|  |
| --- |
| **Programming Fundamentals** |
| **(CL214)** |
| **LABORATORY MANUAL** |
| **Spring 2021** |
| **C:\Users\Aamer\Desktop\nu-new.png**  **LAB 05** |
| ***Pointer Basics*** |
| **Engr. Ibrar Khan**  **MUHAMMAD ADEEL AKHTAR Engr. Sana Saleh** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | \_\_20I-1025\_\_\_\_\_\_\_\_ | | | B\_\_\_ |
| STUDENT NAME | | ROLL NO | | | SEC |
|  | | | | | |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | |
| LAB ENGINEER SIGNATURE & DATE | | | | | |
| **MARKS AWARDED:** \_\_\_**/10** | | | | | |
|  | | | | | |
| **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES (NUCES), ISLAMABAD** | | | | | |
|  | | | | | |
| Last Edited by: | Engr. Sana Saleh | | Version: | 2.00 | |
| Prepared by: | Engr. Sana Saleh | | Date: | 19 Sep, 2017 | |
| Verified by: | Dr. Shahid Qureshi | | Date: | 24 Feb, 2019 | |

|  |  |
| --- | --- |
| **LAB 05** | **Pointer Basics** |

**Lab Objectives:**

1. To learn about pointers.
2. To learn the difference and similarity between pointers and arrays.
3. To learn about different types of Pointer.
4. Learn the method of passing pointers to functions.

**Software Required:**

* Dev C++

**Introduction:**

1. **Pointers**

A pointer is a variable that stores address of another variable. This pointer essentially points to that variable. Pointer can be used to access and change the value of variable it points to, by using the dereference operator (\*).

# include<iostream>

using namespace std;

int main ()

{

int a=3; // A simple int .

int\* ap; // A pointer of type int .

ap = &a; // Pointer ap now points to int a.

\*ap = 5; // Changes value of a to 5 from 3.

cout<< a <<endl ; // Print value of a.

cout<<&a <<endl ; // Print address of a.

cout<<ap<<endl ; // Print value of ap , which is address of a.

cout<< \*ap<<endl ; // Print value of a using pointer ap.

return 0;

}

## Using Pointers for Referencing

Since pointers point to original variables they can be used to change the value of original variables in a function call.

# include<iostream>

using namespace std;

int Area (int \*, int \*); // Function prototype.

int main ()

{

int a, b, c;

a = 3;

b = 5;

c = Area (&a, &b); // Function call by address.

cout<<"Area of "<< a <<"\*"<< b <<" is: "<< c <<endl;

return 0;

}

// Function definition .

int Area (int \* x, int \* y)

{

int z = (\*x)\*(\* y); // Dereferencing pointers .

\*x=7; // Changes value of a in main ().

return z;

}

## Pointers and Arrays

The name of an array is the pointer to its first element. A subscript besides array name can be translated in English as, “access element located this much distance from start." Keep in mind that Array[i] is the same as \*(Array+i) which means that you only need array pointer to access array elements using the subscript. Consider the following code:

# include<iostream>

using namespace std;

int main ()

{ char Array [6] = {'3' ,'7' ,'2' ,'1' ,'5'}; // 5 characters and 1 NULL

int number[2][2]={{1,2},{3,4}};

cout<< Array <<endl ; // Prints the complete array

// Using subscript .

cout<< Array [0] <<endl ; cout<< Array [3] <<endl ;

cout<< Array [5] <<endl ; cout<< Array [6] <<endl ;

// Using Pointers .

cout<< \*(Array +0) <<endl ; // Prints the value of first element which is 3.

cout<< \*(Array +1) <<endl ; // Prints the value of second element which is 7.

cout<< \*(Array +5) <<endl ; // Prints the value of fifth element which is NULL

cout<< \*(Array +6) <<endl ; // Prints the value of sixth element which is out of bound or ERROR .

return 0;}

## Pass By Pointers:

C++ allows to pass a pointer to a function. Then the parameter has to be declared as a pointer type. Following a simple example where we pass an unsigned long pointer to a function and change the value inside the function which reflects back in the calling function

|  |
| --- |
| #include <iostream>  using namespace std;  void getdata(unsigned long \*par);  int main () {  unsigned long sec;  getdata( &sec );  cout << "Number of seconds :" << sec << endl;  //OR SECOND METHOD  unsigned long \*sec1;  unsigned long a;  sec1=&a;  getdata(sec1);  cout << "Number of seconds :" << \*sec1 << endl;  return 0;  }  void getdata(unsigned long \*par) {  // get the current number of seconds  \*par = 123;  return;  } |

