Féidearthachtaí as Cuimse
Infinite Possibilities

Week 9 - Tutorial

OLLSCOIL TEICNEOLAÍOCHTA
BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL
UNIVERSITY DUBLIN

Programming - Week 9

Overview

- Pointers
- Pointer Variables
- Dereferencing
- Mandatory Question – Week 8 Last Lab session

Pointers

- A pointer is a variable that stores the memory address of another variable.
- Instead of storing a specific value (like an integer or character), a pointer holds the location in memory where that value is stored.

Eg.

int age = 17;

Memory Address	Data
0x7ffee5a38587	17

Pointers - Example

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TECHNOLOGICAL UNIVERSITY DUBLIN

example_1.c > ...

```c
#include <stdio.h>
int main()
{
    int var1;
    char var2;
    var1 = 1;
    var2 = 'A';

    printf("var1 contains %d -- memory address %p\n\n", var1, &var1);
    printf("var2 contains %c -- memory address %p\n", var2, &var2);

    return 0;
} // end main()
```

Memory Address	Data
0x7ffee5a38588	1
0x7ffee5a38587	A

```
var1 contains 1 -- memory address 0x7ffee5a38588

var2 contains A -- memory address 0x7ffee5a38587
```

Pointer Variables

- A pointer variable in C is a variable that holds the memory address of another variable
- Rather than storing a direct value (eg. integer or character), it will store the address in memory where a value is located.
- Pointer Variables are Type-Specific. A pointer variable is declared with a specific data type to indicate what kind of variable it points to.

Declaring Pointer Variables

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TECHNOLOGICAL UNIVERSITY DUBLIN

- To declare a pointer variable we use the asterisk (*) before the pointer variable name, along with the data type it will point to.

```
int *ptr;       // Declares a pointer to an integer
char *cptr;     // Declares a pointer to a character
float *fptr;    // Declares a pointer to a float
```

Pointer Variable Example

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TECHNOLOGICAL UNIVERSITY DUBLIN

C example_2.c > ...

```c
#include <stdio.h>

int main() {
    int num = 50;        // Regular int variable
    int *p = &num;       // Pointer variable storing the address of num

    printf("Address of num: %p\n", p);     // Print the address stored in pointer p
    printf("Value of num using pointer: %d\n", *p);  // Dereference p to get the value of num

    *p = 100;   // Change the value of num via pointer p
    printf("New value of num: %d\n", num); // Prints the updated value of num

    return 0;
}
```

```
Address of num: 0x7ffee00e4588
Value of num using pointer: 50
New value of num: 100
```

Dereferencing

- The dereference operator is used to access the contents of a regular variable whose memory address is stored in a pointer variable
  - aka the Indirection operator
- Dereferencing is the operation of accessing or modifying the value located at the address stored in a pointer

How Dereferencing Works

1. **Pointer Declaration:**
   - A pointer is declared to store the address of a variable.
     - Eg: int *ptr declares a pointer to an integer.
2. **Assigning an Address to the Pointer:**
   - Using the address-of operator (&), you can store the address of a variable in a pointer.
     - Eg: ptr = &x; assigns the address of variable x to ptr.
3. **Dereferencing the Pointer:**
   - Using the dereference operator (*), you can access or modify the value at the memory location the pointer holds.
     - Eg: if ptr holds the address of x, then *ptr will access the value of x.

How Dereferencing Works

- Example in C

example_4.c > ...

```c
#include <stdio.h>

int main() {
    int age = 19;           // Declare an integer variable
    int *ptr = &age;        // Declare a pointer to int and assign it the address of age

    printf("Value of age: %d\n", age);    // Output: Value of age: 10
    printf("Address of age: %p\n", &age); // Outputs the address of age in memory
    printf("Address stored in ptr: %p\n", ptr); // Should be the same as address of age
    printf("Value at address stored in ptr: %d\n", *ptr); // Output: 10, which is the value of age

    // Modifying the value of age through the pointer
    *ptr = 20;
    printf("New value of age: %d\n", age);     // Output: New value of age: 20

    return 0;
}
```

Mandatory Question – Prog Lab 6

- Write a program that defines an integer array with 5 elements. Your program must do the following:
- Enter 5 integer values into the array.
- Define another integer array with 5 elements and copy the values from the 1starray into the 2nd array in reverse order (e.g., the value in the first element of the 1st array will be copied into the last element in the 2nd array, etc..).

