CS 16 4/30/19

Makefiles

Variables can be declared like the one below and accesses using $(VARIABLE\_NAME)

( e.g. $(CPP) )

CPP=g++ # variable

FILENAME.cpp : FILENAME.o

g++ -c FILENAME.cpp # makes object code (FILENAME.o)

# $@ is shorthand for the rule name

 # $^ is shorthand for the right side of the ‘:’ after the rulename

# Here is what we had: g++ -c FILENAME.cpp

# Now using variable CPP the above line becomes

# $(CPP) -c FILENAME.cpp

EXECUTABLE : FILENAME.o

g++ FILENAME.o -o EXECUTABLE # makes executable from .o file

# g++ $^ -o $@ (Is another way of saying the line above)

clean:

 rm \*.o EXECUTABLE # removes all compiled code

Memory Storage

Italicized items are not tested on:

*Little Endian – Stores Value 0x00 00 00 2A (ignore whitespace in memory address) as [2A,00,00,00] in memory (Intel and AMD processors)*

*Big Endian - Stores Value 0x00 00 00 2A (ignore whitespace in memory address) as [00,00,00,2A] in memory*

**Arrays**

Initializing array:

 int arr[5]; // has 5 elements which are all integers

Initialize a 2-D array:

 double myArr[10][2]; // has 10 rows and 2 columns that all have the double type

Initialize and assign values to an array in the same line:

 char word[5] = {‘a’,’b’,c’,’a’,’t’};

Sample file:

#include <iostream>

using namespace std;

int main()

{

 char word[] = {'a','b','c','a','t'};

 for(char elem : word){

 cout << elem << endl;

 }

 return 0;

}

Output:

a

b

c

a

t

Using arrays in a function:

Changing the elements in the array in the function, also changes the elements in the main function (or any other function that calls it)!

Sample file:

#include <iostream>

using namespace std;

int findFirstIndexOfC(char arr[], int sizeOfArr){

 for(int i = 0; i < sizeOfArr; ++i){

 if(arr[i] == 'c'){ // check if the element in the array is the character c

 arr[i]= '~';

 return i;

 } // when we find the first c in the array we change it to a ~, exit, and return the index where we found the c

 }

}

int main()

{

 int index;

 char word[] = {'a','b','c','a','t'};

 cout << "Before we run the function the value of each element in the array words is : " << endl;

 for(char elem : word){

 cout << elem << endl;

 }

 index = findFirstIndexOfC(word, 5);

 cout << "The first c occurs at index " << index << endl;

 cout << "After we run the function the value of each element in the array words is : " << endl;

 for(char elem : word){

 cout << elem << endl;

 }

 cout << endl << endl << "Notice that the function call changes the value of the original array which is unlike other data structures." << endl << endl;

 return 0;

}

Output:

Before we run the function the value of each element in the array words is :

a

b

c

a

t

The first c occurs at index 2

After we run the function the value of each element in the array words is :

a

b

~

a

t

Notice that the function call changes the value of the original array which is unlike other data structures.

Binary Search:

Only should understand at a high level.

Array is assumed to be sorted.

Start in the middle of the array.

If the value you are looking for is bigger than the middle value of the array then the value you are looking for must be in the right half of the array.

 So we only have to search this side.

Else the value you are looking for is smaller than the middle value of the array then the value you are looking for must be in the left half of the array.

 So we only have to search this side.

So we keep doing this until we find the value we are looking for.

(E.g. int arr[] = {1,2,3,4,5,6,7,8,9}) (9 elements)

Search for 2.

First choose arr[9/2] (which is index 4 (remember integer division truncates))

 Arr[4] is 5.

 5 > 2, so search the left half of the array (index 0 to 4) (5 elements).

Now choose arr[5/2] (which is index 2 (same rule as before))

 Arr[2] is 3.

 2 < 3, so search the right half of this smaller array. (index 2 to 4) ( 3 elements).

Now choose arr[3/2].

 Arr[1] is 2.

 2 == 2, so we return the index that we found 2 at (index 1).

Call by Reference

Use this to change values of variables that we are using in the main function when we pass it to other functions.

All we have to do is add the ‘&’ symbol when we pass a variable to the function.

Sample file:

#include <iostream>

using namespace std;

void changeVar(int &a){

 a = 5;

}

int main()

{

 int k = 3;

 cout << "Before the function call the value of k is " << k << endl;

 changeVar(k);

 cout << "After the function call the value of k is " << k << endl;

 return 0;

}

Output:

Before the function call the value of k is 3

After the function call the value of k is 5

Pointers

Pointers hold the address of the data type it is declared with.

Declare a pointer:

 int \*p; // a pointer named p

 int r = 3; // a normal int named r which has value 3

 p = &r; // p now stores the address of r

 // Now we can change the value of r with \*p.

 \*p = 2; // Now r’s value is 2.

Here is how we use pointers similar to the program above:

#include <iostream>

using namespace std;

void changeVar(int \*a){

 \*a = 5;

}

int main()

{

 int k = 3;

 cout << "Before the function call the value of k is " << k << endl;

 changeVar(&k);

 cout << "After the function call the value of k is " << k << endl;

 return 0;

}

Output:

Before the function call the value of k is 3

After the function call the value of k is 5