**Department of Computing**

**School of Electrical Engineering and Computer Science**

**CS250 – Data Structures and Algorithms**



**Lab 1: Pointers in C++**

**Submission Details**

|  |  |
| --- | --- |
| Name | CMS ID |
| Muhammad Umer | 345834 |
| Group | GP – 1 |
| Lab Engineer | Anum Asif |
| Faculty Member | Bostan Khan |
| Class | BEE12 |
| Date | 29/01/2024 |
| Time | 10:00 am – 12:50 pm |

**Table of Contents**

[2 Pointers in C++ 3](#_Toc158632072)

[2.1 Introduction 3](#_Toc158632073)

[2.2 Objectives 3](#_Toc158632074)

[2.3 Tools/Software Requirement 3](#_Toc158632075)

[2.4 Description 3](#_Toc158632076)

[2.5 Deliverables 3](#_Toc158632077)

[3 Lab Tasks 4](#_Toc158632078)

[3.1 Task 1 4](#_Toc158632079)

[3.2 Task 2 4](#_Toc158632080)

[3.3 Task 3 5](#_Toc158632081)

[3.4 Task 4 6](#_Toc158632082)

[3.5 Task 5 8](#_Toc158632083)

[4 Conclusion 11](#_Toc158632084)

# Pointers in C++

## Introduction

This lab is about pointers. In C++, a pointer refers to a variable that holds the address of another variable. Like regular variables, pointers have a data type. For example, a pointer of type integer can hold the address of a variable of type integer. A pointer of character type can hold the address of a variable of character type.

## Objectives

This lab will revise the old concepts taught to the students in the previous semesters.

## Tools/Software Requirement

* Visual Studio C++

## Description

Pointers are used to point towards a particular memory address. In this lab we will use the pointers and perform tasks with the help of them.

## Deliverables

Compile a single word document by filling in the solution parts and submit this file on LMS. The name of word document should follow this format. i.e., YourFullName(reg)\_Lab#. You must show the implementation of the tasks in a complete manner to get your work graded.

***Note: Students are required to upload the lab on LMS before deadline.***

# Lab Tasks

## Task 1

Write code to find the memory in bytes occupied by int, long, double, float and char.

Code

#include <iostream>

using *namespace* std;

*int* main()

{

    cout << "Size of int: " << sizeof(*int*) << " bytes" << endl;

    cout << "Size of long: " << sizeof(*long*) << " bytes" << endl;

    cout << "Size of double: " << sizeof(*double*) << " bytes" << endl;

    cout << "Size of float: " << sizeof(*float*) << " bytes" << endl;

    cout << "Size of char: " << sizeof(*char*) << " bytes" << endl;

    return 0;

}

Output

[Running] cd "d:\NUST\Semester 8\Data Structures and Algorithms\Labs\Lab 1\" && g++ task\_1.cpp -o task\_1 && "d:\NUST\Semester 8\Data Structures and Algorithms\Labs\Lab 1\"task\_1

Size of int: 4 bytes

Size of long: 4 bytes

Size of double: 8 bytes

Size of float: 4 bytes

Size of char: 1 bytes

[Done] exited with code=0 in 3.264 seconds

## Task 2

Consider the following program and answer the questions.

void main()

{

int a, \*pa; // Statement 1

pa = &a; // Statement 2

cout<<"pa = &a --> pa = "<<pa<<endl<<endl;

pa = pa + 1; // Statement 3

cout<<"pa = pa + 1 --> pa = "<<pa<<endl<<endl;

pa = pa + 3; // Statement 4

cout<<"pa = pa + 3 --> pa = "<<pa<<endl<<endl;

pa = pa - 1; // Statement 5

cout<<"pa = pa - 1 --> pa = "<<pa<<endl<<endl;

}

Output

[Running] cd "d:\NUST\Semester 8\Data Structures and Algorithms\Labs\Lab 1\" && g++ task\_2.cpp -o task\_2 && "d:\NUST\Semester 8\Data Structures and Algorithms\Labs\Lab 1\"task\_2

pa = &a --> pa = 0xa16e5ff7a4

pa = pa + 1 --> pa = 0xa16e5ff7a8

pa = pa + 3 --> pa = 0xa16e5ff7b4

pa = pa - 1 --> pa = 0xa16e5ff7b0

[Done] exited with code=0 in 0.962 seconds

1. Why does the memory address stored in pointer “pa” vary by 4?

**Answer:** Because the data type of the pointer is int, which is 4 bytes in size.

1. Will the address still vary by 4 if the data type of the above-mentioned code changed from “int” to “long”? Explain your answer.

