Introduction to Computing

MCS1101B

Lecture 10

By
Soumadip Biswas
Associate Professor, IEM

IEM
श्रद्धावान लभते ज्ञानम्

Recap

- Character arrays
  - String
  - Scanning a string
  - Operations on strings - string.h
- Preprocessors

- User defined datatypes
  - Structures
  - Complex numbers example
  - sizeof structures
  - The typedef keyword

Structure - recall complex numbers

- Example:
  - Complex numbers are of the form x + *i* y
  - x and y can be any real numbers

```
typedef struct complex{
        float x;
        float y;
}Q;

Q n1 = {10.0, 20.0};
Q n2;
Q *ptr;
```

Structures and pointers

- Since structures are just another datatype - it is possible to create pointers of it's type
- struct complex *ptr; ⇒ is able to contain the address of structure variable
  - We could also write Q *ptr; ⇒since we renamed it as Q
- So, sizeof(ptr) ⇒ ?

- How do you access the members using pointers
  - Q *ptr; Q v = {10, 20};
  - ptr = &v;
  - *ptr.x ⇒ will not work
  - You can write (*ptr).x
  - Alternatively ptr->x can be used to access the members using pointers

Structures examples

- Store student record with name, roll number, height, weight, DoB, DoJ
- How do you store information about 100 students?
- What happens if one or more student joins later on?
- What happens if you do not know the number of students beforehand?

- Solutions
  — ideas?
-

Array and Structure

- Since structures are just another datatype - it is possible to create an array
- Q arr[5]; ⇒ is equivalent of 5 Q variables
  - We can access the variables using indexes e.g. arr[1], arr[3], etc.
  - We can also access using pointer arithmetic ← remember this?

- arr[i].x, arr[i].y ← to access member variables
- arr[i] == *(arr + i)
- So (arr+i)->x should work

– but how to create array when size is not known beforehand?

Dynamic Memory allocation (DMA)

- This is another way to allocate memory for variables
- It can allocate memory to a variable during the runtime of the program
  - So, you can read/scan the number of elements from the user
  - Then allocate necessary memory
- It works for allocating memory for
  - A single variable of any type
  - An array of any type

- We need a new include library
  - stdlib.h
- We will use two functions from this library for DMA
  - malloc - **m**emory **alloc**ator
  - free - frees some allocated memory
- Prototype: *void* malloc (int size)*
- It allocates a memory space of the given *size* and returns a pointer(*) (without any specific type, i.e. **void**)
- You can **typecast** it to your need

DMA (contd)

- To create a int variable using malloc, declare a int pointer variable
  - int *ptr;
- Allocate memory using malloc
  - ptr = (int*) malloc(sizeof(int));
- Access the values using *ptr
  - *ptr = 10;
  - printf (“%d”, *ptr); // →prints 10

- **Caution**: if you try to access *ptr before allocating memory, the behaviour is undefined
- So, for the structure Q, we can do the same
  - Q *ptr;
  - ptr = (Q*) malloc (sizeof(Q));
  - Access: ptr->x, ptr->y

Array and DMA

- To create an array using DMA
- We need to specify the total memory size required for the array
- e.g., for an integer array of size 10, we can write the following code
  - int *arr;
  - arr = (int*) malloc (sizeof(int) * 10);
  - Access arr[i] or *(arr+i)

- If you need to take size from the user, you can do the following
  - int n;
  - int *arr;
  - scanf (“%d”, &n);
  - arr = (int*) malloc (sizeof(int) * n);
- To free an allocated memory, you can write
  - free (ptr)
    - Make sure the ptr is a valid one
    - Otherwise, it may result in error

Adding an element in array

- Array has a fixed size
  - Be it allocated using DMA or statically
- Assume you have an array of 10 elements
  - You have inserted 5 elements from 0 to 4 indexes, then you want to insert another element in position 2
  - You have already inserted 10 elements, then you want to add another element

- A better solution
  - Linked list

Next Class…

- Files
- Python preliminaries