# Lab Tutorial for Week 1 Session 2: Pointers and Array-based List

## Contents

1. Tutorial 1a: Pointer Basic
2. Tutorial 1b: Passing pointers to functions
3. Tutorial 1c: Array-based List

## Tutorial 1a: Pointer Basic in C++

### Learning outcomes:

1. Understand pointer data types
2. Create pointer variables
3. Use pointer variables by dereference operator
4. Using new and delete operator

### Tasks:

1. Create a new file (CTRL-N).
2. Solve the following programming problem.
   1. Create an integer variable and assign 99 to it
   2. Print out the content and the address of the variable
   3. Create a pointer variable and assign the address of the variable defined in step 1 to it
   4. Print out the value of the pointer variable and the value stored in the address it is currently pointing to.
   5. Change the value stored in the address it is currently pointing to 115
   6. Print out the value of the pointer variable and the value stored in the address it is currently pointing to.
   7. Create another pointer variable
   8. Use new keyword to create a dynamic storage for the pointer variable
   9. Assign 333 to this storage
   10. Print the the value stored in the address that this pointer is currently pointing to
3. You can start from the below code:

#include <iostream>

using namespace std;

int main(){

int x = 99;

cout << "The content of x is " << x << endl;

cout << "The address of x is " << &x << endl;

// your code here

}

1. Complete the program.
2. Save the source file, compile and run it.

### Questions:

1. What new C++ programming keywords that you learned from this exercise?
2. What is the content of a pointer variable?
3. What is the syntax to get an address of a variable?
4. What new keyword for? Why and when do we need it?
5. Why the operator \* is confusing? Tell the difference between int\* p and \*p!

### Summary:

In this tutorial, you have learned how to create pointer data type and pointer variables. You also have learned how to use pointers, new and delete keywords to create dynamic memory. .

## Tutorial 1b: Passing Pointers to Functions

### Learning outcomes:

1. Understand the difference between pass by value and pass by pointers
2. Create a function that passes its arguments by value
3. Create a function that passes its arguments by pointers

### Tasks:

1. Create a new file (CTRL-N).
2. Solve the following programming problem.
   1. Create a function swap that has two integers as parameters
   2. Implement the body of the function
   3. Call the function from the main program
3. You can start from the below code:

#include <iostream>

using namespace std;

void swap(int x, int y);

int main(){

int a, b;

cout << "Enter first number " << endl;

cin >> a;

cout << "Enter second number " << endl;

cin >> b;

// print before swap

cout << "BEFORE SWAP: First number is " << a << " and the Second number is " << b << endl;

// try to swap

swap(a, b);

// print before swap

cout << endl << "AFTER SWAP: First number is " << a << " and the Second number is " << b << endl;

}

void swap(int a, int b){

// your code here

}

1. Complete the program.
2. Save the source file, compile and run it.

### Questions:

1. Were the two values exchanged within the swap function?
2. Were the two values exchanged in the main program?
3. Explain what did happen?
4. Solve the following programming problem.
   1. Create a function swap that has two integer pointers as parameters
   2. Implement the body of the function
   3. Call the function from the main program
5. You can start from the below code:

#include <iostream>

using namespace std;

void swap(int\* x, int\* y);

int main(){

int a, b;

cout << "Enter first number " << endl;

cin >> a;

cout << "Enter second number " << endl;

cin >> b;

// print before swap

cout << "BEFORE SWAP: First number is " << a << " and the Second number is " << b << endl;

// try to swap

swap(&a, &b);

// print before swap

cout << endl << "AFTER SWAP: First number is " << a << " and the Second number is " << b << endl;

}

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

// your code here

}

1. Complete the program.
2. Save the source file, compile and run it.

### Questions:

1. Were the two values exchanged within the swap function?
2. Were the two values exchanged in the main program?
3. Explain what did happen?

### Summary:

In this tutorial, you have learned how to create functions that have pass-by-value parameters and pass-by -pointers parameters.

## Tutorial 1c: Array-based List

### Learning outcomes:

1. Use the Array-based List

### Tasks:

1. Open the project (File, Open or CTRL-O, Select the " ArraybasedList.dev". The project contains two files: "arrayList.h" and "testListApp.cpp"
2. Open the "arrayList.h" and try to understand the statements in it. This kind of collection of functions is called a MODULE. A module makes software development easier since the programmer can simply import (include) the module in their new program and simply call the relevant functions to implement the needed functionalities. Programmers don’t need to write all the functions from scratch, but instead simply used them.

|  |  |
| --- | --- |
|  | #include <iostream>  // global variable  int \*list; //array to hold the list elements  // functions to insert data  void insert(int insertItem);  void insertAt(int location, const int insertItem);  // functions to remove data  void remove(int removeItem);  void removeAt(int location);  void clearList();  // functions to retrieve data  int retrieveAt(int location);  // functions to replace data  void replaceAt(int location, int repItem);  // helper functions  int seqSearch(int item);  bool isEmpty();  bool isFull();  int listSize();  // function to print data  void print(); |

1. Use the functions in the "arrayList.h to create program testProg.cpp " with the following functionalities:
   1. INSERT new data
   2. REPLACE existing data with a new one
   3. DELETE existing data
   4. PRINT data
2. Compile your program and run it.

### Questions:

1. What is a global variable? How does it differ from a local variable and function parameter?
2. How to create an array using dynamic memory?

---end of Tutorial Week 1---