1. \*Default Constructor:\*

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>

using namespace std;

class Student {

public:

string name;

int age;

Student() : name("Unknown"),

age(0) {}

void display() const {

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

}

};

int main() {

Student stu;

stu.display();

return 0;

}

* Firstly, we created an class Student and we declare the name and age.
* And we given a default Constructor name to “Unknown” and Age to “0”.
* Display method prints the name and age.
* And the output will be "Name: Unknown, Age: 0"

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> //header files

using namespace std;

class Rectangle { // Declaring a class rectangle . It is an parameterized constructor

public:

double length;

double width;

Rectangle(int l, int w) : length(l), width(w) {}

double area() const {

return length \* width; // Method to calculate the area of the rectangle

}

};

int main() {

Rectangle rect(7.0, 3.0); // Declaring values of length and width in an object rect

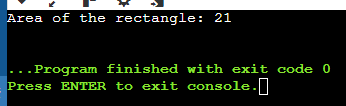
cout << "Area of the rectangle: " << rect.area() << endl; // Displaying the area of the rectangle

return 0;

}

Output:

Area of Rectangle: 21



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;

class Book { //Declaring a class book and declaring the title and pages

public:

string title;

int pages;

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

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

void display() const { // Method to display book details

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

}

};

int main() { //Main method for implementing.

Book Book1; // Book using the default constructor

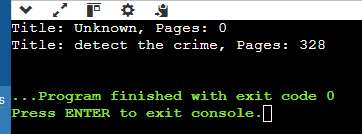
Book1.display(); // Title: Unknown, Pages: 0

Book Book2 ("detect", 328); // Book using the parameterized constructor

Book2.display(); // Title: Detect the crime, Pages: 328

return 0;

}



4. \*Constructor Overloading:\*

Write a class Complex that represents complex numbers. Implement a default constructor that sets both real and imaginary parts to 0, and a parameterized constructor that takes two arguments to initialize the real and imaginary parts.

#include <iostream>

using namespace std;

class Complex { // Declaring class Public and for real and imaginary objects

public:

double real;

double imaginary;

Complex() : real(0), imaginary(0) {} // written for Default constructor

Complex(double r, double i) : real(r), imaginary(i) {} // Declared for Parameterized constructor

void display() const {

cout << real << " + " << imaginary << "i" << endl; // Method

}

};

int main() { //Main method

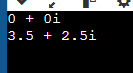
Complex Complex1;

Complex1.display(); // Output: "0 + 0i"

Complex Complex2(3.5, 2.5);

Complex2.display(); // Output:"3.5 + 2.5i"

return 0; }



### Pointers

9. \*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;

void increment(int\* ptr) { // Function to increment the value

if (ptr) {

(\*ptr)++;

}

}

int main() {

int num = 9;

cout << "Before increment: " << num << endl;

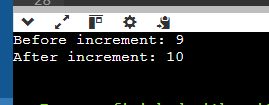
increment(&num); // Calling the increment function with the address of num

cout << "After increment: " << num << endl;

return 0;

}

Output:



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>

#include <cmath>

using namespace std;

class Circle { //Creating a Class Circle and declaring values

public:

double radius;

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

double area() const {

return M\_PI \* radius \* radius; // Method to calculate the area of the circle

}

};

int main() {

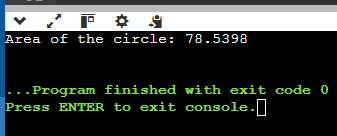
Circle circle1(5); // Giving an value of radius 5.0

Circle\* circlePtr = &circle1; // Creating a pointer to the Circle object

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

return 0;

}



11. \*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() {

int size = 5; // Declaring and initializing the Size of the array

int\* array[size]; // Declaring array to pointer

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

array[i] = new int(i \* 2); // Initializing the array with values & dynamically allocate memory

}

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

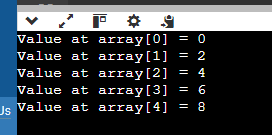
cout << "Value at array[" << i << "] = " << \*array[i] << endl; // Printing the values

}

return 0;

}

Output:



12. \*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;

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

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

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

}

}

int main() {

int size = 6; // Size of the array

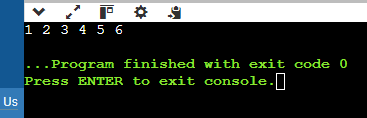
int values[size] = {1, 2, 3, 4, 5,6}; // Initialized the value of arrays

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

return 0;

}

Output:



