July 24, 2015

Launchpad

Lecture - 6

Pointers & Character Arrays

Aman Bahl

Status of Assignment?

Any doubts?

Initialization of Array.

Arrays can only be initialized and not assigned.

If you need change an array, you need to do it element by element.

Pointers

Address of Operator (&)

To get the address of a variable, we can use **the address-of operator (&)**

```
int p = 5;
cout << &p << endl; // It will print address of the variable;
--------------------------
int arr[3];
cout << arr << endl; // it will show you address of first element.
cout << &arr[0] << endl; // same as above.
```

What are pointers?

I. Pointers are one of the most powerful and confusing aspects of the C/C++ language.
II. A pointer is a variable that holds the address of another variable.
III. To declare a pointer, we use an asterisk between the data type and the variable name

Declaring a pointer variable!

```
int *pnPtr; // a pointer to an integer value
double *pdPtr; // a pointer to a double value
int* pnPtr2; // also valid syntax
int * pnPtr3; // also valid syntax

int *pnptr1, *pnptr3 // Declaring Multiple pointers.
```

Note – The space between the type and variable name.

Initializing the pointer variable.

I. Pointer variable when declared store some arbit collection of 1s and 0s. So we can say that they are pointing to some garbage address.
II. We can initialize the pointer variable to some valid address i.e. address of same type of valid memory.
III. Int x = 10; int *q= &x;
IV. We should never store address of a different type in the pointer variable.

Dereference Operator (*)

An interesting property of pointers is that they can be used to access the variable they point to directly. This is done by preceding the pointer name with the dereference operator (*). The operator itself can be read as "**value pointed to by**"

Therefore the value pointed by q in previous example can be accessed as

```
int r = *q;
// or
cout << *q << endl;
// or
Int z = (*q) + 1;
```

So what is * ?

I. Using * in a declaration of variable or as a function argument - is only signifying that this variable is meant to store an address.
II. Using * in an expression can mean two things
    I. As binary operator - Multiplication
    II. As unary operator – Dereferencing the address.

Assignment in pointers

I. A pointer variable is assignable – it means that we can change the address which it is storing now.

```
int x = 10, y = 20;
int *ptr = &x; // Now ptr is storing address of x
ptr = &y; // Now ptr is storing address of y;
```

II. Assigning value which is being pointed at by the variable.

```
*ptr = 25; // This would change the value to which ptr is pointing to i.e. now y would become 25.
```

Lets see a sample input/output question.

We should never de-reference any garbage address!

Null Pointer

Sometimes it is useful to make our pointers point to nothing. This is called a null pointer. We assign a pointer a null value by setting it to address 0:

```
double *p = 0;
```

Dereferencing Null pointer always gives segmentation fault.

Pointers and Functions

Address are also passed by value to a function!

Pointers and Arrays!

I. Pointers and arrays are intricately linked in the C language
II. An Array is actually a pointer that points to the first element of the array! Because the array variable is a pointer, you can dereference it, which returns array element 0:
III. a[i] is same as *(a + i)

Note – An array name is just an alias to address of first element. There is no separate storage for the variable name. It behaves as a pointer but is not actually a pointer.

Lets see some code!

Pointer Arithmetic

I. Addition, Multiplication, Division of two addresses doesn't make any sense.
II. Addition of an address by a constant integer value i.e. ptr +5 means address of cell which is 5 * sizeof(*ptr) away from ptr.
III. Similar for subtraction.
IV. Again Multiplying/Dividing an address with some constant value doesn't make any sense.
V. Subtracting two address of same type would give you number of elements between them.

Precedence & Associativity

TABLE 2-1. PRECEDENCE AND ASSOCIATIVITY OF OPERATORS

OPERATORS	ASSOCIATIVITY
`()  []  ->  .`	left to right
`!  ~  ++  --  +  -  *  &  (type)  sizeof`	right to left
`*  /  %`	left to right
`+  -`	left to right
`<<  >>`	left to right
`<  <=  >  >=`	left to right
`==  !=`	left to right
`&`	left to right
`^`	left to right
`|`	left to right
`&&`	left to right
`||`	left to right
`?:`	right to left
`=  +=  -=  *=  /=  %=  &=  ^=  |=  <<=  >>=`	right to left
`,`	left to right

Pointer Arithmetic contd..

Lets look at few input/output examples to understand more clearly!

So when you are passing an array to a function, you are actually passing the address of the first element.

Lets convert some problems.

I. Sum of Array
II. Bubble Sort

void sumofarray(int arr[], int N)
is same as
void sumofarray(int *arr, int N)

What would happen if I change the call to
cout << sumofarray(arr+5, 10);

Pointer as a return value from functions

Pointers vs Arrays

I. the sizeof operator
   I. sizeof(array) returns the amount of memory used by all elements in array
   II. sizeof(pointer) only returns the amount of memory used by the pointer variable itself
II. the & operator
   I. &array is an alias for &array[0] and returns the address of the first element in array
   II. &pointer returns the address of pointer
III. Pointer variable can be assigned a value whereas array variable cannot be.

```
int a[10];
int *p;
p=a; /*legal*/
a=p; /*illegal*/
```

IV. Arithmetic on pointer variable is allowed.

```
p++; /*Legal*/
a++; /*illegal*/
```

Character Arrays!

