ESc101: Pointers and Arrays

Instructor: Krithika Venkataramani

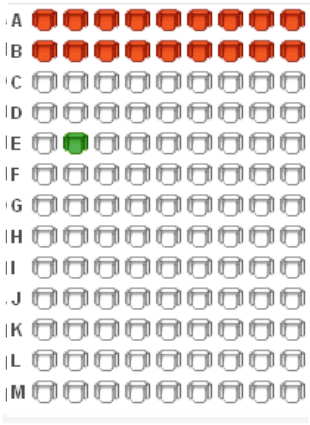
Semester 2, 2011-2012

1

The content of some of these slides are from the lecture slides of Prof. Arnab Bhattacharya

2

Movie Theater Seat Allocation Vs. Variable Allocation



Ticket 1 – E2;

Ticket 1: 'Aditya';

- Memory size ~ theater seat capacity
- variable declaration ~ seat number allocation
- variable address ~ seat number
- variable value ~ name of person in that seat

3

Pointers to variables

Movie Theater Seat Allocation

- Buy Ticket 1
- Seating agent seats the person in the seat E2
- Ticket 1 Seat number = E2
- name of person having Ticket 1 is 'Aditya'
- 'Bharat' is asked to sit in E2
- person with Ticket 1 is now 'Bharat'

Variable Allocation

- char A;
- scanf("%c", &A); /*the input character is stored in address 5*/
- char *pc; pc=&A; //pc=5
- Value of A is 'G';
- *pc = 'V';
- /* A = 'V' now */

4

Swapping contents of pointers in functions

Function: pointer content swap

```
void swap(int *pa, int *pb)
{
int temp;
temp = *pb;
*pb = *pa;
*pa = temp;
return;
}
```

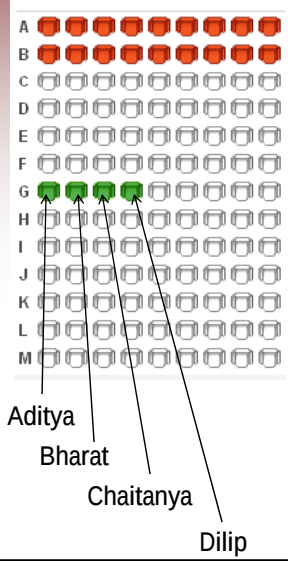
- printf(‘‘a=%d b=%d’’, a, b);
- swap (&a, &b);
- printf(‘‘a=%d b=%d’’, a, b); /*a and b would now be swapped*/

Theater: swap persons in seats

- swapping the persons seated in seats of Ticket 1 and Ticket 2
- E.g. ‘Aditya’ in seat E2 having Ticket 1 is swapped with ‘Bharat’ in seat D5 having Ticket 2.
- Person having Ticket 1 is now ‘Bharat’. Person having Ticket 2 is ‘Aditya’.

5

Theater Seat Allocation Vs. Array Allocation



- Buy 4 tickets for ‘Music’ group
- G1-G4 allotted to ‘Music’ group
- Seat 4 persons from G1 to G4
- size of ‘Music’ group = 4
- ‘Music’ group starts at G1 and starting ‘Music’ group member is ‘Aditya’

- int Music[4];
- Music allotted 4 integer spaces from address 5000
- for (i=0; i<4; i++)
  scanf(“%d”,&Music[i]);

//Input: 6 5 8 7

- sizeof(Music) = 4*sizeof (int)=4*4B=16B
- // Music points to the first location
- // *Music = 6
- // *(Music+i) = Music[i]

6

Theater Seat Allocation Vs. Array Allocation

- array declaration ~ allocation of seat numbers to a group
- array size ~ number of seats
- array element value/content ~ name of person in that seat
- array name ~ seat number of the first person in the group
- The array name is a constant pointer to the address of the first element
- int a[5] = {5, 4, 3, 2, 1};
- a is the address of a[0] (hence a pointer)
- a is a constant pointer, as it is fixed to the address of a[0]
- a[0] is the value of the first element of the matrix, i.e. 5.
- *a is the same as a[0], and is the value of the first element

