The following are members of <cctype> library, but now they are members of <iostream>

char c; //test whether c=space or not if (isspace(c)) returns true if c=’ ‘ false otherwise.

//test whether the content of c is uppercase letter ‘A’-‘Z’ if(isupper(c)) true if c = ‘A’-‘Z’

//test whether the content of c is lowercase letter ‘a’-‘z’ if(islower(c)) true if c=’a’-‘z’

//test whether c=’A’-‘Z’ or ‘a’-‘z’ if(isalpha(c))

//test whether c=’0’-‘9’ if(isdigit(c))

//test whether c=’A’-‘Z’,’a’-‘z’,’0’-‘9’ if(isalnum(c))

//note if(isdigit.. and so on has total of 7 letters)

there are two more

c=toupper(c); //converts the content of c to uppercase.

c=tolower(c); //converts the content of c to lowercase

if we have c=’p’

, if(c>=’A’ && c<=’Z’) //checks if c is uppercase this would be false for c

|| (c>=’a’ && c<=‘z’) //checks if c is lowercase true for c

|| (c>=’0’ && c<=’9’) //checks whether c is a number false for c

Ex. read a sentence and count the number of words, digits, and uppercase letters

ex sentence: Today Is Wednesday 29 //3 uppercase, 3 spaces, 2 digits, 7 lowercase

char c;

int numWords=0;

int numDigits=0;

int numUpper=0;

cout<<”Enter a sentence”<<endl;

while(cin.get(c),c!=’\n’) //lets you take in entire sentence until you get to new line

{

if(isspace(c)) numWords ++;

if(isdigit(c)) numDigits++;

if(isupper(c)) numUpper++;

}

USERDEFINED FUNCTIONS

1. functions that do not return any values are called void

ex: void displayMenu()

{

cout<<”drinks. . . . . . .1.25”

<<”Sandwiches. . . . .3.25”;

} //void has no return statement

1. functions that return only one value

//we want a function to return true or false. we wanna know if c is uppercase or lowercase

NOTE!

!!Character ASCII Memory!!

128 64 32 16 8 4 2 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |

‘A’ 65

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |

‘B’ 66

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |

‘C’ 67

‘a’ 97->97-64+33, 33->32+1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

“ABC” “ABB” -> “ABC” > “ABB”

65 66 67 65 66 66

heart is cout<<char(3); //by doing char(theNumber) will display the asci character associated with that number.

//there is total of 255 ASCII

for(int i=1; 1<-255; ++i)

cout<<i<<’\t\<<char(i)<<endl; //displays number 1-255 and the associated //character from asci 1-255

boolean isvowerl(char c)

{

//we wanna capitalize it first or noncapitalize.. doesn’t matter

//we just want it to be consistent first

c=toupper(c); //this now turns c to a capital letter

if(c==’A’ || c==’E’||c==’I’||c==’O’||c—‘U’)

return true;

else

return false

}

1. return 1 or more values

int a=5, b=6; //we want to compute the average of a and b

float average;

average=computeAverage(a,b);

float computeAverage(int num1, int num2) //we want the return //value to be float

{

return(num1+num2)/2.; //reason for . because we want float

//or we can also do is force the returntype by doing

//float(num1+num2)/2;

}

computeSumAverage()

//now we want a, b, sum, average

int sum;

int a=5, b=6;

float average;

computeSumAverage(a,n,average,sum) // our function will receive //the values from these variables

computeSumAverage(int x, int y, float &z, int &w) //x will be a, y will be b, z will be //average, and w will be sum. also by //adding & you will receive the actual value of average and sum //and not a copy of it

{

z=x+y;

w=z/2.;

}

computeSumAverage(int x, int y, float &z, int &w)

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
| 5 | 6 |

computeSumAverage(a, b, average, sum) //a and b are passed by //value, while z and w are passed by reference… with reference //you are actually sharing the same location of it

the ampersand& goes after the type

float& z;

void f(int a, int&b)

{a++; --b;}

void h(int& a, int& b)

{a \*=2; b \*=3;}

int g(int a. int b)

{

return a+b;

}

int main()

{

int x=5, y=8;

f(x,y); //x becomes a, y becomes b so a is 5, b is 8

//x is passed by value so you pass a copy of it //which is 5, now y is passed be reference so you pass the actual //location of y which is 8 for now

//once it passes the f function, x changed to 6 in there but only in that function, y on the other hand got reduced by 1 so it is not 7, since it is passed by reference, it stays as 7 throughout the program

h(x,y); //in function h, x is still 5 because we only coming from function f we only passed a copy of it, but now the function of h accepts x as a reference because of &. y is passed by reference too so whatever happens to y in function h, y will be changed as well because it is passed by reference. therefore coming out of h, x’s value changes and now becomes 10, and y becomes 21 in that function, outside that function is 21 so when we go to function g, x is 10, y is 21

cout<<g(x,y) //now this returns a value of whatever value was in function g, which shows a+b where a is 10 because x was passed to it, and b is 21 bec y is passed to it, so the output will be 31.

}

cin.ignore();

cin.ignore(5);

cin.ignore(10,’ ‘);

//these are overloaded functions, functions with the same name but do diff things

example:

void print(char c)

{cout<<c;}

void print(int x)

{cout<<x;}

void print(string y)

{cout<<y;}

int main()

{

print(“Tom”); //by putting tom, the print function will look for the one that accepts a string type.

print(‘A’); //by putting A, the print function will look for char type

print(2); //now were looking for the function who’s receiving one integer

}

//these are called overloaded functions.. a lot of same name but have diff parameters

arrays to print with functions

int a[5]={3,2,1,9,8};

char b[3]={‘a’,’f’,’r’};

string c[4]={“Mon”, “Tue”, “Wed”, “Thur”};

display(a, 5);//a, size 5

display(b, 3);//b, size 3

display(c, 4); //these 3 functions have the same name

void display(int x[], int n)

{

for(int i=0; i<n; ++n)

cout<<x[i]<<’\t’;

}

void display(char x[], int n)

{

for (int i=0; i<n;++i)

cout<<x[i]<<’\t’;

}

void display(string x[], int n)

{

for (int i=0; i<n;++i)

cout<<x[i]<<’\t’;

}

1. FUNCTION TEMPLATE read this homie! syntax n stuffs

//use the word template lowercase

//make type x a variable since it is manmade type we use uppercase

//one function will take care of all of it instead or having to overload them

template<class T>

void display(T x[], int n)

{

for(int i=0; i<n; ++i)

{cout<<x[i]<<’\t’;}

}

int a[5]={3,2,1,9,8}; //T is int

char b[3]={‘a’,’f’,’r’};//T is char

string c[4]={“Mon”, “Tue”, “Wed”, “Thur”};//T is string

show(“tom”,”Jerry”);

show(“Tom, 25);

show(3.5, “is my GPA”);

//this is how to construct template function

template<class T1, class T2> //since it is manmade variable we’ll have to use class

void show(T1 x, T2 y)

{

cout<<x<<’\t’<<y;

}

//requirement that the function parameter must have the same number or parameters.

///////////////////////////////////////////////////////////////////////////

We want to sort through an array

//we have to include algorithm library

#include <algorithm> //sort

sortArray(a, 5) //sort array of size 5

sortArray(b, 3)

sortArray(c, 4)

template(class T)

sortArray(T x[], int n)

{

sort (x, x+n)//sort function from algorithm class, asks for array name, and //array size

}