FEU INSTITUTE OF TECHNOLOGY

COLLEGE OF COMPUTER STUDIES

# CCS007L
# (COMPUTER PROGRAMMING 2)

## EXERCISE

# 5

## POINTERS

| | | |
|---|---|---|
| **Student Name / Group Name:** | | |
| **Members (if Group):** | **Name** | **Role** |
| | | |
| | | |
| | | |
| **Section:** | | |
| **Professor:** | | |

## I. PROGRAM OUTCOME/S (PO) ADDRESSED BY THE LABORATORY EXERCISE

- Design, implement and evaluate computer-based systems or applications to meet desired needs and requirements. [PO: C]

## II. COURSE LEARNING OUTCOME/S (CLO)ADDRESSED BY THE LABORATORY EXERCISE

- Apply the fundamental principles of handling CString values, pointers and memory allocation, and structures using C++in solving computing activities [CLO: 2]

## III. INTENDED LEARNING OUTCOME/S (ILO) OF THE LABORATORY EXERCISE

At the end of this exercise, students must be able to:

- Create a program that will traverse pointers on a given array and do some process
- Create a program that simulates some predefined C-String functions using pointers.

## IV. BACKGROUND INFORMATION

A **pointer** is a variable whose value is the address of another variable. Like any variable or constant, you must declare a pointer before you can work with it. The general form of a pointer variable declaration is:

```
Type *var-name;
```

Here, type is the pointer's base type; it must be a valid C++ type and var-name is the name of the pointer variable. The asterisk you used to declare a pointer is the same asterisk that you use for multiplication. However, in this statement the asterisk is being used to designate a variable as a pointer. Following are the valid pointer declaration:

```
int    *ip;    // pointer to an integer
```

```
double *dp;    // pointer to a double
float  *fp;    // pointer to a float
char   *ch     // pointer to character
```

The actual data type of the value of all pointers, whether integer, float, character, or otherwise, is the same, a long hexadecimal number that represents a memory address. The only difference between pointers of different data types is the data type of the variable or constant that the pointer points to.

Simply, a **pointer** is a variable that **stores the memory address as its value**.

A pointer variable points to a data type of the same type, and is created with the * operator. The address of the variable you're working with is assigned to the pointer:

```
#include <iostream>
#include <string>
using namespace std;

int main() {
  string food = "Pizza";  // A string variable
  string* ptr = &food;  // A pointer variable that stores the address of food

  // Output the value of food
  cout << food << "\n";

  // Output the memory address of food
  cout << &food << "\n";

  // Output the memory address of food with the pointer
  cout << ptr << "\n";
  return 0;
}
```

```
Pizza
0x6dfed4
0x6dfed4
```

Create a pointer variable with the name ptr, that **points to** a string variable, by using the asterisk sign * (string* ptr). Note that the type of the pointer has to match the type of the variable you're working with. Use the & operator to store the memory address of the variable called food, and assign it to the pointer. Now, ptr holds the value of food's memory address.

## V. GRADING SYSTEM/ RUBRIC

| Trait | (Excellent) | (Good) | (Fair) | (Poor) |
|---|---|---|---|---|
| **Requirement Specification(30pts)** | Able to identify correctly all input and output and provide alternative. **(28-20pts)** | Able to identify correctly all input and output **(25-17pts)** | Able to identify only one input or output **(22-14pts)** | Unable to identify any input and output **(20-11pts)** |
| **Data type(20pts)** | Able to apply required data type or data structure and produce correct results **(18-20pts)** | Able to apply required data type or data structure and produce partially correct results **(15-17pts)** | Able to identify required data type or data structure but does apply correctly **(12-14pts)** | Unable to identify required data type **(9-11pts)** |
| **Input Validation(20pts)** | The program works and meets all specifications. Does exception al checking for errors and out-of- range data **(18-20pts)** | The program works and meets all specifications. Does some checking for errors and out of range data **(15-17pts)** | The program produces correct results but does not display correctly Does not check for errors and out of range data **(12-14pts)** | The program produce s incorrect results **(9-11pts)** |
| **Free from syntax, logic, and runtime errors (10pts)** | Unable to run program **(10pts)** | Able to run program but have logic error **(8-9pts)** | Able to run program correctly without any logic error and display inappropriate output **(6-7pts)** | Able to run program correctly without any logic error and display appropriate output **(5pts)** |
| **Delivery (10pts)** | The program was delivered on time **(10pts)** | The program was delivered after 5 minutes from the time required. **(8-9pts)** | The program was delivered after 10 minutes from the time required. **(6-7pts)** | The program was delivered after 15 (or more) minutes from the time required. **(5pts)** |
| **Use of Comments (10pts)** | Specific purpose is noted for each function, control structure, input requirements, and output results. **(10pts)** | Specific purpose is noted for each function and control structure. **(8-9pts)** | Purpose is noted for each function. **(6-7pts)** | No comments included. **(5pts)** |

## VI. LABORATORY ACTIVITY

**INSTRUCTIONS:**
Copy your source codes to be pasted in this document as well as a screen shot of your running output.

**ACTIVITY5.1: strcat function using pointers**

Complete the codes for the stringCat function. It will use the same function which is **strcat** function.

```
#include <iostream>
using namespace std;
void stringCat(char *s1, char *s2);

int main()
{
    char str1[20]="The Happy";
    char str2[20]=" Man";
    stringCat(str1,str2);
    cout << str1;
    system("pause > 0");
    return 0;
}

void stringCat(char *s1, char *s2)
{
    while(   )
            ;
    while(*     = *    );
}
```

## ACTIVITY 5.2: Reverse string

Create a program that will return a reverse string using pointer.

```
#include <iostream>
using namespace std;
char* stringRev(char *s);

int main()
{
    char str[]="Happy Day";
    cout << stringRev(str);
    system("pause > 0");
    return 0;
}

char* stringRev(char *s)
{
    char* tmp;
    tmp = new char;
    int i, cnt(0);
    for(   ;           ;   )
    {
        cout << s[ ] << endl;
        cnt++;
    }
    for(   ;     ;   )
    {
        tmp[ ]=s[   - - ];
    }
    tmp[i]='\0';
    return tmp;
}
```

## VII. QUESTION AND ANSWER

Briefly answer the questions below. Avoid erasures. For group activity, specify the name of GROUP MEMBER/s who answered the question. Do not forget to include the source for all NON-ORIGINAL IDEAS.

| |
|---|
| • How do you use pointers? |
| • How do you create a dynamic array? |

## VIII. REFERENCES


- Zak, Dianne (2016). An Introduction to Programming with C++
- Deitel, Paul & Deitel, Harvey (2012). C++ How To Program, Eighth Edition
- https://www.cprogramming.com/tutorial/lesson6.html