7

Dynamic memory allocation

- Dynamic memory allocation is required when the programmer cannot determine in advance how much space will be required by the program
- Space is dynamically allocated using malloc()
- malloc() takes size in bytes as a parameter and returns void *, i.e., a pointer without a specific type
- Explicit type casting of this pointer is required
- Space should be freed after use using free()
- free() takes as input a pointer returned by malloc()

8

Dynamic memory allocation to arrays

Array allocation

- int n, *Music;
- Music = (int *) malloc(n *sizeof(int));
- for (i = 0; i < n; i++)
  Music[i] = i;
- free(Music); /*space freed*/

Theater seating

- Allot agent to hold seats
- Give n seats for ‘Music’ group
- ‘Music’ group member 0 to n-1 are ‘Aditya’, ‘Bharat’, ...
- ‘Music’ group leaves the seats

9

Dynamic double array

```
#include <stdio.h>
#include <stdlib.h>          // required for malloc
int main()
{
    double *a;
    int i, n;
    double b[7] = {10.0, 9.0, 8.0, 7.0, 6.0, 5.0, 4.0};
    printf("Enter the size of array: ");
    scanf("%d", &n);
    a = (double *)malloc(n * sizeof(double));    /* sizeof(double) is required as
    it is in bytes*/
    printf("Size of a is %d\n", sizeof(a));      // size of the pointer
    printf("Size of b is %d\n", sizeof(b));      /*size of array is the total space
    allotted in bytes*/
    printf("Number of elements in b is %d\n", sizeof(b) / sizeof(double));
```

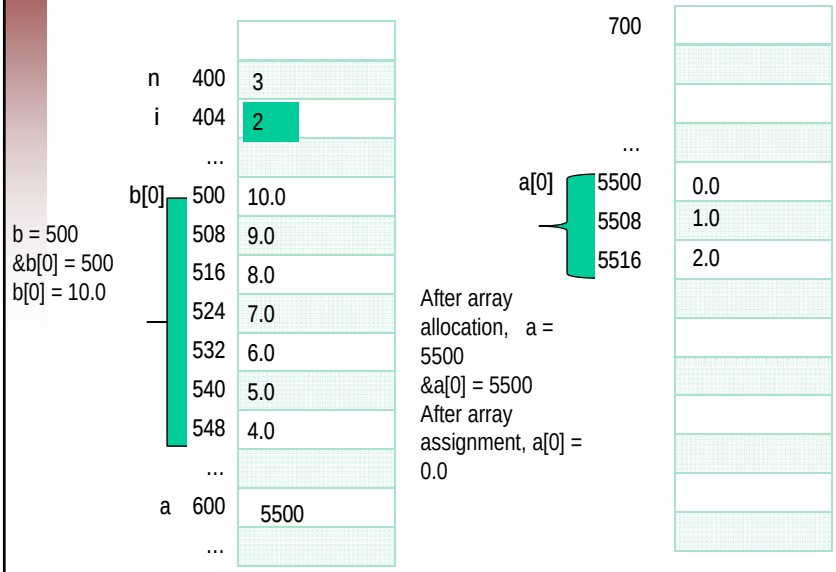
10

Dynamic double array (cont.)

```
    for (i = 0; i < n; i++)
        a[i] = i;    // array notation

    for (i = 0; i < n; i++)
        printf("%lf %lf\t", a[i], *(a + i));
    printf("\n");

    printf("&a is %u, while a[0] is at %u\n", &a, &a[0]);    /*a is a separate
    variable stored elsewhere*/
    printf("&b is %u, while b[0] is at %u\n", &b, &b[0]);   /* b is not stored
    separately*/

    free(a);          // important as otherwise space is not freed
}
```

11

One dimensional array allocation



12

One dimensional array access

- int num[5];
- num: address of num[0]
- num: pointer to num[0]
- num[0]
- num[k]
- &num[k]

- int *num, n=5;
- num = (int *) malloc(n*sizeof(int));
- *num
- *(num+k)
- num+k

13

2D arrays are stored contiguously

	0	
d[0][0]	500	10.0
d[0][1]	508	9.0
d[0][2]	516	8.0
d[1][0]	524	7.0
d[1][1]	532	6.0
d[1][2]	540	5.0
	…	
	600	
	..	
	…	

