Module 3 - Introduction to OOPS Programming (Theory Exercise)

Shivam Gangwani

May 26, 2025

1 Introduction to C++

1.1 First C++ Program: Hello World

- Write a simple C++ program to display "Hello, World!".
- Objective: Understand the basic structure of a C++ program, including **#include**, **main()**, and **cout**.

```cpp
#include<iostream>
using namespace std;

main(){
    cout<<"Hello, World!";
}
```

1.2 Basic Input/Output

- Write a C++ program that accepts user input for their name and age and then displays a personalized greeting.
- Objective: Practice input/output operations using **cin** and **cout**.

```cpp
#include<iostream>
#include <string>
using namespace std;

main(){
    string name;
    int age;
    cout<<"Enter name: ";
    getline(cin, name);
    cout<<"Enter age: ";
    cin>>age;
    cout<<"Hello, "<<name<<". You are "<<age<<" years old.";
}
```

1.3 POP vs. OOP Comparison Program

- Write two small programs: one using Procedural Programming (POP) to calculate the area of a rectangle, and another using Object-Oriented Programming (OOP) with a class and object for the same task.
- Objective: Highlight the difference between POP and OOP approaches.

```
#include<iostream>
using namespace std;
int recarea(int l, int b){
    return l*b;
}
main(){
    int l, b;
    cout<<"Enter length: ";
    cin>>l;
    cout<<"Enter breadth: ";
    cin>>b;
    cout<<"Rectangle area = "<<recarea(l,b)<<endl;
}
```

```
#include<iostream>
using namespace std;
class recarea{
    public:
        int l, b;
        recarea(){
            cout<<"Enter length: ";
            cin>>l;
            cout<<"Enter breadth: ";
            cin>>b;
            cout<<"Area of rectangle = "<<l*b<<endl;
        }
};
main(){
    recarea ra;
}
```

1.4 Setting Up Development Environment

- Write a program that asks for two numbers and displays their sum. Ensure this is done after setting up the IDE (like Dev C++ or CodeBlocks).
- Objective: Help students understand how to install, configure, and run programs in an IDE.

```
#include<iostream>
using namespace std;
main(){
    int a, b;
    cout<<"Enter first number: ";
    cin>>a;
    cout<<"Enter second number: ";
    cin>>b;
    cout<<a<<" + "<<b<<" = "<<a+b<<endl;
}
```

2 Variables, Data Types, and Operators

2.1 Variables and Constants

- Write a C++ program that demonstrates the use of variables and constants. Create variables of different data types and perform operations on them.
- Objective: Understand the difference between variables and constants.

```
#include<iostream>
using namespace std;
main(){
    int a = 5;
    double d = 1.3;
    char c = 'a';
    string s = "hello";
    bool b = 1;
    const double p = 3.14;
    const string w = "world";
    d = p - d;
    s += " " + w;
}
```

2.2 Type Conversion

- Write a C++ program that performs both implicit and explicit type conversions and prints the results.
- Objective: Practice type casting in C++.

```
#include <iostream>
using namespace std;
main() {
    char c = 'a';
    int n = ++c;
    c = (char)n;
    cout<<n<<endl;
    cout<<c<<endl;
}
```

2.3 What are the different types of operators in C++? Provide examples of each.

- Write a C++ program that demonstrates arithmetic, relational, logical, and bitwise operators. Perform operations using each type of operator and display the results.
- Objective: Reinforce understanding of different types of operators in C++.

```
/*Write a C++ program that demonstrates arithmetic, relational,
logical, and bitwise operators. Perform operations using each
type of operator and display the results.*/
#include<iostream>
using namespace std;
main(){
    int a=10, b=5;
    cout<<a<<" + "<<b<<" = "<<a+b<<endl;
    cout<<a<<" - "<<b<<" = "<<a-b<<endl;
    cout<<a<<" * "<<b<<" = "<<a*b<<endl;
    cout<<a<<" / "<<b<<" = "<<a/b<<endl;
    cout<<a<<" % "<<b<<" = "<<a%b<<endl;
    cout<<"++"<<a<<" = "<<++a<<endl;
    cout<<"--"<<a<<" = "<<--a<<endl;
    cout<<a<<" == "<<b<<" = "<<(a==b)<<endl;
    cout<<a<<" != "<<b<<" = "<<(a!=b)<<endl;
    cout<<a<<" > "<<b<<" = "<<(a>b)<<endl;
    cout<<a<<" < "<<b<<" = "<<(a<b)<<endl;
    cout<<a<<" >= "<<b<<" = "<<(a>=b)<<endl;
    cout<<a<<" <= "<<b<<" = "<<(a<=b)<<endl;
    cout<<a<<" && "<<b<<" = "<<(a&&b)<<endl;
    cout<<a<<" || "<<b<<" = "<<(a||b)<<endl;
    cout<<"!"<<a<<" = "<<!a<<endl;
    cout<<a<<" & "<<b<<" = "<<(a&b)<<endl;
    cout<<a<<" | "<<b<<" = "<<(a|b)<<endl;
    cout<<a<<" ^ "<<b<<" = "<<(a^b)<<endl;
    cout<<a<<" << "<<b<<" = "<<(a<<b)<<endl;
    cout<<a<<" >> "<<b<<" = "<<(a>>b)<<endl;
    cout<<"~"<<a<<" = "<<~a<<endl;
}
```

