10/21/2019

Enter some texts: *Everyone, hello!*

The number of E’s (upper and lower case) is *4*

#include <iostream>

#include <string>

using namespace;

int main ()

{

cout << “Enter some texts: “;

string s;

getline(cin, s);

int eAmt;

for (int i = 0; i != s.size(), i++)

{

if (s[i] == ‘e’ || s[i] == ‘E’) //Not: if (t[k] == ‘E’ || ‘e’)

eAmt++;

}

cout << “The number of E’s (upper and lower case) is “ << eAmt << endl;

}

string s = “Hello”; // s is a string; “Hello’ is a string literal

char ch = s[1]; // ch is a char (initialized to a lower case e)

* ‘E’ is a character constant

Tab ‘\t’

Next line ‘\n’

Different types (character is not string, vice versa)

char c = ‘A’ // ‘’ -> type char

char c = “A” // ERROR -> won’t compile

string s = “A”; // “” -> type string

string s = ‘A’ // ERROR -> won’t compile

string: library type -> cannot convert automatically

cout << “Enter a phone number: “;

#include <iostream>

#include <string>

#include <cctype> //declare functions (isdigit, isalpha, etc.)

using namespace std;

int main ()

{

string num;

getline(cin, num);

int numOdig;

for (int k = 0; k != num.size(); k++)

{

if (isdigit(num[k]))

numOdig ++;

}

if (numOdig != 10)

cout << “A phone number must have 10 digits.” << endl;

}

C++ library -> collection of function -> classification

isdigit(c) -> ‘0’ ‘1’ … ‘9’

islower(c) -> ‘a’ ‘b’ … ‘z’

isupper(c) -> ‘A’ ‘B’ … ‘Z’

isalpha(c) -> ‘a’ … ‘z’ ‘A’ … ‘Z’

if(! boolean) -> ! not operator

Ex: if ( ! (x >= 1 && x <= 10)) -> if (x < 1 || x > 10)

string s;

getline(cin, s);

* converting to first letter to uppercase

toupper (s[0]) -> return uppercase of lowercase letter

if (s != “”)

s[0] = toupper(s[0]) -> if not lowercase, then it returns unchanged

toupper(*HAS TO BE CHAR)*… toupper(s[0]) //doesn’t do anything – mistake -> don’t change

s[0] = toupper(s[0]) -> What if s is an empty string? -> undefined behavior

toupper

tolower

if (t[k] == ‘E’ || t[k] == ‘e’) -> if (tolower(t[k]) == ‘e’) -> change of local

number – decimals -> commas (based on location methods) – stay in default

int main ()

{

string num;

getline(cin, num);

int numOdig;

for (int k = 0; k != num.size(); k++)

{

if (isdigit(num[k]) )

numOdig ++;

}

if (numOdig != 10)

cout << “A phone number must have 10 digits.” << endl;

}

C++ library -> collection of function -> classification

isdigit(c) -> ‘0’ ‘1’ … ‘9’

islower(c) -> ‘a’ ‘b’ … ‘z’

isupper(c) -> ‘A’ ‘B’ … ‘Z’

isalpha(c) -> ‘a’ … ‘z’ ‘A’ … ‘Z’

if(! boolean) -> ! not operator

Ex: if ( ! (x >= 1 && x <= 10)) -> if (x < 1 || x > 10)

string s;

getline(cin, s);

converting to first letter to uppercase

toupper (s[0]) -> return uppercase of lowercase letter

if (s != “”)

s[0] = toupper(s[0]) -> if not lowercase, then it returns unchanged

toupper(HAS TO BE CHAR)… toupper(s[0]) //doesn’t do anything – mistake -> don’t change

s[0] = toupper(s[0]) -> What if s is an empty string? -> undefined behavior

toupper

tolower

if (t[k] == ‘E’ || t[k] == ‘e’) -> if (tolower(t[k]) == ‘e’) -> change of local

number – decimals -> commas (based on location methods) – stay in default

Functions

void greet ()

{

for (int k = 1; k <= 3; k++)

cout << “Hello” << endl;

}

int main ()

{

…

greet();

…

…

…

greet();

}

Organization: main routine -> last on the file ~ opposite of real life[

Reorganized:

void greet(); //function prototype – declaration -> passing through the function

Run through the bottom -> detail of the body of the function -> machine language translation

void greet (int n, string msg);

int main ()

{

…

greet(3, “Hello”);

…

…

…

string s;

getline(cin, s);

int x;

cin >> x;

greet(x, s);

}

void greet (int n, string msg)

{

if (msg == “I hate you!”)

