Data Structures and Program Design in C

Topic 4 : Pointers, Functions and Arrays

Dr Manjusri Wickramasinghe

TAKE
NOTES

Outline

- What is a Pointer?
- What are Functions?
- Arrays

What is a Pointer? …(1)

- A pointer is a variable that is capable of holding the memory address of a given variable.
  - Pointers are declared using the unary * (indirection or differencing operator)

- Address represents a logical memory address.

- Memory has two from
  - **Logical Memory -->** the memory where information is stored as seen by the computer program.
  - **Physical Memory -->** The actual Random Access Memory (RAM) of the computer.

What is a Pointer? …(2)

- Logical memory is visible to the user and is a virtual address generated by the CPU.
- Logical address are what the user utilizes to access the physical memory.
- The set of all logical address generated by the CPU is called the logical address space.

What is a Pointer? …(3)

- A given program sees memory as consecutively numbered cells.
- The cells can be manipulated individually or as contiguous groups.
- The address operator (&) gives the address of an object.
    - It is a unary operator.
    - Not to be confused with the bitwise AND operator.

What is a Pointer? …(4)

- To define a pointer following syntax is used:

```
<data type> *ptr;   //to declare
 ptr = &<object>    //to assign
```

- To retrieve and assign values to a memory location pointed by the pointer following syntax is used:

```
<variable> = *ptr //variable is assigned the value
*ptr =  <value>   // stores the value
```

UCSC

What is a Pointer? …(5)

- Size of a pointer depends on multiple factors such as the operating system (OS) and the CPU architecture (e.g. x86, x64, ARM).
  - On 32 – bit systems the size is 4 bytes.
  - On 64 – bit systems the size is 8 bytes.

- If the size of the pointer is the same for a given system, why do we need to have a data type when defining a pointer?

What is a Pointer? …(6)

- Arithmetic operation can be performed on the dereferenced pointer.
- Since pointer are variables, they can be used without the dereferencing operator such as to assign one pointer to another pointer of the same data type.
- What is the address of a pointer?

What is a Function?

- Functions represent a block of functionality which takes inputs and produces an output.
  - Not all functions take inputs
  - Not all functions produce outputs.
- Functions allow the reuse of logic within the same program.

UCSC

Anatomy of a Function

```
<return_type> <function_name> ([<params>,*])
{
    //function body
}
```

Return Type

- Function can have any return type.
- It can be a primitive data types, user defined data types.
- It is not always required for function to return a value. `void` type is used to indicate a function will be not be returning any value.
- Depending on the context of it is used `void` may mean one of the following
  - No value
  - No type
  - No parameter

Parameters

- Parameters are inputs to the functions
  - It is not mandatory to have parameters for a function.
  - Parameters cannot be optional

- When calling the function real values are passed to the function. Such values are called arguments.

Call by Value and Call by Reference …(1)

- When parameters are passed to the function such parameters are called actual parameters.
- When the actual parameters are received by the function they are called formal parameters.
- This is called “pass by value” and “pass by reference”.

UCSC

Call by Value and Call by Reference ...(2)

- When parameters are passed using the pass by value methods, the actual parameters are copied to the formal parameters.
  - There exists two copies of parameters.
  - Actual variables do not change.
  - Considered a much safer method than pass by reference.

- When parameters are passed using the pass by reference, the address of actual parameters are passed as formal parameters.
  - Actual variables change if they are modified within the function.
  - Pointers are required to perform pass by reference.
  - Preferred when large amounts of data is being passed to the function.

scanf(…) …(1)

- `scanf(…)` stands for scan formatted string.

- Reads the input data from the standard input (`stdin`) and writes the result into the given arguments.

```
int scanf(const char *format, variable_address)
```

scanf(…) …(2)

- scanf(…) return three types of values. These are:
  - Greater than zero (> 0) → The number of values are converted and assigned to the variables.
  - Equal to zero (= 0) → No value has been assigned to the variables.
  - Less than zero (< 0) → Read error of stdin has occurred or and end-of-file (EOF) is reached before the assignment was made.

- The function supports scanset specifiers which are represented by ‘%[]’.
  - The scanf(…) when scansets are defined will process only the characters that form the part of the scansets.

Arrays …(1)

- Arrays are group of elements of the same data type that is stored contiguously in memory.
    - Arrays are also called homogeneous data structure.
    - Arrays are fixed size data structures.

- Arrays can be defined in the following ways:

```
          <data_type> <array_name> [ <size>];
<data_type> <array_name> [ ] = {<value 1>,… <value n>} :
```

Arrays …(2)

- In the C Programming language arrays have zero based indexing.
- Arrays have a fixed size.
- To find the length of the array `sizeof(…)` function can be used.
- Elements of the array can be accessed through pointer arithmetic.

Arrays …(3)

- Arrays can have more than one subscript or an index.

- Multidimensional arrays have multiple subscripts and is declared in the form

```
<data_type> <name>[size][[size]..[size]];
```

- When considering organizing and accessing elements in a multi-dimensional array two methods are available.
    - Row-Major Order
    - Column-Major Order

Arrays ...(4)

- Arrays are used as a base data structure to implement abstract data types such as:
  - Stacks
  - Queues

- When passing arrays to functions two way exist as shown below

```
<return type> function(array_type name[size], …)
   <return type> function(array_type* name, …)
```

Arrays ...(5)

- Properties
  - Arrays are homogeneous.
  - Arrays are contiguous
  - Arrays are fixed in size and **cannot** be resized.
  - Can be stored in row-major order or column-major order.
  - Array is a linear data structure
  - Array is a random access data structure and can also be accessed sequentially.

Questions?