Lab Assignment: Pointers, Dynamic Memory Management, and Virtual Functions

1. The Address-of Operator (&)

Write a C++ program to demonstrate the use of the address-of operator (&).

The program should:

* Declare an integer variable and a pointer to the integer.
* Use the address-of operator to store the address of the integer variable into the pointer.
* Print the value of the integer, the address of the integer, and the value of the pointer.

#include <iostream>

using namespace std;

int main() {

int num = 50;

int\* ptr;

ptr = &num;

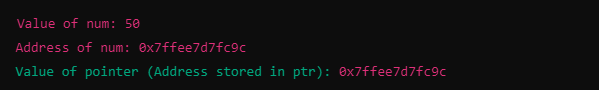
cout << "Value of num: " << num << endl;

cout << "Address of num: " << &num << endl;

cout << "Value of pointer (Address stored in ptr): " << ptr << endl;

return 0 ;

}



2. Pointer Basics

Write a program that:

* Declares a pointer to a float and assigns it a value using dynamic memory allocation.
* Use the pointer to assign a value and display the value using both the pointer and dereferencing it.
* Free the dynamically allocated memory using delete.

#include <iostream>

using namespace std;

int main() {

float\* ptr = new float;

\*ptr = 9.5f;

cout << "Value at the address stored in ptr: " << \*ptr << endl;

delete ptr;

ptr = nullptr;

return 0;

}



3. Dynamic Memory Allocation: new and delete

Write a C++ program that:

* Dynamically allocates memory for an array of integers using the new operator.
* Initialise the array with random values or user input.
* Print the contents of the array using pointer arithmetic.
* Release the dynamically allocated memory using the delete []operator.

#include<bits/stdc++.h>

using namespace std;

int main() {

int n;

cout << "Enter the number of elements: ";

cin >> n;

int\* arr = new int[n];

srand(time(0));

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

arr[i] = rand() % 100;

}

cout << "Array contents: ";

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

cout << \*(arr + i) << " ";

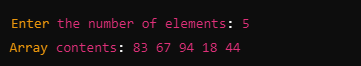
}

cout << endl;

delete[] arr;

return 0;

}



4. Pointers to Objects

Write a program to create a class Car with the following members:

Data members: brand, year.

Member function: display() to show the details of the car.

The program should:

Dynamically allocate memory for a Car object using the new operator.

Assign values to the data members.

Use a pointer to access the display() function and print the car's details.

Finally, release the allocated memory.

#include<bits/stdc++.h>

using namespace std;

class Car {

public:

string brand;

int year;

void display() {

cout << "Car Brand: " << brand << endl;

cout << "Year of Manufacture: " << year << endl;

}

};

int main() {

Car\* carPtr = new Car;

carPtr->brand = "BMW";

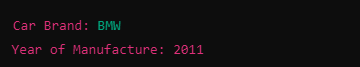
carPtr->year = 2011;

carPtr->display();

delete carPtr;

return 0;

}



5. Virtual Functions and Pointers

Create a base class Shape with a virtual function area() and two derived classes:

Rectangle with width and height attributes.

Circle with radius attribute.

The program should:

Dynamically allocate memory for both Rectangle and Circle objects

using pointers.

Override the area() function in the derived classes to calculate and display

the area.

Use a base class pointer to demonstrate the behavior of the virtual

function.

Free the memory after usage.

#include <bits/stdc++.h>

class Shape {

public:

virtual double area() const = 0;

virtual ~Shape() {}

};

class Rectangle : public Shape {

private:

double width, height;

public:

Rectangle(double w, double h) : width(w), height(h) {}

double area() const {

return width \* height;

}

~Rectangle() {}

};

class Circle : public Shape {

private:

double radius;

public:

Circle(double r) : radius(r) {}

double area() const {

return M\_PI \* radius \* radius;

}

~Circle() {}

};

int main() {

Shape\* rectangle = new Rectangle(5.0, 7.0);

Shape\* circle = new Circle(3.0);

std::cout << "Area of Rectangle: " << rectangle->area() << std::endl;

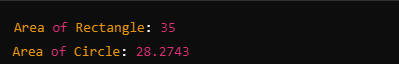
std::cout << "Area of Circle: " << circle->area() << std::endl;

delete rectangle;

delete circle;

return 0;

}



6. Assignment and Copy Initialization

Write a C++ program to demonstrate assignment and copy initialization:

Create a class Point that stores x and y coordinates.

Write a constructor to initialize the Point object, a copy constructor, and

an assignment operator.

Create a Point object using direct initialization.

Create another object using copy initialization.

Assign one object to another using the assignment operator.

Print the details of each object to verify the correct execution of copy

initialization and assignment.

#include <iostream>

using namespace std;

class Point {

private:

int x, y;

public:

Point(int xCoord = 0, int yCoord = 0) : x(xCoord), y(yCoord) {}

Point(const Point& other) : x(other.x), y(other.y) {

cout << "Copy Constructor called\n";

}

Point& operator=(const Point& other) {

if (this == &other) {

return \*this;

}

x = other.x;

y = other.y;

cout << "Assignment Operator called\n";

return \*this;

}

void print() const {

cout << "Point(" << x << ", " << y << ")\n";

}

};

int main() {

Point P1(5, 10);

cout << "P1 details: ";

P1.print();

Point P2 = P1;

cout << "P2 details: ";

P2.print();

Point P3;

P3 = P2;

cout << "P3 details: ";

P3.print();

return 0;

}

