CSE101-Lec#25-26-27

- Array and Pointers

Outline

- Pointer to Array
  - Pointer to group of 1D arrays
- Array of pointers.

(lec-25)Pointer to Array

- Array name itself is an address or pointer. It points to the first element(0^{th} element) of array.
- The arrays are accessed by pointers in same way as we access arrays using array name.
- Consider an array `b[5]` and a pointer `bPtr`:
  - `bPtr[3]` is same as `b[3]`

Program to find the mean of array using array name as a pointer.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[] = {3, 7, -1, 4, 6};
int i;
double mean = 0;      //compute mean of values in a
for (i = 0; i < 5; ++i)
{
mean += *(a + i);
}
mean /= 5;
printf("Mean = %.2f\n", mean);
getch();
}
```

Program to find the mean of array using pointer to array.

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[] = {3, 7, -1, 4, 6};
int i;
int *aptr = a;
double mean = 0; //compute mean of values in a
for (i = 0; i < 5; ++i)
{
mean += *(aptr + i);
}
mean /= 5;
printf("Mean = %.2f\n", mean);
getch();
}
```

Practice Session

- Practice various programs of arrays and same program by using pointer to array.
  - Example program:
    - Find sum of squares and sum of cubes of array elements using pointers.
    - To copy elements of one array to another using pointer to array.
    - To find the maximum value out of the array elements.
    - All the operations possible on arrays.

(Lec-26)Pointer to group of 1D Arrays

- 1-D array can be represented in terms of a pointer (array name) and an subscript,
- 2-D array can also be represented with an equivalent pointer notation.
- A 2-D array is actually a collection of 1-D arrays.
- Therefore , **we can define a 2-D array as a pointer to a group of contiguous 1-D arrays.**
- 2 D array declaration can be written as

data-type (*ptrvar) [expression 2];

rather than

data type array[expression 1] [expression 2];

- Eg: `myArray[10][20]` is 2 D array having 10 rows and 20 columns. The item in row 2 and column 5 can be accessed by writing:

  `myArray`[2][5];

  or

  ((`myArray`+2)+5)

- `myArray` is a 2 D array having rows and 20 columns. We can declare x as:

```
int (*myArray)[20];
```

  Rather than

```
int myArray[10][20];
```

- In this first declaration, `myArray` is defined to be a pointer to a group of contiguous 1 –D 20 element integer arrays.
- Thus `myArray` points to the first 20 element array which is actually the first row i.e row 0 of original 2 D array.
- Similarly, (`myArray`+1) points to the second 20 element array and so on.

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
        int *p[5];      //2-D
        int (*q)[5];    //1-D
        printf("%d  %d \n",sizeof(int), sizeof(int *));
        printf("%d  %d \n",sizeof(p), sizeof(p[0]));
        printf("%d  %d \n",sizeof(q), sizeof(*q));
}
```

Output:

4	8
40	8
8	20

int *(p[5]);

int (*q) [5];

8*5=40

4*5=20

Array of pointers

```
#include <stdio.h>
main () {
   int  var[] = {10, 100, 200};
   int i, *ptr[3];   //Array if 3 pointers

   for ( i = 0; i < 3; i++) {
      ptr[i] = &var[i]; /* assign the address of integer. */
   }
   for ( i = 0; i < 3; i++) {
      printf("Value of var[%d] = %d\n", i, *ptr[i] );
   }
}
```

Arrays of Pointers

- We can have arrays of pointers since pointers are variables.
- An array of pointers is a collection of **addresses**.
- A common use of an **array of pointers** is to form an **array of strings**, referred to simply as a **string array**.
- Each entry in the array is a string, but in C a string is essentially a pointer to its first character.
- So each entry in an array of strings is actually a pointer to the first character of a string.

Arrays of Pointers

The suit[4] portion of the definition indicates an array of 4 elements.

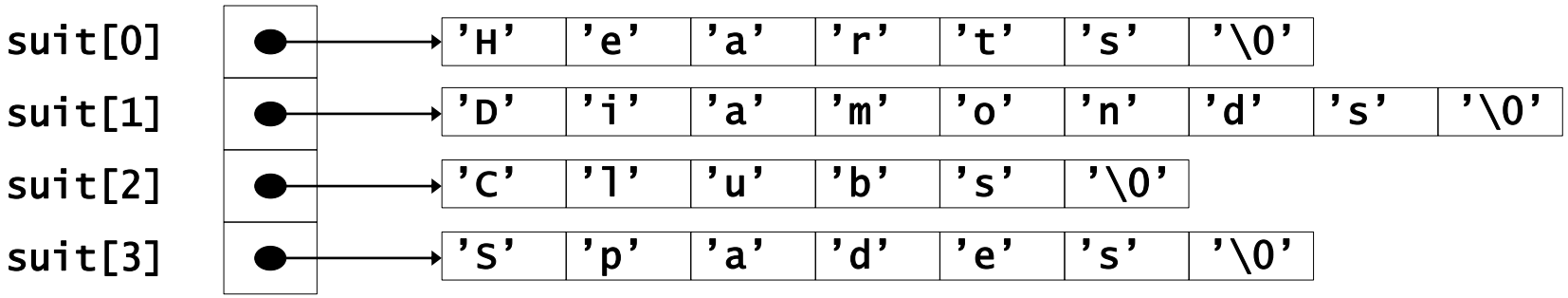
- **Example:**

```
const char *suit[ 4 ] = {"Hearts","Diamonds",
    "Clubs", "Spades" };
```

The char * portion of the declaration indicates that each element of array suit is of type “pointer to char.”

Arrays of Pointers

- `char *` indicates that each element of `suit` is a "pointer to a `char`"
- The strings are not actually stored in the array `suit`, only pointers to the strings are stored



`suit` array has a fixed size, but strings can be of any size

Arrays of Pointers

- The four strings are 7, 9, 6 and 7 characters long, respectively.
- Although it appears as though these strings are being placed in the suit array, only pointers are actually stored in the array
- Each pointer points to the first character of its corresponding string.
- Thus, even though the suit array is fixed in size, it provides access to character strings of any length.

Program to show array of pointers.

```
#include<stdio.h>
void main()
{
 int i;
 char *suit[4]={"spades","hearts","clubs“,
“diamonds”};
 printf(“The suit of cards have:”);
 for(i=0; i<4; i++){
   printf("%s\n",suit[i]);
 }
}
```

The suit of cards have:
spades
hearts
clubs
diamonds

- However pointers and arrays are different:
- A pointer is a variable. We can do

  pa = a and pa++.
- An Array **is not** a variable.

  a = pa and a++ **ARE ILLEGAL**

Next Class: Dynamic Memory Management

cse101@lpu.co.in