**Answer:** Yes, because long is also 4 bytes in size.

1. If we try to multiply the address pointed to by “pa” what will happen? Is this logically or programmatically, correct? Attach screen shot of the output you get when you try this multiplication.

**Answer:** The compiler will throw an error because it is not logically correct to multiply an address.

*task2.cpp: In function 'int main()':*

*task2.cpp:47:32: error: invalid operands of types 'int\*' and 'int'*

*to binary 'operator\*'*

*47 |      cout << "pa \* 2 = " << pa \* 2 << endl;*

*|                             ~~ ^ ~*

*|                             |    |*

*|                             int\* int*

## Task 3

Given the following:

int list[5]={3,6,9,12,15};

int \*pArr= list;

Your task is to write a piece of code that prints all values stored in the array list using only pointer variable pArr. Do not use the conventional way of printing values by numbering indexes.

Code

#include <iostream>

using *namespace* std;

*int* main()

{

*int* list[5] = {3, 6, 9, 12, 15}; *// Initialize array*

*int* \*pArr = list; *// Initialize pointer to array*

    for (*int* i = 0; i < 5; i++)

    {

        cout << \*pArr << endl; *// Dereference pointer to print value*

        pArr++; *// Increment to next address*

    }

    return 0;

}

Output

[Running] cd "d:\NUST\Semester 8\Data Structures and Algorithms\Labs\Lab 1\" && g++ task\_3.cpp -o task\_3 && "d:\NUST\Semester 8\Data Structures and Algorithms\Labs\Lab 1\"task\_3

3

6

9

12

15

[Done] exited with code=0 in 1.035 seconds

## Task 4

Write output of the following C++ codes in your document without executing it.

Code a)

int x[4] = {0,4,6,9};  
int \*p, a=3;  
p=x;  
(\*p)++;  
cout<<\*p<<endl;  
cout<<\*(p+1)<<endl;  
p++;  
\*p=\*p+a;  
cout<<\*p<<endl;  
p=p+2; //What is happening here?  
cout<<\*p<<endl;

Comments

*int* main()

{

*int* x[4] = {0, 4, 6, 9};

*int* \*p, a = 3;

    p = x;

    (\*p)++; *// Increment value at address*

    cout << \*p << endl; *// Print value at address*

    cout << \*(p + 1) << endl; *// Print value at next address*

    p++; *// Increment address*

    \*p = \*p + a; *// Add value at address with a*

    cout << \*p << endl;

    p = p + 2; *// What is happening here? Answer: Increment address by 2*

    cout << \*p << endl;

    return 0;

}

Output

1

4

7

9

Code b)

int a, \*p, \*q;

int arr[4]= {0};

p=arr;

q=p;

\*p=4;

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

a=\*p;

p++;

\*p=(a+i);

}

for (int j=0; j<4; j++){

cout<<\*q<<" ";

q++;

}

Comments

*int* main()

{

*int* a, \*p, \*q; *// Initialize a, and pointers p and q*

*int* arr[4] = {0}; *// Initialize array of size 4*

    p = arr; *// Set p to point to first index of array*

    q = p; *// Set q to point to same address as p*

    \*p = 4; *// Set value of zeroth index to 4*

    for (*int* i = 0; i < 3; i++)

    {

        a = \*p; *// Set a to value at address, (4, 4, 5)*

        p++; *// Increment to next address*

        \*p = (a + i); *// Set value at address to a + i, (4 + 0, 4 + 1, 5 + 2)*

    }

*// Final array: {4, 4, 5, 7}*

*// Print array*

    for (*int* j = 0; j < 4; j++)

    {

        cout << \*q << " "; *// Print value with space*

        q++;

    }

    return 0;

}

Output

4 4 5 7

## Task 5

Given the following:

int a=5, b=10;

int \*pa=&a; //pa and pb are pointer variables of type int.

int \*pb=&b;

int \*\*ppa=&pa; //ppa and ppb are called double pointers or pointers-to-pointers.

int \*\*ppb=&pb;

1. Write code of a function that swaps values of variables a and b. Input to the function should be the address of both the variables.