3 Control Flow Statements

3.1 Grade Calculator

- Write a C++ program that takes a student's marks as input and calculates the grade based on if-else conditions.
- Objective: Practice conditional statements (**if-else**).

```cpp
#include<iostream>
using namespace std;
main(){
   int m;
   char g;
   cout<<"Enter student’s marks: ";
   cin>>m;
   if(m>=90){
        g=’A’;
    }
    else if(m>=80){
        g=’B’;
    }
    else if(m>=70){
        g=’C’;
    }
    else if(m>=60){
        g=’D’;
    }
    else{
        g=’F’;
    }
    cout<<"Grade = "<<g<<endl;
}
```

3.2 Number Guessing Game

- Write a C++ program that asks the user to guess a number between 1 and 100. The program should provide hints if the guess is too high or too low. Use loops to allow the user multiple attempts.
- Objective: Understand **while** loops and conditional logic.

```cpp
#include<iostream>
using namespace std;
main(){
    srand(time(0));
    int g, r=(rand()%100)+1;
    do{
        cout<<"Guess number between 1 and 100: ";
        cin>>g;
        if(g>r){
            cout<<g<<" is too high"<<endl;
        }
        else if(g<r){
            cout<<g<<" is too low"<<endl;
        }
        else{
            cout<<g<<" is the corrent guess"<<endl;
        }
    }
    while(g!=r);
}
```

3.3 Multiplication Table

- Write a C++ program to display the multiplication table of a given number using a for loop.
- Objective: Practice using loops.

```
#include<iostream>
using namespace std;
main(){
    int n, i;
    cout<<"Enter number: ";
    cin>>n;
    for(i=1;i<=10;i++){
        cout<<n<<" x "<<i<<" = "<<n*i<<endl;
    }
}
```

3.4 Nested Control Structures

- Write a program that prints a right-angled triangle using stars (*) with a nested loop.
- Objective: Learn nested control structures.

```
#include<iostream>
using namespace std;
main(){
    int i, j;
    for(i=1;i<6;i++){
        for(j=1;j<=i;j++){
            cout<<"*";
        }
        cout<<endl;
    }
}
```

4 Functions and Scope

4.1 Simple Calculator Using Functions

- Write a C++ program that defines functions for basic arithmetic operations (add, subtract, multiply, divide). The main function should call these based on user input.

- Objective: Practice defining and using functions in C++.

```cpp
#include<iostream>
using namespace std;
int add(int a, int b){
    return a+b;
}
int sub(int a, int b){
    return a-b;
}
int mul(int a, int b){
    return a*b;
}
int d(int a, int b){
    return a/b;
}
main(){
    int a,b;
    char o;
    cout<<"Enter first number: ";
    cin>>a;
    cout<<"Enter second number: ";
    cin>>b;
    cout<<"Enter operation: ";
    cin>>o;
    switch(o){
        case '+':
            cout<<add(a,b)<<endl;
            break;
        case '-':
            cout<<sub(a,b)<<endl;
            break;
        case '*':
            cout<<mul(a,b)<<endl;
            break;
        case '/':
            cout<<d(a,b)<<endl;
            break;
    }
}
```

4.2 Factorial Calculation Using Recursion

- Write a C++ program that calculates the factorial of a number using recursion.
- Objective: Understand recursion in functions.

```cpp
#include<iostream>
using namespace std;
int f(int n){
    if(n==1){
        return 1;
    }
    else{
        return (n*f(n-1));
    }
}
main(){
    int n;
    cout<<"Enter number: ";
    cin>>n;
    cout<<n<<"! = "<<f(n);
}
```

4.3 Variable Scope

- Write a program that demonstrates the difference between local and global variables in C++. Use functions to show scope.
- Objective: Reinforce the concept of variable scope.

```
#include<iostream>
using namespace std;
int g=5;
void func1(){
    int l=3;
    cout<<"local variable: "<<l<<endl;
    cout<<"global variable: "<<g<<endl;
}
main(){
    func1();
    cout<<"local variable: "<<l<<endl;  //gives error
    int l=7;
    cout<<"local variable: "<<l<<endl;
    cout<<"global variable: "<<g<<endl;
}
```

5 Arrays and Strings

5.1 Array Sum and Average

- Write a C++ program that accepts an array of integers, calculates the sum and average, and displays the results.
- Objective: Understand basic array manipulation.

```
#include<iostream>
using namespace std;
```

```cpp
main(){
    int a[5], i, s=0;
    for(i=0;i<5;i++){
        cout<<"Enter "<<i<<"th element: ";
        cin>>a[i];
        s+=a[i];
    }
    cout<<"sum = "<<s<<endl;
    cout<<"average = "<<s/5<<endl;
}
```

