Day 2: Assignment

1. Write a class Student with a default constructor that initializes the student's name to "Unknown" and age to 0. Add a method display to print the student's details.

#include <iostream>

using namespace std;

class Student {

public:

string name;

int age;

Student() {

name = "Unknown";

age = 0;}

void display() {

cout << "Name: " << name << endl;

cout << "Age: " << age << endl;}

};

int main()

{

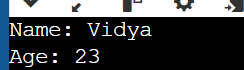
Student student1;

cout << "Student details:" << endl;

student1.display();

return 0;

}



1. Write a class Rectangle with a parameterized constructor that initializes the length and width. Add a method area that returns the area of the rectangle.

#include <iostream>

class Rectangle {

private:

double length;

double width;

public:

Rectangle(double len, double wid) : length(len), width(wid) {}

double area() const {

return length \* width;

}

};

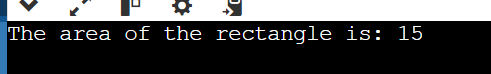
int main() {

Rectangle rect(5.0, 3.0);

std::cout << "The area of the rectangle is: " << rect.area() << std::endl;

return 0;

}



1. Write a class Book that has both a default constructor and a parameterized constructor. The default constructor should set the title to "Unknown" and the number of pages to 0. The parameterized constructor should initialize the title and pages with given values.

#include <iostream>

#include <string>

class Book {

private:

std::string title;

int pages;

public:

Book() : title("Unknown"), pages(0) {} // Default constructor

Book(const std::string& t, int p) : title(t), pages(p) {}// Parameterized constructor

void display() const { //book details.

std::cout << "Title: " << title << ", Pages: " << pages << std::endl;

}

};

int main() {

Book defaultBook;

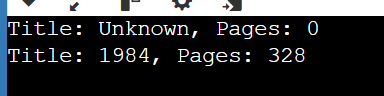
defaultBook.display();

Book customBook("1984", 328);

customBook.display(); // Output: Title: 1984, Pages: 328

return 0;

}



1. Write a function increment that takes a pointer to an integer and increments its value by 1. Demonstrate the function in the main program.

#include <iostream>

void increment(int\* ptr) { // Function to increment the value of an integer using a pointer

if (ptr != nullptr) {

(\*ptr)++;

}

}

int main() {

int value = 20;

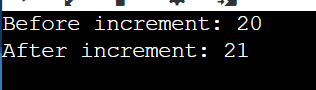
std::cout << "Before increment: " << value << std::endl; // Output: Before increment: 20

increment(&value);

std::cout << "After increment: " << value << std::endl; // Output: After increment: 21

return 0;

}



1. Write a class Circle with a method area. Create a pointer to an object of this class and call the area method using the pointer.

#include <iostream>

#include <cmath>

class Circle {

private:

double radius;

public:

Circle(double r) : radius(r) {} // Constructor to initialize the radius

double area() const { // Method to calculate the area

return M\_PI \* radius \* radius;

}

};

int main() {

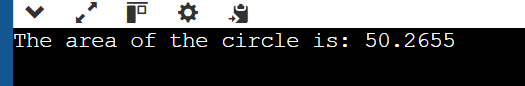
Circle circle(4.0); // Creating a Circle object

Circle\* circlePtr = &circle; //Creating a pointer to the Circle object

std::cout << "The area of the circle is: " << circlePtr->area() << std::endl; // Calling the area method using the pointer

return 0;

}



1. Write a program that creates an array of pointers to integers. Initialize the array with values and print them using the pointers.

#include <iostream>

int main() {

const int size = 10; // No.of elements in the array

int\* intPtrArray[size]; // Creating an array of pointers to integers

for (int i = 0; i < size; ++i) { // Allocating memory and initializing values

intPtrArray[i] = new int(i + 1); // Initializing with values 1, 2, 3, 4, 5

}

std::cout << "Values in the array of pointers:" << std::endl; // Print the values using the pointers

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

std::cout << \*intPtrArray[i] << " ";

}

std::cout << std::endl;

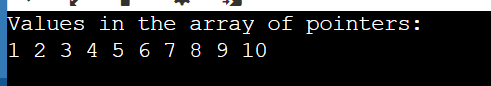
for (int i = 0; i < size; ++i) { // Deallocate the memory

delete intPtrArray[i];

}

return 0;

}



1. Write a function that takes a pointer to an array of integers and the size of the array. The function should print all elements of the array.

#include <iostream>

void printArray(int\* array, int size) { // Function to print all elements of an array

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

std::cout << array[i] << " ";

}

std::cout << std::endl;

}

int main() {

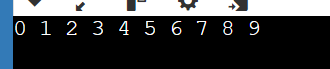
int arr[] = {0,1, 2, 3, 4, 5,6,7,8,9};

int size = sizeof(arr) / sizeof(arr[0]);

printArray(arr, size); // Calling the function

return 0;

}



1. Write a function swap that takes two integer references and swaps their values. Demonstrate the function in the main program.

#include <iostream>

void swap(int& a, int& b) {

int temp = a;

a = b;

b = temp;

}

int main() {

int x = 25;

int y = 35;

std::cout << "Before swap:" << std::endl;

std::cout << "x = " << x << ", y = " << y << std::endl;

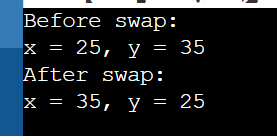
swap(x, y);

std::cout << "After swap:" << std::endl;

std::cout << "x = " << x << ", y = " << y << std::endl;

return 0;

}



1. Write a class Box with a method volume. Create an object of this class and a reference to this object. Call the volume method using the reference.

#include <iostream>

class Box {

private:

double length;

double width;

double height;

public:

Box(double l, double w, double h) : length(l), width(w), height(h) {} //constructor for the dimensions

double volume() const { //method to calculate volume

return length \* width \* height;

}

};

int main() {

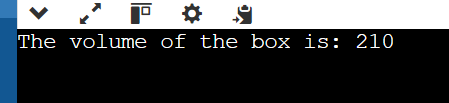
Box myBox(5.0, 6.0, 7.0); // Creating a Box object

Box& boxRef = myBox; // Creating a reference to the Box object

std::cout << "The volume of the box is: " << boxRef.volume() << std::endl; // Calling the volume method using the reference

return 0;

}



1. Write a function addTen that takes an integer by value and adds 10 to it. Demonstrate how the original value is not changed after calling the function.

#include <iostream>

void addTen(int num) { // Function to add 10 to an integer (passed by value)

num += 10;

std::cout << "Inside addTen function: num = " << num << std::endl;

}

int main() {

int originalValue = 35;

std::cout << "Before calling addTen function: originalValue = " << originalValue << std::endl;

addTen(originalValue); // Call addTen function

std::cout << "After calling addTen function: originalValue = " << originalValue << std::endl; //originalValue remains unchanged

return 0;

}