Character Array Basics

I. char str[100];
II. char str[4] = { ‘A’, ‘B’, ‘C’, ‘D’};
III. char str[] = {‘A’, ‘B’, ‘C’};
IV. char str[] = “Welcome”;
V. char str[8] = “Welcome”;

So what are strings?

I. In C/C++ we use a character array to simulate strings.
II. By convention, a string is a sequence of characters followed by a null character.
III. Null characters is a special character whose ascii value is 0 and its representation is '\0'
IV. In the previous slide example 4 and 5 are valid strings.

Printing a character array

I. cout command treats characters address differently.
II. If you give it any other type of address it will just print the address
III. But if you give it an address of type character it will print characters byte by byte starting from that byte till it finds a byte which stores null character.
IV. It doesn't care about the allocated space.

Lets see it with some code.

Reading a string.

I. We can read character by character from the screen and keep adding to an array till we find our delimiter which in most cases is '\n' and append 0 character to the end of the array.

II. cin.getline

cin.getline

I. cin.getline(char * BA, int max_space);
II. cin.getline(char * BA, int max_space, char delimiter);

max_space is the available space starting from the passed address.

delimiter is the character which specifies the end of the string. By default it is '\n'.

cin.getline would automatically add '\0' at the end.

Since end of the string can be checked by looking for null character('\0') we don't need to pass number of elements to a function.

Lets do some problems!

I. Calculate Length of the String
II. Check if a string is palindrome or not
III. Read N strings from a user and print the largest string.
IV. Write a function which takes two strings A and B and appends B to A

Again – We can only initialize the array and not assign!
So if you want to update the string, you need do it character by character.

Always remember to append null character at the end of the string after any operation.

char * ptr VS char arr[]

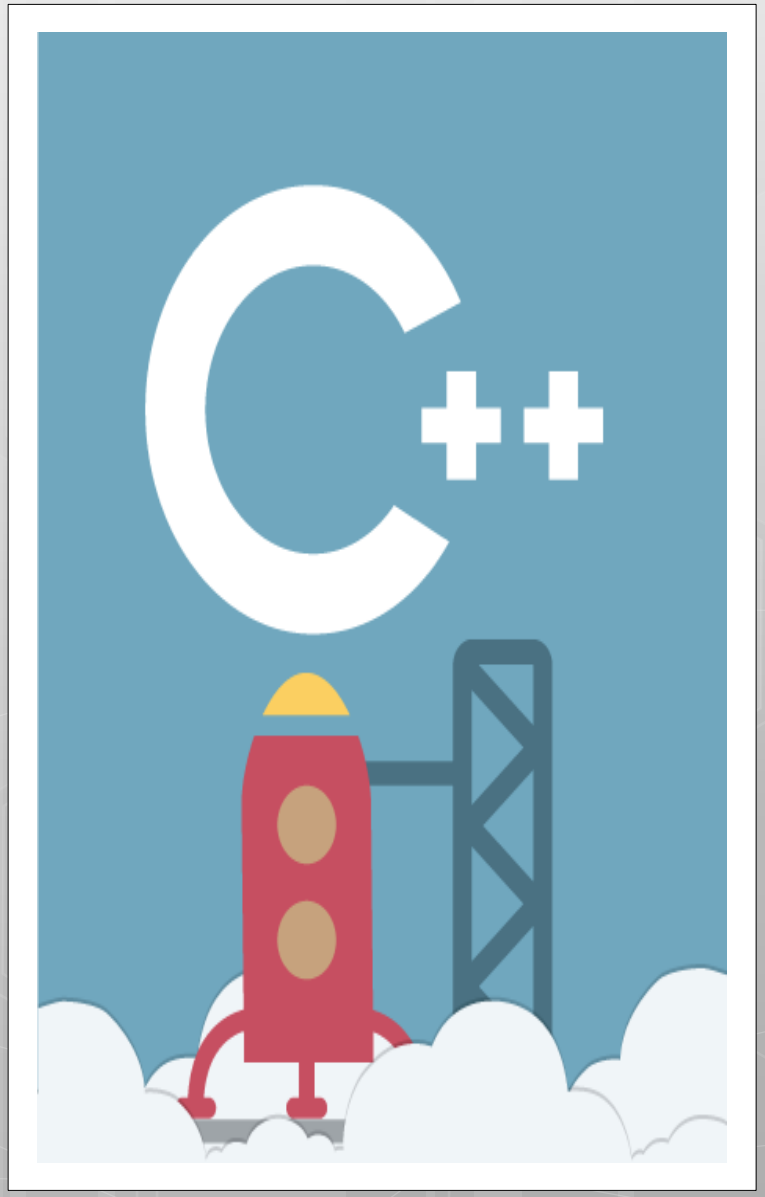
I. char array[] = “abc” sets the first four elements in array to ‘a’, ‘b’, ‘c’, and ‘\0’
II. char *pointer = “abc” sets pointer to the address of the “abc” string (which may be stored in read-only memory and thus unchangeable)

Time to try?

I. Write a function to reverse a string.
II. Given a string rotate it by n characters. e.g. if the string is CodingBlocks and n =3 then the output should be cksCodingBlo
III. Write a function to check if two strings are permutations of each other.
IV. Write a program to print all substrings of a given string

What is next class about?

I. Arrays contd… + Recursion



Thank You!

Aman Bahl

aman.or.b@gmail.com
+91-9908124628