Discussion 10.25.19

Chars

* ASCII – char -> represented in number (able to do arithmetic function)
* Ex. ch letter = ‘a’; letter++; cout << letter;

char nine = ‘9’;

char eight = ‘8’;

cout << nine – eight; -> printing at the integer of the subtracting value (1 in this case)

String: sequence of chars

string str = “w$. ]”;

char ch = str[4];

->labeling from 0 (0, 1, 2, 3) -> str[4] undefined

Unable to map string to char (ex. string str1 = str[3]; -> wrong, cannot string s = ‘a’;)

String functions

Length of a string: .size() – string s = “hello”; int size = s.size();

<cctype>

isdigit(char) – return nonzero int if digit (0-9)

isalpha(char) – return nonzero int if letter

* Implement if(cctypeFunction) -> test it directly
  + If statement – runs when the condition is not 0

islower(char) – return nonzero int if uppercase letter

isupper(char) – return nonzero int if lowercase letter

Returning (converting character)

tolower(char) – return uppercase letter if lowercase letter, else return original

toupper(char) – return lowercase letter if lowercase letter, else return original

Exercise:

[Discussion 10.25.19 Codes.txt](Discussion%2010.25.19%20Codes.txt)

* Another way to find unique string (create an empty string, add unique ones into the empty string -> see if it’s in there -> count

Adding string:

string str = “Hel”;

string str1 = “lo”;

count << str+str1; -> “Hello

Adding character as well

Function:

Why? Reduce repetition, organization (readability), easier to debug (and test)

Nesting functions -> cannot -> each function has to be in one’s own scope

Function signature:

returnType functionName (paramType1 name1, paramType2 name2) {}

Calling – FunctionName(varName1, varName2)

Example:

validPoint(int x, int y)

{

x = x+1;

if (x > y)

return true;

return false; //required to stop

}

int main ()

{

int var = 5;

int y = 10;

bool valid = validPoint(var, y); //return false

}

returnType can be void

Separate memory location -> not reference to each other

Pass by Value

* Modification do not affect the input
* Out of scope (do not matter if variable shares name) – based on the position in the parameter list (not the actual name) – need to be same type

Pass by Reference

validPoint(int x, int &y)

{

x = x+1;

if (x > y)

return true;

return false; //required to stop

}

int main ()

{

int var = 5;

int y = 10;

bool valid = validPoint(var, y); //return false

cout << y; //print 11;

}

y,main = 10 = y,vP

(Pass by reference)

instead of

y,main = 10

y.vP = 10

(Pass by value)

& in parameter

6.)

int input;

cin >> input;

int c;

if (input < 1)

cout << “Error” << endl;

else if (input == 1)

cout << 0;

else {

int min = 2;

while (min <= input)

{

min \*= 2;

c++;

}

cout << c <<endl;  
}

int input;

cin >> input;

int c = 0;

if (input < 1)

cout << "Error" << endl;

else if (input == 1)

cout << 0;

else {

int max = 2;

while (max <= input)

{

max \*= 2;

c++;

}

cout << c << endl;

}