C++taskprograms

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.

Code:

#include <iostream>

#include <string>

using namespace std;

// Define the Student class

class Student {

public:

string name; // Member variable to store the student's name

int age; // Member variable to store the student's age

// Default constructor

Student() {

name = "Unknown"; // Initialize name to "Unknown"

age = 0; // Initialize age to 0

}

// Method to display the student's details

void display() {

cout << "Name: " << name << ", Age: " << age << endl;

}

};

int main() {

// Create a Student object using the default constructor

Student student;

// Display the student's details

student.display();

return 0;

}

2. 2. \*Parameterized Constructor:\*

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>

using namespace std;

// Define the Rectangle class

class Rectangle {

public:

double length; // Member variable to store the length of the rectangle

double width; // Member variable to store the width of the rectangle

// Parameterized constructor

Rectangle(double l, double w) {

length = l; // Initialize length with the provided value

width = w; // Initialize width with the provided value

}

// Method to calculate the area of the rectangle

double area() {

return length \* width; // Calculate and return the area (length \* width)

}

};

int main() {

// Create a Rectangle object using the parameterized constructor

Rectangle rect(5.0, 3.0);

// Calculate and display the area of the rectangle

cout << "Area of the rectangle: " << rect.area() << endl;

return 0;

}

3. 3. \*Multiple Constructors:\*

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>

using namespace std;

// Define the Book class

class Book {

public:

string title;

int pages;

// Default constructor

Book() {

title = "Unknown";

pages = 0;

}

// Parameterized constructor

Book(string t, int p) {

title = t;

pages = p;

}

// Method to display the book's details

void display() {

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

}

};

int main() {

Book defaultBook; // Using default constructor

Book paramBook("The Great Gatsby", 180); // Using parameterized constructor

defaultBook.display();

paramBook.display();

return 0;

}

5. \*Pointer to an Integer:\*

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>

using namespace std;

// Function to increment the value of an integer by 1

void increment(int \*p) {

(\*p)++; // Dereference the pointer and increment the value by 1

}

int main() {

int num = 5; // Declare an integer variable and initialize it to 5

cout << "Before increment: " << num << endl; // Print the value of num before increment

increment(&num); // Call the increment function, passing the address of num

cout << "After increment: " << num << endl; // Print the value of num after increment

return 0;

}

6. Array of Pointers:\*

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

#include <iostream>

using namespace std;

int main() {

const int size = 5; // Define the size of the array

int values[size] = {10, 20, 30, 40, 50}; // Initialize an array of integers

int\* ptrArray[size]; // Declare an array of pointers to integers

// Initialize the array of pointers with the addresses of the elements in values array

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

ptrArray[i] = &values[i];

}

// Print the values using the array of pointers

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

cout << "Value at ptrArray[" << i << "] = " << \*ptrArray[i] << endl;

}

return 0;

}

7. 10. \*Pointer to a Class:\*

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>

using namespace std;

// Define the Circle class

class Circle {

public:

double radius; // Variable to store the radius of the circle

// Constructor to initialize the radius

Circle(double r) {

radius = r;

}

// Method to calculate the area of the circle

double area() {

return 3.14 \* radius \* radius; // πr^2

}

};

int main() {

Circle c(5.0); // Create a Circle object with radius 5.0

Circle\* ptr = &c; // Create a pointer to the Circle object

// Call the area method using the pointer and print the result

cout << "Area of the circle: " << ptr->area() << endl;

return 0;

}

8. Array of Pointers:\*

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

#include <iostream>

using namespace std;

int main() {

const int size = 5; // Define the size of the array

int values[size] = {10, 20, 30, 40, 50}; // Initialize an array of integers with values

int\* ptrArray[size]; // Declare an array of pointers to integers

// Initialize the array of pointers with the addresses of the elements in the values array

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

ptrArray[i] = &values[i];

}

// Print the values using the array of pointers

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

cout << "Value at ptrArray[" << i << "] = " << \*ptrArray[i] << endl;

}

return 0;

}

9. \*Pointer to an Array:\*

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>

using namespace std;

// Function to print all elements of an array

void printArray(int\* arr, int size) {

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

cout << arr[i] << " ";

}

cout << endl;

}

int main() {

const int size = 5; // Define the size of the array

int values[size] = {10, 20, 30, 40, 50}; // Initialize an array of integers

// Call the function to print the array

printArray(values, size);

return 0;

}

10. \*Dynamic Memory Allocation:\*

Write a program that dynamically allocates memory for an integer, assigns a value to it, and then frees the memory.

#include <iostream>

using namespace std;

int main() {

// Dynamically allocate memory for an integer

int\* ptr = new int;

// Assign a value to the allocated memory

\*ptr = 42;

// Print the value

cout << "Value of the dynamically allocated integer: " << \*ptr << endl;

// Free the allocated memory

delete ptr;

return 0;

}