To pass array in pointer notation to function, see following example:

|  |
| --- |
| #include <iostream>  using namespace std;    // function declaration:  double getAverage(int \*arr, int size);  int main () {  // an int array with 5 elements.  int balance[5] = {1000, 2, 3, 17, 50};  double avg;  // pass pointer to the array as an argument.  avg = getAverage( balance, 5 ) ;    // output the returned value  cout << "Average value is: " << avg << endl;    return 0;  }  double getAverage(int \*arr, int size)  {  int i, sum = 0;  double avg;  for (i = 0; i < size; ++i)  {  sum += arr[i];  }  avg = double(sum) / size;  return avg;  } |

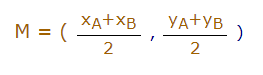
## Practice Problem

1. Dry Run this code:

|  |  |
| --- | --- |
| CODE  int main  { int set[5] = { 15,30,45,60,75};  int w=10; char \*ch; float \*fl; double \*db;  int \*ptr = &w;  void \*ptr1,\*ptr2,\*ptr3,\*ptr4;  int i=1;  char c=’a’;  float f=3.12;  double d=3.1234;  db=&d; fl=&f;  int \*ptr= set;  cout<<\*ptr<<endl;    for(int q=3;q<5;q++)  {  cout<<\*ptr++<<endl;  }  cout<<\*(ptr+2)<<endl;  cout<<\*(ptr+3)<<endl;  cout<<db<<”\t” <<f1<<endl;  ptr1=&i;  ptr2=&c;  ptr3=&f;  ptr4=&d;  cout<<\*ptr1<<\*ptr2<<f<<d;  } | OUTPUT  15  15  30  45  60  0x9ffdd8 0x9ffde0  1a3.123.1234 |

## \*Implement all tasks using pointer notation.

1. Write a program that can find the distance between two points in 2D plane. Ask user to enter coordinate of 3 points. One array store x-axis coordinate of all these points and second array store y-axis coordinate. Using this data, calculate distance between points. Also find the midpoint of line segment joining these points.
   * + void Input\_data( int \*arr1,int \*arr2) //x-coordinate and y-coordinate
     + void Output­\_data(int \*arr1)
     + void Distance(int \*arr1,int\* arr2)
     + void Midpoint(int\* arr1,int \*arr2)

Formula:



#include<iostream>

#include <cmath>

using namespace std;

void Input\_data( int \*arr1,int \*arr2); //x-coordinate and y-coordinate

void Output\_data(int \*arr1,int\* arr2);

void Distance(int \*arr1,int\* arr2);

void Midpoint(int\* arr1,int \*arr2) ;

int main()

{

int arr1[2],arr2[2];

Input\_data(arr1,arr2);

Output\_data(arr1,arr2);

Distance(arr1,arr2);

Midpoint(arr1,arr2);

}

void Input\_data( int \*arr1,int \*arr2)

{

cout<<"enter first point x axis"<<" ";

cin>>\*arr1;

cout<<"enter second point x axis"<<" ";

cin>>\*arr2;

cout<<"enter first point y axis"<<" ";

cin>>\*(arr1+1);

cout<<"enter second point y axis"<<" ";

cin>>\*(arr2+1);

}

void Output\_data(int \*arr1,int\* arr2)

{

cout<<"first point x and y coordinates are"<<\*(arr1+0)<<" "<<\*(arr1+1)<<endl;

cout<<"second point x and y coordinates are"<<\*(arr2+0)<<" "<<\*(arr2+1)<<endl;

}

void Distance(int \*arr1,int\* arr2)

{

int a,b,k,z,ans;

a=\*(arr2+0)-\*(arr1+0);

a\*=a;

b=\*(arr2+1)-\*(arr1+1);

b\*=b;

ans=sqrt((a+b));

cout<<"DISTANCE IS"<<" "<<ans<<endl;

}

void Midpoint(int\* arr1,int \*arr2)

{

float z;

z=(\*(arr1+0)+\*(arr2+0))/2;

float k;

k=(\*(arr1+1)+\*(arr2+1))/2;

cout<<"MID POINTS ARE"<<endl;

cout<<z<<" "<<k<<endl;

}

1. Write a program to print all the LEADERS in the array. An element is leader if it is greater than all the elements to its right side. And the rightmost element is always a leader. For example: in the array {16, 17, 4, 3, 5, 2}, leaders are 17, 5 and 2.
   * + void input(int \*arr)
     + void LEARDERS(int \*arr)

#include<iostream>

using namespace std;

void input(int \*arr);

void LEADERS(int \*arr);

int main()

{

int arr[6];

input(arr);

LEADERS(arr);

}

void input(int \*arr)

{

cout<<"enter name of 6 elements"<<endl;

for(int i=0;i<6;i++)

{

cin>>\*(arr+i);

}

}

void LEADERS(int \*arr)

{

cout<<"leaders are "<<endl;

for(int i=0;i<6;i++)

{

if(\*(arr+i)>\*(arr+i+1))

{

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

}

}

}