CSC180: Lecture 11

Wael Aboulsaadat

wael@cs.toronto.edu

http://portal.utoronto.ca/

Acknowledgement: These slides are partially based on the slides supplied with Prof. Savitch book: Problem Solving with C

Multidimensional Arrays

Multi-Dimensional Arrays

- C allows arrays with multiple index values
  - char page [30] [100];
    declares an array of characters named page
    - page has two index values:
      The first ranges from 0 to 29
      The second ranges from 0 to 99
  - Each index in enclosed in its own brackets

Index Values of page

- The indexed variables for array page are
  page[0][0], page[0][1], …, page[0][99]
  page[1][0], page[1][1], …, page[1][99]
- …
  page[29][0], page[29][1], … , page[29][99]

- page is actually an array of size 30
  - page's base type is an array of 100 characters

Arrays in Functions

Arrays in Functions

- Indexed variables can be arguments to functions
  - Example: If a program contains these declarations:

```
int i, n, a[10];
void my_function(int n) {…..}
```

  - Variables a[0] through a[9] are of type int, making these calls legal:

```
my_function( a[ 0 ] );
my_function( a[ 3 ] );
my_function( a[  i ]  );
```

Array Parameter Declaration

- An array parameter is indicated using empty brackets in the parameter list such as

```
void fill_up(int a[ ], int size)
{
  .......
}
```

Function Calls With Arrays

- If function fill_up is declared in this way:

  void fill_up(int a[ ], int size) {…..}

- and array score is declared this way:

  int score[5], number_of_scores;

- fill_up is called in this way:

  fill_up(score, number_of_scores);

Function Call Details

- A parameter is identified as an array parameter by the [ ]'s with no index expression

```
void fill_up(int a[ ], int size);
```

- An array argument does not use the [ ]'s

```
fill_up(score, number_of_scores);
```

Array Parameters

- An array parameter is a placeholder for the argument
  - When an array is an argument in a function call, an action performed on the array parameter is performed on the array argument
  - The values of the indexed variables can be changed by the function

How does the function know how to access the array elements?

- To access element i, the function uses the formula
  - address in memory of element i =

  start address of array + i * element size
  - Start address of array = address of first element in array
  - E.g.

    score[2] is an indexed variable to the location identified by the above formula

Array Parameter Considerations

- Because a function does not know the size of an array argument…
  - The programmer should include a parameter that specifies the size of the array
  - E.g.

```
fill_up(score, 5);
fill_up(score, 10);
```

const Modifier

- Array parameters allow a function to change the values stored in the array argument
- If a function should not change the values of the array argument, use the modifier const
- An array parameter modified with const is a constant array parameter
  - E.g.:

```
void printArray(const int a[ ], int size)
{
   ……
}
```

Using const With Arrays

- If const is used with an array parameter:
  - The compiler will issue an error if you write code that changes the values stored in the array parameter

Function Calls and const

- If a function with a constant array parameter calls another function using the const array parameter as an argument…
  - The called function must use a constant array parameter as a placeholder for the array
  - The compiler will issue an error if a function is called that does not have a const array parameter to accept the array argument

const Parameters Example

- double compute_average(int a[ ], int size);

```
void show_difference(const int a[ ], int size)
{
      double average = compute_average(a, size);
       …
}
```

- compute_average has no constant array parameter
- This code generates an error message because compute_average could change the array parameter