Code

#include <string>

#include <iostream>

using *namespace* std;

*void* swap(*int* *\*a*, *int* *\*b*); *// Function prototype*

*int* main()

{

*int* a = 5, b = 10;

*int* \*pa = &a; *// pa and pb are pointer variables of type int*

*int* \*pb = &b;

*int* \*\*ppa = &pa; *// ppa and ppb are called double pointers or*

*pointers-to-pointers*

*int* \*\*ppb = &pb;

*// a) Swap values of variables a and b*

    cout << "Before swap: " << endl;

    cout << "a: " << a << " b: " << b << endl;

    swap(&a, &b); *// Arguments are addresses of a and b*

    cout << "\nAfter swap: " << endl;

    cout << "a: " << a << " b: " << b << endl;

    return 0;

}

*// Function definition*

*void* swap(*int* *\*a*, *int* *\*b*)

{

*int* temp = \**a*; *// Store value of a in temp*

    \**a* = \**b*; *// Set value of a to value of b*

    \**b* = temp; *// Set value of b to temp*

}

Output

[Running] cd "d:\NUST\Semester 8\Data Structures and Algorithms\Labs\Lab 1\" && g++ task\_5.cpp -o task\_5 && "d:\NUST\Semester 8\Data Structures and Algorithms\Labs\Lab 1\"task\_5

Before swap:

a: 5 b: 10

After swap:

a: 10 b: 5

[Done] exited with code=0 in 0.99 seconds

1. Write code of a function that swaps values of the variables a and b using pointer-to-pointer variables ppa and ppb.

*The same function as before can be used, however, we should dereference the double pointers to get the address of the variables ‘a’ and ‘b’.*

Code

#include <string>

#include <iostream>

using *namespace* std;

*void* swap(*int* *\*a*, *int* *\*b*); *// Function prototype*

*int* main()

{

*int* a = 5, b = 10;

*int* \*pa = &a; *// pa and pb are pointer variables of type int*

*int* \*pb = &b;

*int* \*\*ppa = &pa; *// ppa and ppb are called double pointers or pointers*

*-to-pointers*

*int* \*\*ppb = &pb;

*// b) Swap values of variables a and b using pointer-to-pointer*

*variables ppa and ppb*

    cout << "\nBefore swap: " << endl;

    cout << "a: " << a << " b: " << b << endl;

*// Swaps the values of the addresses stored in ppa and ppb*

*// effectively swapping the values of a and b*

    swap(\*ppa, \*ppb); *// Arguments are addresses of pa and pb*

    cout << "\nAfter swap: " << endl;

    cout << "a: " << a << " b: " << b << endl;

    return 0;

}

*// Function definition*

*void* swap(*int* *\*a*, *int* *\*b*)

{

*int* temp = \**a*; *// Store value of a in temp*

    \**a* = \**b*; *// Set value of a to value of b*

    \**b* = temp; *// Set value of b to temp*

}

Output

[Running] cd "d:\NUST\Semester 8\Data Structures and Algorithms\Labs\Lab 1\" && g++ task\_5.cpp -o task\_5 && "d:\NUST\Semester 8\Data Structures and Algorithms\Labs\Lab 1\"task\_5

Before swap:

a: 5 b: 10

After swap:

a: 10 b: 5

[Done] exited with code=0 in 1.014 seconds

# Conclusion

In this lab, we revised our concepts of pointers in C++. Through hands-on experimentation, we explored the fundamental concept that a pointer serves as a reference to the memory address of another variable, wielding the ability to point to data of specific types. Our focus on integer and character type pointers allowed us to comprehend the versatility and precision that pointers bring to programming. By manipulating addresses and values, we gained a deeper understanding of memory management and the efficient utilization of pointers in C++.