- 2D arrays are stored row-wise
- double d[2][3] = {10.0,9.0, 8.0, 7.0, 6.0, 5.0};
- d = 500, i.e. address of array d
- d[0] = 500, i.e. address of d[0][0]
- d[0][0] = 10.0
- d[1] = 524, i.e. pointer to d[1][0]
- d[1][0] = 7.0

14

Arrays of pointers

- Since a pointer is a variable, arrays of pointers can be declared
- The declaration

char *a[3];

- declares a to be an array of 3 pointers to char
- a[i] is a pointer to char
- Common way to declare arrays of strings instead of char a[3][30];
- Useful since the strings can be of variable size

```
char *a[3] =
{
`` Kolkata '',
`` Kanpur '',
`` Hyderabad ''
};
```

- The array pointed to by each a[i] is allotted space through dynamic memory allocation

15

Array of pointers to characters

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    char *a[3];
    int i, n;
    for (i = 0; i < 3; i++)
    {
        printf("Enter maximum length of string %d: ", i);
        scanf("%d", &n);
        a[i] = (char *)malloc(n * sizeof(char)); // allocate space for each a[i]
    }
```

16

Array of pointers to characters (cont.)

```
for (i = 0; i < 3; i++)
{
        printf("Enter string %d: ", i);
        scanf("%s", a[i]);
}
for (i = 0; i < 3; i++)
        printf("%s\n", a[i]);
printf("Size of a is %d\n", sizeof(a)); //size of 3 pointer variables=3*8B=24B
for (i = 0; i < 3; i++)
        printf("Size of a[%d] is %d\n", i, sizeof(a[i])); //size of ith char array
printf("a is at %u\n", &a);
for (i = 0; i < 3; i++)
        printf("a[%d] is at %u\n", i, &a[i]);
```

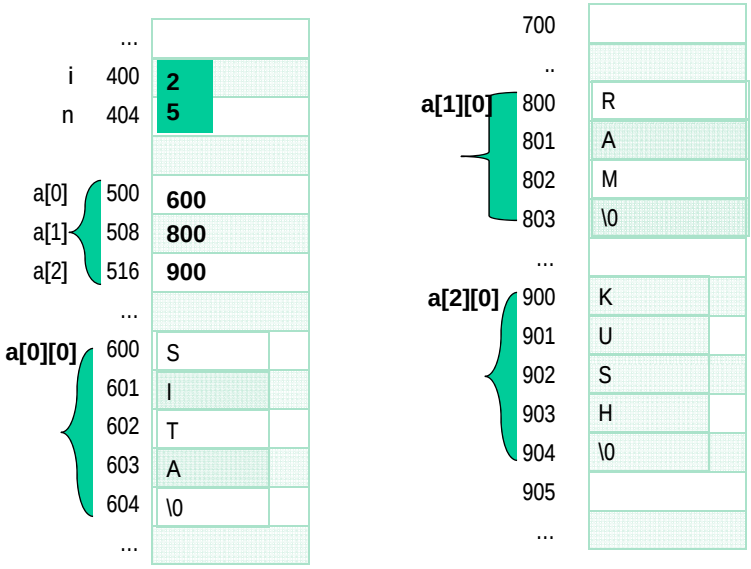
17

Array of pointers to characters (cont.)

```
for (i = 0; i < 3; i++)
        printf("a[%d][0] is at %u\n", i, &a[i][0]);

for (i = 0; i < 3; i++)
        free(a[i]); // free each a[i]
}
```

18

Array of pointers to characters



19

Two dimensional array access

- int num[5][3];
- num: address of num[0]
- num[j]: address of num[j][0]

- num[0][0]
- num[j][k]
- num[j]
- &(num[j][k])
- Can access through pointers

- int **num, n=5, m=3;
- num = (int **) malloc(n*sizeof(int *));
- num+j = (int *) malloc (m*sizeof(int));
- *(*(num))
- *(*(num+j)+k)
- *(num+j)
- *(num+j)+k
- Can access through array notation. e.g. num[j][k]

20