13. \*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() {

int \*ptr = new int; // Dynamically allocate memory for an integer

if (ptr == nullptr) { //Check if memory allocation was successful

cout << "Failed to allocate memory!" << endl;

return 1;

}

\*ptr = 10; // Assign a value to the allocated memory

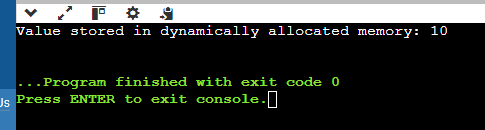
cout << "Value stored in dynamically allocated memory: " << \*ptr << endl; // Print the value

delete ptr; // Free the allocated memory

return 0;

}

Output:



### References

14. \*Reference to an Integer:\*

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

#include <iostream>

using namespace std;

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

int temp = a; // Function to swap the values of two integers

a = b;

b = temp;

}

int main() {

int x = 5; // Declare the values to swap

int y = 10;

cout << "Before swap: x = " << x << ", y = " << y << endl; // Before swap

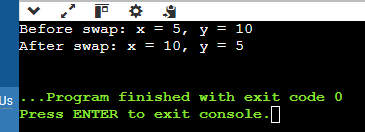
swap(x, y); // Calls the function

cout << "After swap: x = " << x << ", y = " << y << endl; //After swap

return 0;

}

Output:



15. \*Reference to a Class Object:\*

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>

using namespace std;

class Box { // Declaring a class box and values.

public:

double length;

double width;

double height;

Box(double l, double w, double h) : length(l), width(w), height(h) {} // Constructor

double volume() {

return length \* width \* height; //Method to calculate the volume

}

};

int main() {

Box box1(3.0, 4.0, 5.0); //Declaring values

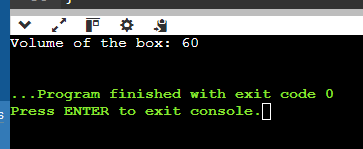
Box &boxRef = box1; //Using the reference

cout << "Volume of the box: " << boxRef.volume() << endl; // Calling the volume method

return 0;

}

Output:



16. \*Returning Reference from a Function:\*

Write a function that takes an array of integers and returns a reference to the largest element. Demonstrate the function in the main program.

### Pass by Value and Reference

17. \*Pass by Value:\*

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>

using namespace std;

void addTen(int num) {

num += 10; // Function to add 10 to an integer passed by value

}

int main() {

int originalValue = 5;

cout << "Original value: " << originalValue << endl;

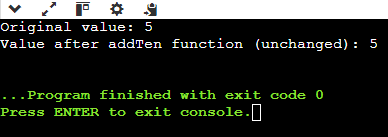
addTen(originalValue);

cout << "Value after addTen function (unchanged): " << originalValue << endl; // After function

return 0;

}

Output:



18. \*Pass by Reference:\*

Write a function addTenRef that takes an integer by reference and adds 10 to it. Demonstrate how the original value is changed after calling the function.

#include <iostream>

using namespace std;

void addTenRef(int &num) {

num += 10; // Function to add 10 to an integer passed by reference

}

int main() {

int originalValue = 5;

cout << "Original value: " << originalValue << endl; //original value

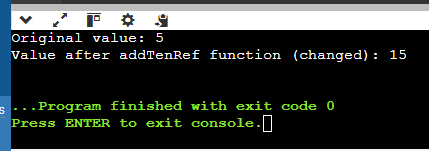
addTenRef(originalValue); // Call the function addTenRef with the original value

cout << "Value after addTenRef function (changed): " << originalValue << endl;

return 0;

}

Output:



19. \*Function Returning a Reference:\*

Write a function that returns a reference to a static variable. Modify the returned value in the main function and print it.

#include <iostream>

using namespace std;

int& staticVariable() { // Function that returns a reference to a static variable

static int num = 5; // Static variable initialized to 5

return num; // Returning reference to the static variable

}

int main() {

int& ref = staticVariable(); // Calling the function and getting a reference to the static variable

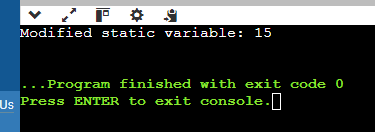
ref += 10; // Modifying the static variable through its reference

cout << "Modified static variable: " << ref << endl;

return 0;

}

Output:



20. \*Passing Objects by Value and Reference:\*

Write a class Employee with attributes name and salary. Write two functions: one that takes an Employee object by value and another that takes an Employee object by reference. Modify the salary in both functions and demonstrate the difference in the main program.