Mandatory Question – Prog Lab 6

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

Problem Solving – break the problem into smaller pieces

1. Create an array.
2. Reverse the array.
3. Display the original array.
4. Display the reversed array.

Step 1

Create the 2 arrays and get the users 5 numbers

mandatory_lab_6.c > ...

```c
#include <stdio.h>

#define SIZE 5

int main() {
    int arr1[SIZE];
    int arr2[SIZE];

    // Step 1: Enter 5 integer values into arr1
    printf("Enter %d integer values:\n", SIZE);
    for (int i = 0; i < SIZE; i++) {
        printf("Enter value %d: ", i + 1);
        scanf("%d", &arr1[i]);
    }

    return 0;
}
```

Step 2

Reverse the content of array 1 and store in array 2

```
16      // Step 2: Copy values from arr1 into arr2 in reverse order
17      for (int i = 0; i < SIZE; i++) {
18          arr2[i] = arr1[(SIZE-1) - i];
19      }
```

Array 1

Index	0	1	2	3	4
Value	23	45	54	2	19

Array 2

Index	0	1	2	3	4
Value	19	2	54	45	23

We loop starting at zero. We access the last element in arr1 via arr1[SIZE-1] and we store this in the first position in arr2[0]. As I increments we go backwards through arr1 and forwards through arr2

Step 3

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TECHNOLOGICAL UNIVERSITY DUBLIN

Display the arrays

```
21      // Display the results
22      printf("\nOriginal array (arr1): ");
23      for (int i = 0; i < SIZE; i++) {
24          printf("%d ", arr1[i]);
25      }
26
27      printf("\nReversed array (arr2): ");
28      for (int i = 0; i < SIZE; i++) {
29          printf("%d ", arr2[i]);
30      }
31
```

```
Enter 5 integer values:
Enter value 1: 1
Enter value 2: 2
Enter value 3: 3
Enter value 4: 4
Enter value 5: 5

Original array (arr1): 1 2 3 4 5
Reversed array (arr2): 5 4 3 2 1 %
```

Programming Pitfall

1.

```
int a =10;
int b = 2;
int c;

int *ptr1;
int *ptr2;

ptr1 = &a;
ptr2 = &b;

c = a / b;

c =  *ptr1/*ptr2;   // Wrong. This is the start of a multi-line comment
c =  *ptr1 / *ptr2;   // Correct


printf("c contains %d", c);
```

2. If you declare 2 or more pointer variables on the same line, you must write the * character before EACH variable.

e.g.,

```
int *ptr1, *ptr2, *ptr3;  // Correct

int *ptr1, ptr2; // Wrong
```

Additional Code from Tutorial Session

- Problem 1
  - Mandatory Question – Prog Lab 6 (Jonathan's solution coded in-class)
- Problem 2
  - Prog Lab 6 – Q2 - (Jonathan's solution coded in-class)

Problem 1 – in-class code

- Write a program that defines an integer array with 5 elements. Your program must do the following:
- Enter 5 integer values into the array.
- Define another integer array with 5 elements and copy the values from the 1starray into the 2nd array in reverse order (e.g., the value in the first element of the 1st array will be copied into the last element in the 2nd array, etc..).

Problem 1 – in-class code

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TECHNOLOGICAL UNIVERSITY DUBLIN

C in_class_1.c > ...

```c
#include <stdio.h>

#define SIZE 5

int main() {
    int arr1[SIZE];
    int arr2[SIZE];

    printf("Please enter 5 values:\n");

    // get input from from user
    for ( int i = 0; i<SIZE ; i++ ) {
        scanf(" %d", &arr1[i]);
    }

```

Problem 1 – in-class code

```
16      for ( int i = 0; i<SIZE ; i++ ) {
17          printf("arr[%d] is: %d and we copy to arr2[%d]\n", i, arr1[i], SIZE-1-i);
18          arr2[SIZE-1-i] = arr1[i];
19      }
20
21      for ( int i = 0; i<SIZE ; i++ ) {
22          printf("%d - ", arr2[i]);
23      }
24
25      return 0;
26  }
27
```

