. ClassName(parameters) { 2. // constructor body 3. } 6. }; 7. |

**Example with Constructor:**

|  |
| --- |
| 1. #include <iostream> 2. using namespace std; 3.   4. class Car { 5. public:  6. string brand; 7. int year; 8.   1. // Constructor 2. Car(string b, int y) { 3. brand = b; 4. year = y; 13. } 14. 5. void display() { 6. cout << "Brand: " << brand << ", Year: " << year << endl; 7. } 18. }; 19. 8. int main() { 9. Car myCar("Honda", 2019); // Constructor is automatically called 10. myCar.display(); 11. return 0; 12. } |

**Explanation:**

* The constructor Car(string b, int y) sets the initial values of brand and year when the object is created.
* No need to assign values manually later in main().

**Destructors in C++**

A **destructor** is a special function that is automatically called **when an object goes out of scope** or is explicitly deleted. It is used to clean up resources such as memory, files, or database connections.

**Rules:**

* It has the **same name** as the class, but preceded by a tilde (~).
* It has **no parameters** and **no return type**.
* A class can have **only one destructor**.

**Syntax:**

|  |
| --- |
| 1. class ClassName { 2. public:   1. ~ClassName() { 2. // cleanup code 3. } 4. }; |

**Example with Destructor:**

|  |
| --- |
| 1. #include <iostream> 2. using namespace std; 3.   4. class Car { 5. public:  6. string brand; 7.   1. // Constructor 2. Car(string b) { 3. brand = b; 4. cout << brand << " is created." << endl; 12. } 13. 5. // Destructor 6. ~Car() { 7. cout << brand << " is destroyed." << endl; 8. } 18. }; 19. 9. int main() { 10. Car myCar("BMW"); 11. // Destructor will be called automatically when main ends 23. return 0;   24. }  25. |

**Explanation:**

* When myCar is created, the constructor runs.
* When main() ends and myCar goes out of scope, the destructor runs automatically.

Q1: Create a class Employee with data members employeeID, name, and salary. Add member functions to read and display these details. WAP to input and display any number of employees as desired by the user.

Code:

|  |
| --- |
| #include <iostream> #include<string> using namespace std; class Employee  { private: int EmployeeID; string name; float salary;  public:  void setData(int id, string n, float s) {  EmployeeID = id; name = n; salary = s;  }  void displayData() {  cout << "Employee ID: " << EmployeeID << endl; cout << "Name: " << name << endl; cout << "Salary: " << salary << endl;  }  }; int main(){  cout << "Enter number of employees: "; int n; cin >> n; Employee emp[n];    for(int i = 0; i < n; i++) { int id; string name; float salary;  cout << "Enter details for employee " << (i + 1) << endl; cout << "Employee ID: "; cin >> id; cout << "Name: "; cin.ignore(); getline(cin, name); cout << "Salary: "; cin >> salary;  emp[i].setData(id, name, salary);  } cout << "\nEmployee Details:\n"; for(int i = 0; i < n; i++) {  cout << "\nEmployee " << (i + 1) << ":\n"; emp[i].displayData();  }  return 0;  } |

Enter number of employees: 2

Enter details for employee 1

Employee ID: 001

Name: magellanic

Salary: 150000

Enter details for employee 2

Employee ID: 002

Name: glaive

Salary: 180000 Employee Details: Employee 1:

Employee ID: 1

Name: magellanic

Salary: 150000 Employee 2:

Employee ID: 2

Name: glaive

Salary: 180000

Q2: Design a class Student with members name, roll, and marks. Provide appropriate methods to read and display data.

Code:

|  |
| --- |
| #include <iostream> #include <string> using namespace std; class Student  { private: string name; int roll; float marks;  public: void setData()  {  cout << "Enter Name: "; cin.ignore(); getline(cin, name);  cout << "Enter Roll Number: "; cin >> roll;  cout << "Enter Marks: "; cin >> marks;  }  void displayData()  {  cout << "Name: " << name << endl; cout << "Roll Number: " << roll << endl; cout << "Marks: " << marks << endl;  } };  int main()  { int n;  cout << "Enter number of students: "; cin >> n;  Student \*students = new Student[n];    for (int i = 0; i < n; i++)  {  cout << "Enter details for student " << (i + 1) << endl; students[i].setData();  }  cout << "\nStudent Details:\n"; for (int i = 0; i < n; i++)  {  cout << "\nStudent " << (i + 1) << ":\n"; students[i].displayData();  }  delete[] students; return 0;  } |

Enter number of students: 2

Enter details for student 1

Enter Name: devdas

Enter Roll Number: 01

Enter Marks: 65

Enter details for student 2

Enter Name: champa

Enter Roll Number: 01

Enter Marks: 71 Student Details: Student 1:

Name: devdas

Roll Number: 1

Marks: 65 Student 2:

Name: champa

Roll Number: 1

Marks: 71

Q3: Write a program designing a class Customer with member variables firstName, lastName, address, and phoneNo. Add member functions to accept input and display these variables.

