CSCI 1300

Strings, Arrays and random numbers
June 16th, 2021

Outline

- Strings
  - How to initialize it?
  - Different ways to take input
  - String concatenation
  - String Functions
    - length, at, substr, erase, find
  - Converting strings to other data types
  - check characteristics of a string
- Arrays
  - Initialize arrays
  - Indexing arrays
  - Multi dimensional arrays
  - Loops on arrays
- Pseudorandom numbers

Please use the github link for the programing examples and slides.
https://github.com/rahul-aedula95/CSCI-1300

What are strings?

- They are sequence of characters (any symbol which can be expressed on the keyboard and some which can’t be.
- Usually these symbols also have numerical equivalents (look into what the ASCII codes are)
- The symbols are not interpreted to their data type directly
  - For example 3 is not equal to “3”
- String data type is not a primitive type which exists in c++ natively rather it has been made common because of the demand in the c++ community

Initialize strings

- Some compilers don’t need this header file (preprocessor directive) but it is safer to include it
  - #include<string>
- When you want to initialize a string go ahead and treat it like a variable use have used so far.
  - string varName = “Some string inside quotes”;
  - string str = “Hey there”;
- You can also make it empty string by default:
  - string str;
  - string str = “”;

Two ways to take input to a string variable

- Using cin:
  - string var;
  - cin >> var;

- Using getline:
  - string var;
  - getline(cin, var);

Strings and their indexing

Let us look at the word “HELLO”

H	E	L	L	0
0	1	2	3	4

Blank spaces between words also get counted as a character (example: “HI THERE”

H	I		T	H	E	R	E
0	1	2	3	4	5	6	7

String concatenation

- The plus (+) operator work to concatenate two strings if given the choice.
- string str1 = “Hi”;
- string str2 = “ there”;
- string str3 = str1 + str2; // str3 will contain Hi there

string functions

- Here str is a string variable.
- str.length() and str.size() - both these functions return the number of characters.
- str.at(index) - returns what character exists at that index.
- str.substr(start_index, size) - returns the substring starting from start_index and the next consecutive character of length size.
- str.erase(start_index, size) - returns the string after erasing the substring starting from start_index and the next consecutive character of length size.
- str.find(substring) - returns the starting index of the substring

converting strings to another data type

- As mentioned strings do not interpret other data types directly
- So we need to convert a string to a data type
- string str = “4”;
- int num = stoi(str) // now num will be the integer conversion of the string.
- string str2 = “5.7”;
- float f = stof(str2)
- double d = stod(str2)
- // Both these will convert the string to float and double respectively.
- can also use the function to_string to convert any of these types to string

functions to test characteristics of a string

isdigit - returns true if string is a digit (number)

isalpha - returns true if string has alphabet

islower - returns true if string is lower case

isupper - returns true if string is upper case

Arrays

- Similar to strings arrays are basically sequence of other data types
- There are basically consecutively allocated memory for the same data type.
- Visualization of array

Value	10	20	30	40	50	60
Index	0	1	2	3	4	5

Better to store data as arrays rather than use multiple variables to store values.

Initialize an array

If you want to preset values on array

```
int arr[5] = {10,20,30,40,50};
```

You can also create an empty array with some size and allocate values in later.

```
int arr[5];
```

indexing arrays

- If you want to set or access a value inside an array you can refer to the index.
- Lets say
- int arr[5] = {10,20,30,40,50}
- indexing starts from zero
- so arr[0] has the value of the element 10
- arr[1] has the value 20 and so on

Multi dimensional arrays

- We will mostly be dealing with 2 dimensional arrays so lets talk about that since it is easier to visualize.
- Row index and column index can be used to address values

Index	0	1	2
0	10	20	30
1	40	50	60
2	70	80	90

Multi dimensional arrays

If you want to create one we can follow:

int arr[3][3] = {{10,20,30},{40,50,60}, {70,80,90}};

This will create the values as shown in the previous example

we can use arr[row_number][column_number] to access an element at some row or column.

Loops on arrays

- Since arrays can be very long using loops are very useful to index.
- let us say if an array is int arr[5] = {10,20,30,40,50};

- Since we know the size we can loop over it to access each element individually.
- for (int i = 0; i<array_length;i++) // here our array length is 5

```
{
  cout<<arr[i]<<endl;
}
```

- We can also use this technique on 2d arrays as well.

Pseudorandom number generation

True random does not exist and often requires a way of generating a random number

```
#include<ctime>
int main()
{
int randomNumber;
srand(time(NULL)); // This seeds the rand function.
randomNumber = rand() % 3; // This generates random numbers from 0 to 2. (3 is not included)

return 0;
}
```