C PROGRAMING

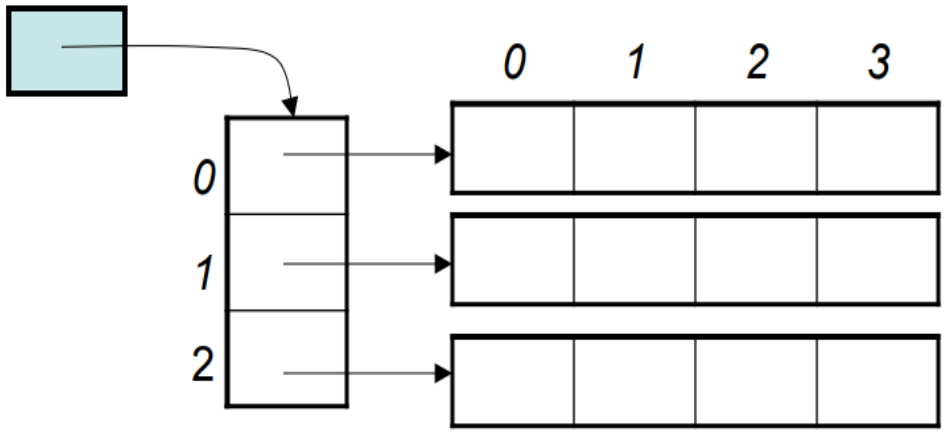
by Akshita Chanchlani @ Sunbeam Infotech

Day4 : 2D Array and Command Line Argument

2D Array

2D Array

- Arrays that we have consider up to now are onedimensional arrays, a single line of elements.
- Often data come naturally in the form of a table, e.g., spreadsheet, which need a two-dimensional array.
- Two-dimensional (2D) arrays are indexed by two subscripts, one for the row and one for the column.
- Example: int a[3][5];
  - Logically it may be viewed as a two-dimensional collection of data, three rows and five columns, each location is of type int.



	0	1	2	3	4
0	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
1	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
2	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]

A 2D array is a 1D array of (references to) 1D arrays.

2-D array

- 2-D array is collection of 1-D arrays in contiguous memory locations.
  - Each element is 1-D array.
- int arr[3][4] = { {1, 2, 3, 4}, {10, 20, 30, 40}, {11, 22, 33, 44} };

	0				1				2			
	0	1	2	3	0	1	2	3	0	1	2	4
arr	**1**	**2**	**3**	**4**	**10**	**20**	**30**	**40**	**11**	**22**	**33**	**44**
	400	404	408	412	416	420	424	428	436	440	444	448
	400				416				436			

2-D array and Pointer

- Pointer to array is pointer to 0^{th} element of the array.
  - Scale factor of the pointer = number of columns * sizeof(data-type).
- int arr[3][4] = { {1, 2, 3, 4}, {10, 20, 30, 40}, {11, 22, 33, 44} };
- int (*ptr)[4] = arr;

ptr	arr	0				1				2			
		0	1	2	3	0	1	2	3	0	1	2	4
400		**1**	**2**	**3**	**4**	**10**	**20**	**30**	**40**	**11**	**22**	**33**	**44**
1000		400	404	408	412	416	420	424	428	436	440	444	448
		400				416				436			

2D Array Declaration

- **Valid Declarations :**

1. int mat[2][2]={{1,1},{1,2},{2,1},{2,2}}; //allowed
2. int mat1[ROW][COL]={{1,1},{1,2},{2,1},{2,2}}; //allowed
3. int mat3[][COL]={{1,1},{1,2},{2,1},{2,2}}; // allowed
4. int mat4[2][2];

- **Invalid Declarations :**

1. int mat[][]={{1,1},{1,2},{2,1},{2,2}};// not allowed
2. int mat2[ROW][]={{1,1},{1,2},{2,1},{2,2}}; //not allowed

Passing 2-D array to Functions

- 2-D array is passed to function by address.
- It can be collected in formal argument using array notation or pointer notation.
- While using array notation, giving number of rows is optional. Even though mentioned, will be ignored by compiler.

Pointer Arithmetic

- Increment operator when used with a pointer variable returns next address pointed by the pointer.
- The next address returned is the sum of current pointed address and size of pointer data type.
- Decrement operator returns the previous address pointed by the pointer.
- The returned address is the difference of current pointed address and size of pointer data type.
- Array can be interchangeably used with pointer that is called as Pointer arithmetic.

Program Demo of Pointer Arithmetic

```
int arr[3][3]={{11,12,13},{14,15,16},{17,18,19}};
int *p[4]={*arr,*(arr+1),*(arr+2)}; //declaring pointer array
```

arr[0][0] = *(*(p))
arr[0][1] = *(*(p+1))
arr[0][2] = *(*p+2)

((p+1)),arr[1][0]

((p+1)+1),arr[1][1]

((p+2)),arr[2][0]
((p+2)+1),arr[2][1]