Code:

|  |
| --- |
| #include <iostream> #include <string> using namespace std; class Customer { private: string firstName; string lastName; string address; string phoneNo; public: void setData() {  cout << "Enter First Name: "; cin.ignore(); getline(cin, firstName); cout << "Enter Last Name: "; getline(cin, lastName); cout << "Enter Address: "; getline(cin, address); cout << "Enter Phone Number: "; getline(cin, phoneNo);  }  void displayData() {  cout << "Customer Details:\n";  cout << "First Name: " << firstName << endl; cout << "Last Name: " << lastName << endl; cout << "Address: " << address << endl; cout << "Phone Number: " << phoneNo << endl;  } }; int main() { int n;  cout << "Enter number of customers: "; cin >> n;  Customer\* customers = new Customer[n];    for (int i = 0; i < n; i++) {  cout << "Enter details for customer " << (i + 1) << endl; customers[i].setData();  }  cout << "\nCustomer Details:\n"; for (int i = 0; i < n; i++) {  cout << "\nCustomer " << (i + 1) << ":\n"; customers[i].displayData();  } delete[] customers; return 0;  } |

Enter number of customers: 2

Enter details for customer 1

Enter First Name: magellanic

Enter Last Name: glaive

Enter Address: hattiban

Enter Phone Number: 9860205359

Enter details for customer 2

Enter First Name: aditya

Enter Last Name: sharma

Enter Address: janakpur

Enter Phone Number: 9866384154 Customer Details: Customer 1:

Customer Details:

First Name: magellanic

Last Name: glaive

Address: hattiban

Phone Number: 9860205359 Customer 2:

Customer Details:

First Name: ditya

Last Name: sharma

Address: janakpur

Phone Number: 9866384154

Q4: Create a class Information to store name and address of students. Store information for two students and write a function to swap the contents of these two objects.

Code:

|  |
| --- |
| #include <iostream> #include <string> using namespace std; class Information { private: string name; string address; public: void setData() {  cout << "Enter Name: "; getline(cin, name); cout << "Enter Address: "; getline(cin, address);  }  void displayData() {  cout << "Name: " << name << endl; cout << "Address: " << address << endl;  }  void swap(Information& obj) {    string tempName = name; name = obj.name; obj.name = tempName;    string tempAddress = address; address = obj.address; obj.address = tempAddress;  }  };  int main(){  Information student1, student2;  cout << "Enter details for Student 1:\n"; student1.setData();  cout << "Enter details for Student 2:\n"; student2.setData();  cout << "\nBefore swapping:\n"; cout << "Student 1:\n"; student1.displayData(); cout << "Student 2:\n"; student2.displayData();    student1.swap(student2);    cout << "\nAfter swapping:\n"; cout << "Student 1:\n"; student1.displayData(); cout << "Student 2:\n"; student2.displayData(); return 0;  } |

|  |
| --- |
| Enter details for Student 1:  Enter Name: saswot  Enter Address: baneswor  Enter details for Student 2:  Enter Name: alish  Enter Address: ratnapark    Before swapping:  Student 1:  Name: saswot Address: baneswor  Student 2:  Name: alish  Address: ratnapark    After swapping: Student 1:  Name: alish  Address: ratnapark  Student 2:  Name: saswot  Address: baneswor |

Design a class to represent a bank account with members: depositor’s name, account number, and account type. Include member functions to initialize and display the depositor’s name and account type.

Code:

|  |
| --- |
| #include <iostream> #include <string> using namespace std; class BankAccount { private: string depositorName; string accountNumber; string accountType; public: void setData() {  cout << "Enter Depositor's Name: "; cin.ignore();  getline(cin, depositorName); cout << "Enter Account Number: "; getline(cin, accountNumber); cout << "Enter Account Type: "; getline(cin, accountType);  }  void displayData() {  cout << "Depositor's Name: " << depositorName << endl; cout << "Account Number: " << accountNumber << endl; cout << "Account Type: " << accountType << endl;  } };  int main() {  BankAccount acc1; acc1.setData();  cout << "\nBank Account Details:\n"; acc1.displayData(); return 0;  } |

Output:

Enter Depositor's Name: rajdip

Enter Account Number: 00003124353

Enter Account Type: fixed

Bank Account Details:

Depositor's Name: rajdip

Account Number: 00003124353

Account Type: fixed

Write a program to design a class Rectangle with constructors and member functions to calculate area and perimeter.