5.2 Matrix Addition

- Write a C++ program to perform matrix addition on two 2x2 matrices.
- Objective: Practice multi-dimensional arrays.

```cpp
#include<iostream>
using namespace std;
main(){
    int a[2][2], b[2][2], i, j;
    for(i=0;i<2;i++){
        for(j=0;j<2;j++){
            cout<<"Enter ["<<i+1<<"]["<<j+1<<"] element of 1st
               matrix: ";
            cin>>a[i][j];
        }
    }
    for(i=0;i<2;i++){
        for(j=0;j<2;j++){
            cout<<"Enter ["<<i+1<<"]["<<j+1<<"] element of 2nd
               matrix: ";
            cin>>b[i][j];
        }
    }
    cout<<"sum = "<<endl;
    for(i=0;i<2;i++){
        for(j=0;j<2;j++){
            cout<<a[i][j]+b[i][j]<<"  ";
        }
        cout<<endl;
    }
}
```

5.3 String Palindrome Check

- Write a C++ program to check if a given string is a palindrome (reads the same forwards and backwards).
- Objective: Practice string operations.

```
#include<iostream>
#include<string>
#include<cmath>
using namespace std;
main(){
    string s;
    cout<<"Enter string: ";
    cin>>s;
    int i, j, l=s.length()-1;
    for(i=0, j=l; i<l, j>=0;i++, j--){
        if(s[i] != s[j]){
            cout<<s<<" is not a palindrome";
            break;
        }
    }
    if(i==l+1){
        cout<<s<<" is a palindrome";
    }
}
```

6 Introduction to Object-Oriented Programming

6.1 Class for a Simple Calculator

- Write a C++ program that defines a class **Calculator** with functions for addition, subtraction, multiplication, and division. Create objects to use these functions.
- Objective: Introduce basic class structure.

```
#include<iostream>
using namespace std;
class Calculator{
    public:
        int addition(int a, int b){
            return a+b;
        }
        int subtraction(int a, int b){
            return a-b;
        }
        int multiplication(int a, int b){
            return a*b;
        }
        int division(int a, int b){
            return a/b;
        }
};
main(){
    int a, b;
    cout<<"Enter 1st number: ";
    cin>>a;
    cout<<"Enter 2nd number: ";
    cin>>b;
    Calculator o;
    cout<<a<<" + "<<b<<" = "<<o.addition(a,b)<<endl;
    cout<<a<<" - "<<b<<" = "<<o.subtraction(a,b)<<endl;
    cout<<a<<" * "<<b<<" = "<<o.multiplication(a,b)<<endl;
    cout<<a<<" / "<<b<<" = "<<o.division(a,b)<<endl;
}
```

6.2 Class for Bank Account

- Create a class **BankAccount** with data members like balance and member functions like **deposit** and **withdraw**. Implement encapsulation by keeping the data members private.
- Objective: Understand encapsulation in classes

```
#include<iostream>
using namespace std;
class BankAccount{
    private:
        int balance;
    public:
        void deposit(int b){
            int d;
            cout<<"Enter amount to deposit: ";
            cin>>d;
            balance=b;
            cout<<"Balance after deposit: "<<balance+d<<endl;
        }
        void withdraw(int b){
            int w;
            cout<<"Enter amount to withdraw: ";
            cin>>w;
            balance=b;
            cout<<"Balance after withdraw: "<<balance-w<<endl;
        }
};
main(){
    int b, ch;
    cout<<"Enter current balance: ";
    cin>>b;
    BankAccount o;
    cout<<"1. Deposit"<<endl;
    cout<<"2. Withdraw"<<endl;
    cout<<"3. Exit"<<endl;
    cout<<"Enter your choice: ";
    cin>>ch;
    switch(ch){
        case 1:
            o.deposit(b);
            break;
        case 2:
            o.withdraw(b);
            break;
        case 3:
            cout<<"Thank you"<<endl;
            break;
        default:
            cout<<"Invalid input"<<endl;
            break;
    }
}
```

6.3 Inheritance Example

- Write a program that implements inheritance using a base class **Person** and derived classes **Student** and **Teacher**. Demonstrate reusability through inheritance.
- Objective: Learn the concept of inheritance.

```
#include<iostream>
using namespace std;
class Person{
    public:
        string n;
        int a;
        void inP(string t){
            cout<<"Enter "<<t<<" name: ";
            cin>>n;
            cout<<"Enter "<<t<<" age: ";
            cin>>a;
        }
};
class Student: public Person{
    public:
        int c;
        string t="student's";
        void inS(){
            inP(t);
            cout<<"Enter student's class: ";
            cin>>c;
        }
        void outS(){
            cout<<"Student's name: "<<n<<endl;
            cout<<"Student's age: "<<a<<endl;
            cout<<"Student's class: "<<c<<endl;
        }
};
```

```
class Teacher: public Person{
    public:
        string s, t="teacher's";
        void inT(){
            inP(t);
            cout<<"Enter teacher's subject: ";
            cin>>s;
        }
        void outT(){
            cout<<"Teacher's name: "<<n<<endl;
            cout<<"Teacher's age: "<<a<<endl;
            cout<<"Teacher's subject: "<<s<<endl;
        }
};
main(){
    Student o1;
    Teacher o2;
    o1.inS();
    o2.inT();
    o1.outS();
    o2.outT();
}
```