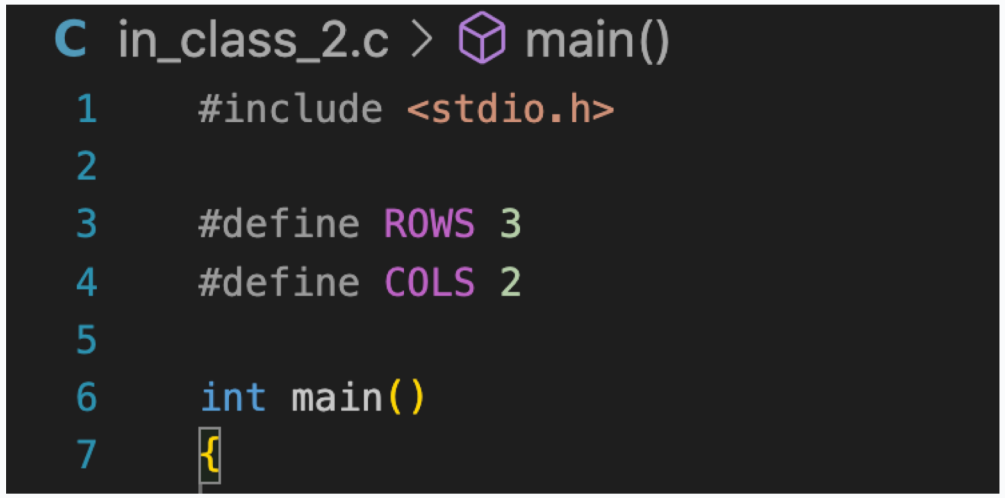
Problem 2 - Prog Lab 6 – Q2

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

2. Write a program that uses a 3x2 (2-D) array. Your program must do the following:

a) Enter values into the array
b) Display the values in the array
c) Find the smallest & largest value and display both to standard output
d) Calculate the average of the values and display to standard output

Problem 2 (in-class code)



- Define ROWS and COLS to reuse this in the codebase for our array definitions.

Problem 2 (in-class code)

```
13
14      // Ask the user to enter data
15      for (int i = 0; i < ROWS; i++)
16      {
17          printf("This is the rows\n");
18          for ( int j = 0; j < COLS ; j++) {
19              printf("Enter the value for arr1[%d][%d]", i,j);
20              scanf(" %d", &arr1[i][j]);
21          }
22      }
23      printf("\n******Data Entry Completed******\n");
24
```

- Use a for loop to get the user to enter data.
- We have a 3x2 array
- Eg.

12	34
54	65
01	24

Problem 2 (in-class code)

```
24
25      // Display the data stored
26      for (int i = 0; i < ROWS; i++)
27      {
28          for ( int j = 0; j < COLS ; j++) {
29              printf("arr1[%d][%d]: %d\n", i,j, arr1[i][j]);
30          }
31      }
```

- Loop through the 2D array and print out the values

Problem 2 (in-class code)

```
33      // find smallest and largest - initially
34      // set largest and smallest to the first num
35      smallest = arr1[0][0];
36      largest = arr1[0][0];
37
38      for (int i = 0; i < ROWS; i++)
39      {
40          for ( int j = 0; j < COLS ; j++) {
41              if (arr1[i][j] > largest) {
42                  largest = arr1[i][j];
43              }
44
45              if (arr1[i][j] < smallest) {
46                  smallest = arr1[i][j];
47              }
48
49              // add all the values in the 2D array
50              // as the loop iterates
51              sum_total = sum_total + arr1[i][j];
52          }
53      }
54
55      printf("Largest: %d\n" ,largest);
56      printf("Smallest: %d\n" ,smallest);
57      printf("Average: %f\n" , sum_total/6);
58
59  }
```

- First we set the smallest and largest values to the first value in the array.
- We will loop through the values in the array and compare them to the largest and smallest int values we are storing. If bigger or smaller we update the largest and smallest int values .
- For the average value we use sum_total to add all the values together. This will be divided by the number of values in the array to get the average.

Questions