|  |
| --- |
| #include <iostream> using namespace std; class Rectangle  { private: double length; double width;  public:  Rectangle() { length = 0; width = 0;  }    Rectangle(double l, double w)  { length = l; width = w;  }  double area()  {return length \* width;  }  double perimeter()  { return 2 \* (length + width);  }  void display()  { cout << "Length: " << length << ", Width: " << width << endl;  cout << "Area: " << area() << ", Perimeter: " << perimeter() << endl;  } };  int main()  {  double l, w;  cout << "Enter length of rectangle: "; cin >> l;  cout << "Enter width of rectangle: "; cin >> w;  Rectangle rect(l, w); rect.display(); return 0;  } |

Output:

Enter length of rectangle: 5

Enter width of rectangle: 2

Length: 5, Width: 2

Area: 10, Perimeter: 14

Write a program demonstrating that destructors execute in the reverse order of constructors. Display the corresponding object IDs during construction and destruction.

Code:

|  |
| --- |
| #include <iostream> using namespace std; class MyClass { int value; public:  MyClass(int i) { value = i;  cout << "Constructor called for object at address: " << value << endl; }    ~MyClass() {  cout << "Destructor called for object at address: " << value << endl;  }  }; int main(){  cout << "Creating objects" << endl;  MyClass obj1(1);  MyClass obj2(2);  MyClass obj3(3);    cout << "end of main, destructors will be called now." << endl;    return 0;  } |

Output:

Creating objects

Constructor called for object at address: 1

Constructor called for object at address: 2 Constructor called for object at address: 3 end of main, destructors will be called now. Destructor called for object at address: 3

Destructor called for object at address: 2

Destructor called for object at address: 1

Create a class Time with members for hours, minutes, and seconds. Write a member function AddTime() that adds two Time objects passed as arguments and returns the result.

Code:

|  |
| --- |
| #include <iostream> using namespace std; class Time { private: int hours; int minutes; int seconds; public: void setData(){  cout << "Enter hours: "; cin >> hours;  cout << "Enter minutes: "; cin >> minutes;  cout << "Enter seconds: "; cin >> seconds;    }  Time AddTime(Time t2) { Time temp;  temp.seconds = seconds + t2.seconds;  temp.minutes = minutes + t2.minutes + (temp.seconds / 60); temp.seconds %= 60;  temp.hours = hours + t2.hours + (temp.minutes / 60); temp.minutes %= 60; return temp;  }  void display() {  cout << "Time: " << hours << "h " << minutes << "m " << seconds << "s" << endl;  } };  int main() {  Time t1, t2, result; cout << "Enter first time:\n"; t1.setData();  cout << "Enter second time:\n"; t2.setData();  result = t1.AddTime(t2);  cout << "Resulting time after addition:\n"; result.display(); return 0;  } |

Output:

|  |
| --- |
| Enter first time:  Enter hours: 30  Enter minutes: 2 Enter seconds: 67 Enter second time:  Enter hours: 3  Enter minutes: 11  Enter seconds: 56  Resulting time after addition: Time: 33h 15m 3s |

Design a class LandMeasure that stores Ropani, Ana, Paisa, and Dam. Write a member function to add two LandMeasure objects and return their sum as a new object.

Nepali Land Measurement Reference:

1 Ropani = 16 Ana

1 Ana = 4 Paisa 1 Paisa = 4 Dam

Code:

|  |
| --- |
| #include <iostream> using namespace std; class LandMeasure { private:  int ropani; int ana; int paisa; int dam; public:  LandMeasure() : ropani(0), ana(0), paisa(0), dam(0) {}    void setData() {  cout << "Enter Ropani: "; cin >> ropani; cout << "Enter Ana: "; cin >> ana;  cout << "Enter Paisa: "; cin >> paisa; cout << "Enter Dam: "; cin >> dam;  }    LandMeasure add(LandMeasure lm) { LandMeasure result; result.dam = dam + lm.dam;  result.paisa = paisa + lm.paisa + (result.dam / 4); result.dam %= 4;  result.ana = ana + lm.ana + (result.paisa / 4); result.paisa %= 4;  result.ropani = ropani + lm.ropani + (result.ana / 16); result.ana %= 16; return result;  }  void display() {  cout << "Land Measure: " << ropani << " Ropani, "  << ana << " Ana, "  << paisa << " Paisa, "  << dam << " Dam" << endl;  } };  int main() {  LandMeasure lm1, lm2, result; cout << "Enter first land measure:\n"; lm1.setData(); |

cout << "Enter second land measure:\n"; lm2.setData();

result = lm1.add(lm2);

cout << "Resulting land measure after addition:\n"; result.display();

return 0;

}

Output:

Enter first land measure:

Enter Ropani: 3

Enter Ana: 23

Enter Paisa: 533

Enter Dam: 3

Enter second land measure:

Enter Ropani: 4

Enter Ana: 23

Enter Paisa: 76

Enter Dam: 32

Resulting land measure after addition:

Land Measure: 19 Ropani, 8 Ana, 1 Paisa, 3 Dam

Q10: Create a class Employee and add a member function to increase the salary of each employee by 10%. Read and display the details of all employees before and after the salary increase.

