**ASSIGNMENT 1**

**NAME**- SAPNADARSHI SINGHA

**STREAM**- COMPUTER SCIENCE AND BUSINESS SYSTEM

**YEAR**- SECOND (FOURTH SEMESTER)

**ROLL NUMBER**- 18731122014

**SUBJECT NAME**- OBJECT ORIENTED PROGRAMMING WITH C++ LAB

**SUBJECT CODE-** PCC CSBS 492

1. **Write a program in C that will tabulate the marks of three different subjects and print the total and percentage of marks for three students. Use structure to store student details like Roll no, Name and marks and user defined functions tabulate (), and display ().**

#include <stdio.h>

#include <string.h>

int i;

struct Student {

int rollNo;

char name[50];

float marks[5];

};

void tabulate(struct Student\* student) {

student->marks[3] =student->marks[0] + student->marks[1] + student->marks[2];

student->marks[4] = student->marks[3] / 3;

}

void display(struct Student student) {

printf("Roll No: %d\n", student.rollNo);

printf("Name: %s\n", student.name);

printf("Subject Marks: %.2f, %.2f, %.2f\n", student.marks[0], student.marks[1], student.marks[2]);

printf("Total Marks: %.2f\n", student.marks[3]);

printf("Percentage: %.2f%%\n", student.marks[4]);

}

int main() {

struct Student students[3];

for ( i = 0; i < 3; ++i) {

printf("Enter details for student %d:\n", i + 1);

printf("Roll No: ");

scanf("%d", &students[i].rollNo);

printf("Name: ");

scanf("%s", students[i].name);

printf("Marks in three subjects: ");

scanf("%f %f %f", &students[i].marks[0], &students[i].marks[1], &students[i].marks[2]);

tabulate(&students[i]);

}

printf("\nStudent Details:\n");

for ( i = 0; i < 3; ++i) {

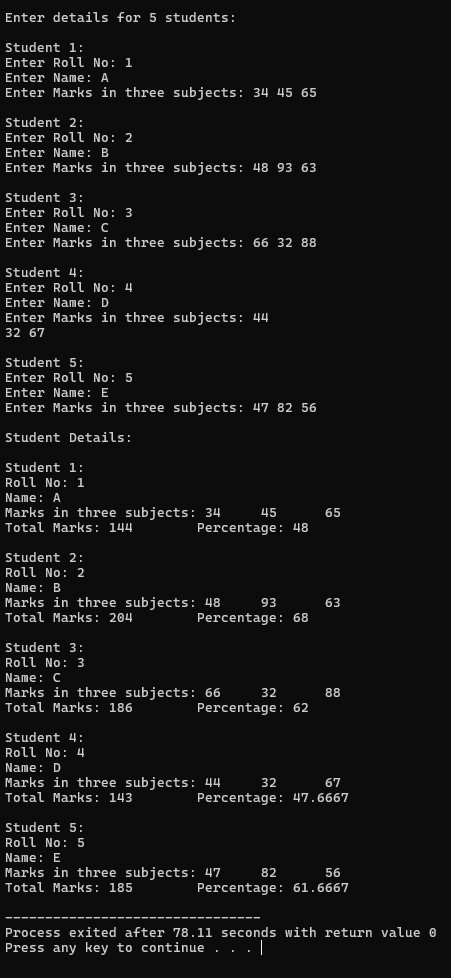
printf("\nStudent %d:\n", i + 1);

display(students[i]);

}

return 0;

}



1. **Design a class STUDENT that will have the following data members and member functions.**

**Data member – Name, Roll No, Marks of three subjects.**

**Member functions- getdata (), putdata (), tabulate().Write a program using C++ to calculate the total and percentage of marks for three different students.**

#include <iostream>

#include <string>

using namespace std;

class STUDENT {

public:

string name;

int rollNo;

int marks[3];

void getdata() {

cout << "Enter the Name of the Student: ";

cin >> name;

cout << "Enter the Roll Number of the Student: ";

cin >> rollNo;

cout << "Enter the Marks of Three Subjects: ";

for (int i = 0; i < 3; i++) {

cin >> marks[i];