{

cout << “I like you!” << endl;

return;

}

for (int k = 1; k <= n; k++)

cout << msg << endl;

}

Without void greet () {} -> definition at the end -> failed when missing

Unable to declare a function inside the main function -> cannot nest functions (comp. err)

Main routine -> called greet () -> complier already checked with declaration above -> remember where to come back -> setup the environment for the greet function -> copy the argument into the corresponding parameter -> run the function -> leave the loop (since k is declared inside the function -> irrelevant (reset) ~ variables gone) -> repeat

Distinction between argument and parameter ~semi-interchangeably

int square (int k);

int main ()

{

cout << “Square what number?”;

int a;

cin >> a;

int x = square(a); //a = 3 -> x = 9

int y = square(x+3)//x= 9 -> y= 144

}

int square (int k)

{

int sq = k\*k;

return sq;

OR

return k\*k;

}

Assignment 3 – Warm up (function reviews + mistakes)

Ex:

Enter a phone number: (310)825-4321

The digits in the phone number are 3108254321

#include <iostream>

#include <string>

#include <cctype>

using namespace std;

boolean phoneValidate (string p);

string digitsOf(string p);

int main ()

{

cout << “Enter a phone number: “;

string phone;

getline (cin, phone);

if (phoneValidate(phone))

{

} else

{

}

}

bool phoneValidate (string p)

{

int x = 0;

for (int a = 0; a != p.size(); a++)

{

if (isdigit(p[a]))

x++;

}

return (x == 10);

}

string digitsOf(string p)  
{

string digit;

for (int a = 0; a != p.size(); a++)

{

if (isdigit(p[a]))

digit += p[a];

}

return digit;

}

10/23/2019

Predicate Function

Predicate naming convention:

Ex: Ray is an engineer.

-> Property of subject

Ex:

isdigit(s[k]), phoneValidate, hasMoreThanTwoPrimeFactors(n), livesIn(personName, city)

Similar name in CS to English

if(condition)

true;

else

false;

-> return condition;

Ex: use of polar coordinate vs cartesian coordinate

1st Attempt WRONG

void polarToCartesian (double radius, double theta, double& xx, double& yy);

-> unable to change outside variables

int main ()

{

double r; double angle; …//get values

double x; double y; polarToCartesian(r, angle, x, y)

…; ;

double 2\*r, angle, x2, y)

}

void polarToCartesian (double radius, double theta, double& xx, double& yy)

{

double r = radius;

double angle = theta;

xx = radius\*cos(angle); yy = radius\*sin(angle);

}

Passing by value -> no effect on the original argument – copied version is changed

Passing by reference: no copy of the original is made -> connected variable name

Any class is a type by ->pass by reference

- doble& “reference to duble, another name for existing double)

Void censorDIgits(string s)

Int main ()

{

Cout <<”What should I censor?”;

String msg;

csensorDigit(msg)

out << msg << endl;

}

Void censorDigits (string)

{

For (int k = 0; k != size(); k++)

{

If (isdigit(s[k])

S[k] =

}

Alternative:  
string censorDigits2 (string s);

Int main ()

{

string msg;

getline(cin, msg);

Cout << censorDIgits2(msg) << endl

String m2 = cenrsorDigits2(msg);

msg = censorDigits(msg);

}

String censorDigits2 (string s)

{

for (…)

return a;

}

Continuation of phone code:

#include <iostream>

#include <string>

#include <cctype>

using namespace std;

boolean phoneValidate (string p);

string digitsOf(string p);

int main ()

{

cout << “Enter a phone number: “;

string phone;

getline (cin, phone);

if (phoneValidate(phone))

{

} else

{

}

}

bool phoneValid (string p)

{

int x = 0;

for (int a = 0; a != p.size(); a++)

{

if (isdigit(p[a]))

x++;

}

return (x == 10);

}

string digitsOf(string p)  
{

string digit;

for (int a = 0; a != p.size(); a++)

{

if (isdigit(p[a]))

digit += p[a];

}

return digit;

}

while(true)

{

string phone;

cout << “Enter a phone number: “;

getline(cin, phone);

if (phoneValid(phone))

break;

cout << “A phone number must have 10 digits; try again.” << endl;

}

cout << “The digits in the number are “ << digitsOf(phone) << endl;

“N-and-a-half-times loop”

Ex: for (;;) {}

Why inf loop? (ex. microwave – wait for an event -> handle an event – always ready)

continue;

-> abandoned the rest of current iteration of the nested current loop and go on to the next iteration

Can be used when if… else… - else is through the end of the for loop -> continue skips the rest -> shifting the codes to the left (cleaner)