11.15.19

C Strings

* Array of characters
* Terminated by null ‘\0’

char str[10] = “hello”;

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ‘h’ | ‘e’ | ‘l’ | ‘l’ | ‘o’ | ‘\0’ |  |  |

0 1 2 3 4 5

Traverse a C string

1.

for (int c = 0; str[c] != ‘\0’; c++) {

…

}

2.

for (int c = 0; c < strlen(str); c++) { //strlen ~ .size()

…

}

Assignment:

string str1 = “hi”;

string str2 = str1;

str2 += “!!!”;

cout << str1 << endl << str2;

Output:

hi

hi!!!

C String

char cstr1[20] = “hi”;

char cstr2[10];

strcpy(cstr2, cstr1); //copying each element of cstr1 into cstr2 (traverse)

strcat(cstr2, “!!!”); //adding “!!!” to cstr2 – appending

cout << cstrl << endl << cstr2 << endl;

Output:

hi

hi!!!

Simplified Memory: large array, index = address

Pointer

* Variable holding addresses of other variables

int a = 5;

int b = 6;

char ch = ‘k’;

int\* ptr = &a; (pointer – arrow to an int, in this case, a)

ex. a= 4 bytes (location 500-503)

pointing to directly the first instance of the variable -> variable type determines the size

* specification of the type for pointers
  + when doing pointer arithmetic -> know how much it is going
    - ex. ptr + 2 -> advance by 8 bytes as each int holds 4 bytes

int a = 5;

int b = 6;

char ch = ‘k’;

int\* ptr = &a;

ptr = &b; //ptr -> address to 509 (address of b, holding the value of 6)

\*ptr = 10; (go to the address of b -> \*(&b) = 10 -> b =10)

* &: addressing
* \* going to that address
* initialized: variable type\* -> going to the address of that variable type

int c = b; -> copying the value //c = 10;

Pointers and Arrays: ex~

double arr[4] = {5, 10, 15, 20};

double\* ptr = arr; //array itself is pointers to different values

OR

double \*ptr = &(arr[0]);

Only & of an non-pointer object

void doubleElems(double\* start, int len) {

}