Command Line Argument

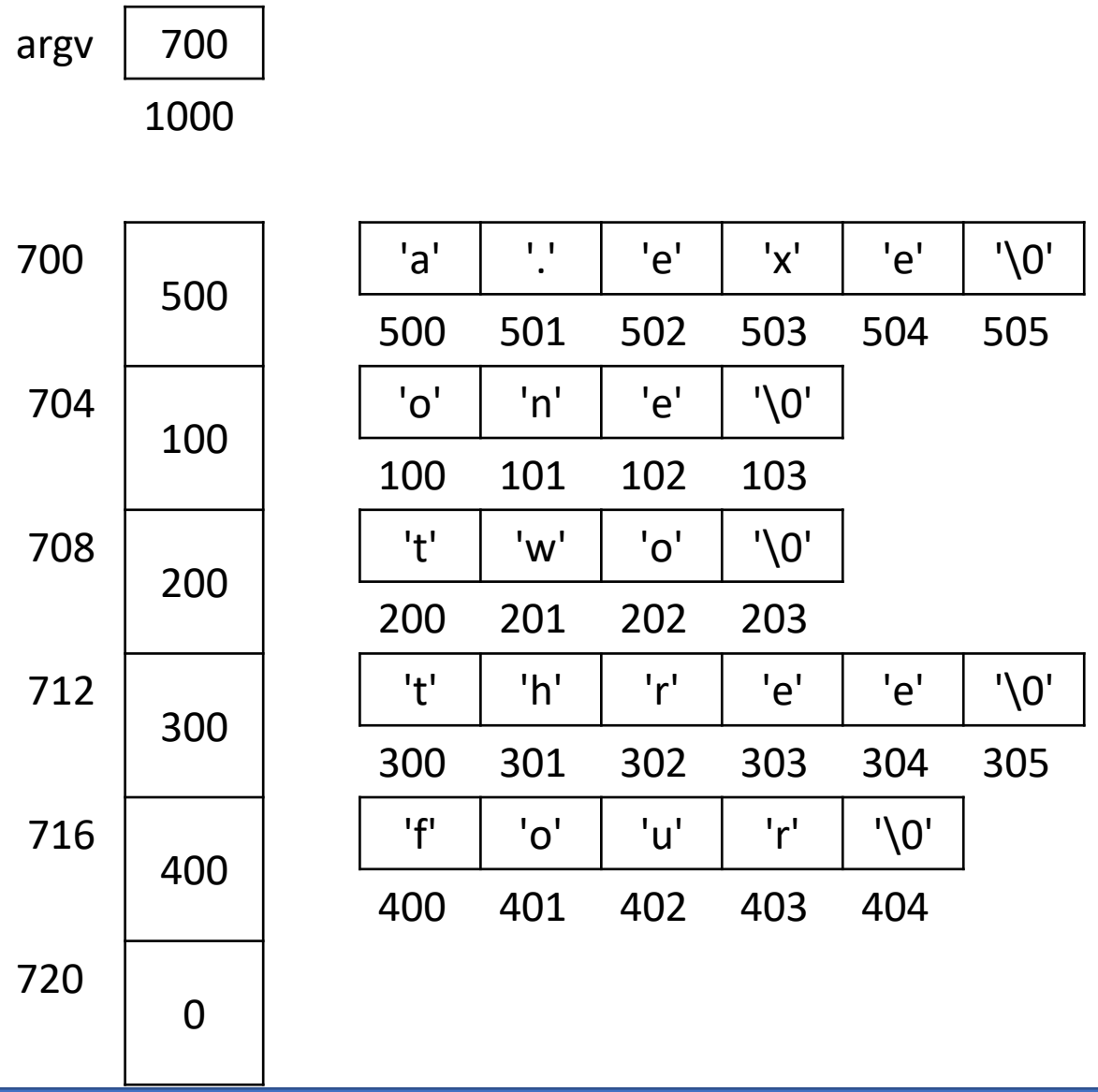
Command Line Argument

- The C language provides a method to pass parameters to the main() function.
- This is typically accomplished by specifying arguments on the operating system command line.
- The prototype for main() looks like:
  - int main(int argc, char *argv[]) { … }
  - The first parameter is the number of items on the command line (int argc).
  - argc always retains the count of arguments passed to main
  - Each argument on the command line is separated by one or more spaces, and the operating system places each argument directly into its own null-terminated string.
    - Note: The name of the program is counted and is the first value.
    - Note: Values are defined by lists of characters separated by whitespace.
  - The second parameter passed to main() is an array of pointers to the character strings containing each argument (char *argv[]).
  - Argv catches actual arguments passed at command prompt to main function
    - Note: The array has a length defined by the number_of_args parameter.
- If we add **char **env** an argument to main it will display list of environment variables.
- Environment variables are used for information about your home directory, terminal type, and so on; you can define additional variables for other purposes. The set of all environment variables that have values is collectively known as the *environment*.

Command line arguments

- Command line arguments are information passed to the program while executing it on command line.
- cmd> a.exe one two three four

```
int main(int argc, char *argv[]) {
        int i;
        for(i=0; i < argc; i++)
                puts(argv[i]);
        return 0;
}
```



Standard main() prototypes

- int main();
- int main(int argc, char *argv[]);
- int main(int argc,char*argv[],char*env[]);
- argc represents number of arguments passed to program when it is executed from command line.
- argv represents argument vector or argument values.
- envp represents system information.

Thank you!

Akshita Chanchlani <akshita.chanchlani@sunbeaminfo.com>