}

}

void putdata() {

cout << "\nName of Student: " << name;

cout << "\nRoll No: " << rollNo;

cout << "\nMarks in Subjects: ";

for (int i = 0; i < 3; i++) {

cout << marks[i] << " ";

}

}

void tabulate() {

int total = 0;

for (int i = 0; i < 3; i++) {

total += marks[i];

}

float percentage = (float)total / 3;

cout << "\nTotal Marks = " << total;

cout << "\nPercentage = " << percentage<<endl;

}

};

int main() {

STUDENT s[3];

for(int k=0;k<3;k++){

cout << "Enter details for Student "<<k+1<<":\n";

s[k].getdata();

}

for(int k=0;k<3;k++){

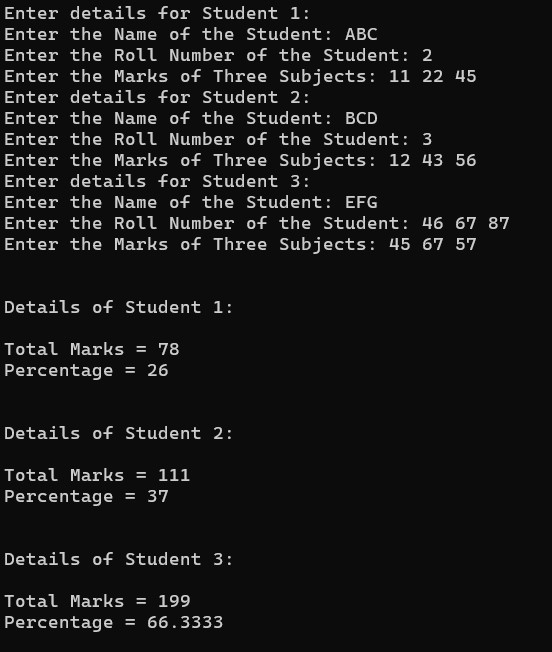
cout << "\n\nDetails of Student "<<k+1<<":\n";

s[k].tabulate();

}

return 0;

}



1. Modify the above Program to tabulate the marks for 5 students using array of objects.

#include <iostream>

#include <string>

using namespace std;

class STUDENT {

public:

string name;

int rollNo;

int marks[3];

void getdata() {

cout << "Enter the Roll Number: ";

cin >> rollNo;

cout << "Enter the Name: ";

cin >> name;

cout << "Enter the Marks: ";

for (int i = 0; i < 3; i++) {

cin >> marks[i];

}

}

void putdata() {

cout << "\nName of Student: " << name;

cout << "\nRoll No: " << rollNo;

cout << "\nMarks in Subjects: ";

for (int i = 0; i < 3; i++) {

cout << marks[i] << " ";

}

}

void tabulate() {

int total = 0;

for (int i = 0; i < 3; i++) {

total += marks[i];

}

float percentage = (float)total / 3;

cout << "\nTotal Marks: " << total<<endl;

cout << "\nPercentage: " << percentage<<endl;

}

};

int main() {

STUDENT s[5];

for(int k=0;k<5;k++){

cout << "Enter details: "<<k+1<<":\n";

s[k].getdata();

}

for(int k=0;k<5;k++){

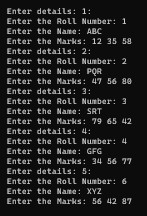
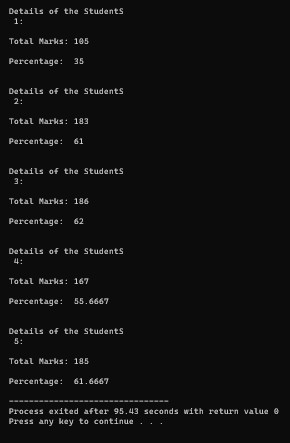
cout << "\n\nDetails of the StudentS\n "<<k+1<<":\n";

s[k].tabulate();

}

return 0;

}

1. **Explain the features of Object Oriented Language that C++ supports.**

C++ is a powerful and versatile object-oriented programming language that supports several features inherent to the object-oriented paradigm. Below are the key features of objectoriented programming that C++ supports:

1. **Classes and Objects**: C++ allows the definition of user-defined types using classes. Objects are instances of classes that encapsulate data (attributes) and methods (functions) that operate on that data. This feature promotes code organization and enhances modularity and reusability.
2. **Encapsulation:** Encapsulation is the bundling of data and methods that operate on the data within a single unit, i.e., a class. C++ supports access specifiers (public, private, and protected) to control the visibility of class members, allowing data to be hidden from the outside world and accessed only through well-defined interfaces.
3. **Inheritance**: Inheritance is a mechanism by which a class (subclass or derived class) can inherit attributes and methods from another class (superclass or base class). C++ supports single, multiple, hierarchical, and multilevel inheritance. Inheritance promotes code reuse and enables the creation of class hierarchies with shared functionality.
4. **Polymorphism**: Polymorphism allows objects of different classes to be treated as objects of a common superclass. C++ supports two types of polymorphism:

* Compile-time (Static) Polymorphism: Achieved through function overloading and operator overloading.
* Run-time (Dynamic) Polymorphism: Achieved through virtual functions and function overriding. C++ uses virtual functions and inheritance to enable dynamic binding and late binding, allowing the appropriate function to be invoked at runtime based on the actual object type.

1. **Abstraction:** Abstraction refers to the process of hiding complex implementation details and presenting only the essential features of an object. C++ supports abstraction through classes and interfaces, allowing programmers to create abstract data types and define interfaces without exposing internal implementation details.
2. **Data Hiding:** C++ supports data hiding by allowing class members to be declared as private, preventing direct access from outside the class. This encapsulation ensures that the internal state of an object is protected and can only be accessed or modified through designated methods. Message Passing: In an object-oriented paradigm, objects communicate with each other by sending messages. C++ supports message passing through method calls, where an object invokes a method on another object to request a service or perform an operation.
3. **Depict the basic differences between Object Oriented Programming and Procedure Oriented programming.**

Object-Oriented Programming (OOP) and Procedure-Oriented Programming (POP) are two different programming paradigms with distinct approaches to designing and implementing software systems. Here are the basic differences between the two:

1. **Primary Focus**:

* The primary focus of OOP is on objects. It emphasizes organizing code into objects, each representing a real-world entity, with attributes (data) and behaviors (methods).
* The primary focus of POP is on procedures or functions. It revolves around breaking down a problem into a set of sequential steps or procedures and then implementing those procedures using functions.

1. **Data and Functions**:

* In OOP, data and functions (methods) are encapsulated within objects. Objects interact with each other by sending messages and invoking methods.
* In POP, data and functions are separate entities. Data is stored in variables, and functions operate on that data. Functions may have access to global data, and data flow is typically managed through function parameters and return values.

1. **Abstraction:**

* OOP provides mechanisms like classes and interfaces for abstraction. Classes encapsulate data and methods, hiding the internal implementation details and exposing only necessary interfaces.
* Abstraction in POP is achieved through functions, where each function abstracts a specific task or operation. However, data and functions may not be tightly bound together as in OOP.

1. **Inheritance:**

* Inheritance is a fundamental concept in OOP, allowing the creation of new classes based on existing ones. It promotes code reuse and allows for the creation of hierarchical relationships between classes.
* POP typically does not have built-in support for inheritance. Code reuse is achieved through procedural decomposition, where common functionalities are implemented as separate functions and reused across different parts of the program.

1. **Polymorphism:**

* OOP supports both compile-time polymorphism (function overloading, operator overloading) and run-time polymorphism (virtual functions, function overriding).
* POP does not inherently support polymorphism. Polymorphic behavior can be achieved to some extent through function overloading, but it's not as flexible or powerful as in OOP.

1. **Data Hiding**:

* OOP promotes data hiding by allowing data members to be encapsulated within objects and accessed only through member functions, thus preventing direct manipulation.
* POP does not inherently support data hiding. Data can be declared as global or within functions, making it accessible throughout the program.run-time polymorphism (virtual functions, function overriding). • POP: POP does not inherently support polymorphism. Polymorphic behavior can be achieved to some extent through function overloading, but it's not as flexible or powerful as in OOP.