Code:

|  |
| --- |
| #include <iostream> #include <string> using namespace std; class Employee { private: string name; double salary; public: void setData() {  cout << "Enter Employee Name: "; cin.ignore(); getline(cin, name); cout << "Enter Salary: "; cin >> salary;  }  void displayData() {  cout << "Name: " << name << ", Salary: " << salary << endl; } void increaseSalary() { salary \*= 1.10;  } }; int main() { int n;  cout << "Enter number of employees: "; cin >> n;    Employee\* employees = new Employee[n];    for (int i = 0; i < n; i++) {  cout << "Enter details for Employee " << (i + 1) << ":\n"; employees[i].setData();  } cout << "\nEmployee Details Before Salary Increase:\n"; for (int i = 0; i < n; i++) { employees[i].displayData();  }  for (int i = 0; i < n; i++) { employees[i].increaseSalary();  }  cout << "\nEmployee Details After Salary Increase:\n"; for (int i = 0; i < n; i++) { employees[i].displayData();  }  delete[] employees; return 0;  } |

Output:

|  |
| --- |
| Enter number of employees: 2 Enter details for Employee 1:  Enter Employee Name: saswot  Enter Salary: 120000  Enter details for Employee 2:  Enter Employee Name: aditya  Enter Salary: 130000    Employee Details Before Salary Increase:  Name: saswot, Salary: 120000  Name: aditya, Salary: 130000    Employee Details After Salary Increase:  Name: saswot, Salary: 132000  Name: aditya, Salary: 143000 |

Q11: Write a program to find the area of a square and a rectangle using classes. Initialize the objects dynamically using pointers.

Code:

|  |
| --- |
| #include <iostream> using namespace std; class square { private:  double side; public: void setData() {  cout << "Enter side of square: "; cin >> side;  }  double area() {  return side \* side;  }  void display() {  cout << "Square Side: " << side << ", Area: " << area() << endl;  } };  class rectangle { private: double length; double width; public: void setData() {  cout << "Enter length of rectangle: "; cin >> length;  cout << "Enter width of rectangle: "; cin >> width;  }  double area() {  return length \* width;  }  void display() {  cout << "Rectangle Length: " << length << ", Width: " << width << ", Area: " << area() << endl; } }; int main(){  square\* sq = new square;  rectangle\* rect = new rectangle;    sq->setData(); rect->setData();    cout << "\nSquare Details:\n"; sq->display(); cout << "\nRectangle Details:\n"; rect->display(); delete sq; delete rect; return 0;  } |

Output:

Enter side of square: 14

Enter length of rectangle: 8

Enter width of rectangle: 6 Square Details:

Square Side: 14, Area: 196

Rectangle Details:

Rectangle Length: 8, Width: 6, Area: 48

Q12: Write a program defining an inline member function calculateVolume() outside the class using the inline keyword for a class Box.

Code:

|  |
| --- |
| #include <iostream> using namespace std; class Box { private: double length; double width; double height; public: void setData() {  cout << "Enter length of the box: "; cin >> length;  cout << "Enter width of the box: "; cin >> width;  cout << "Enter height of the box: "; cin >> height;  } inline double calculateVolume() { return length \* width \* height;  }  void display() {  cout << "Box Dimensions: " << length << " x " << width << " x " << height << endl; cout << "Volume: " << calculateVolume() << endl;  } }; int main() { Box box; box.setData(); box.display(); return 0;  } |

Output:

Enter length of the box: 14.2

Enter width of the box: 8.7

Enter height of the box: 4.44

Box Dimensions: 14.2 x 8.7 x 4.44

Volume: 548.518

**Discussion:**

Classes and objects help us group data and related functions together in a neat and organized way. A class is like a template, and objects are the actual things we use in the program. By using classes, we can model real-life things (like employees, students, etc.) more easily. It also makes our code easier to manage and reuse. Overall, it’s a simple but powerful way to structure programs in C++.

**Conclusion:**

Classes and objects are key parts of C++ that make programs more organized and easier to understand. They help us keep related data and functions together, which is useful for writing clean and reusable code. Learning how to use them is an important step toward writing better programs and solving realworld problems with C++.