```
#include <iostream>
#include <string> //provides definitions for string header
using namespace std;

        const double HOURLY_RATE_THRESHOLD = 12.00; //all constants must be initialized WITH A VALUE ASSIGNED
        cout.setf(ios::fixed);
        cout.precision(2);

int main()
{
        string personsName;
        string quest;
        int age;

        cout << "What is your name? ";
        getline(cin, personsName); //instead of cin >> personsName;, which only gives the first word of consecutive chars (minus blanks)

        cout << "How old are you? ";
        cin >> age;
        cin.ignore(10000, '\n'); //get rid of all new lines so getline is not satisfied immediately without user input
        //used almost only when you've read in a number, and the next operation is getline

        cout << "What is your quest? ";
        getline(cin, quest);
}
```

```
statement blocks
{
        everything within here is a statement block
        variables instantiated in this block with same name
        as one outside used if referred to (concept of scope)
}

//double:
//      +/- about 10 to the -308 to 10 to the 308
//      about 15 significant digits
//      anything with a dot '.' or 'e'

//int:
//      about -2 billion to 2 billion (historically, ints work must
faster; now, it affects readability; easier or makes more sense)

//identifier:
//      letter: letter/digit/underscore
//      hoursworked
//      hours_worked
//      hoursWorked ("camel case")
//      HoursWorked

//declaration:
//      type variable; (i.e., double doubleNameHere)
//      type variable = expression;

//arithmetic expressions: * / + -
//      (3+4)*(7-2) = 7*5 = 35
//      3+4*5 = 23
//      14.3/5.0 = 2.86
//      14.3/5 = 2.86
//      14/5.0 = 2.8
//      14/5 = 2 (integer result, since both operands are ints)
```

```
to read a number x from the input: cin >> x;
to read a string s from the input: getline(cin, s);
/       flash, forward slash
\       backslash

ints, doubles, floats
identifiers - variable names
case-sensitive
"code paragraphs"
comments

math in c++
        similarity - order of operations
        difference - no implied multiplication, i.e. (1+2)(3+4)

truncation
        int/int
        int a = 5.0/2.0

undefined
        dividing by zero
        overflow

outputting value with specified number of digits after decimal
        cout.setf(ios::fixed);
        cout.precision(2);


"string" type
getline()
pitfall of mixing cin and getline - cin doesn't consume trailing newline

cin.ignore(10000, '\n');

if-else statements

int main()
{
        string whee = "Wheeeeeeeeeeeeeeeeeeeeoooeoooeoeoeoeoeoeoeoeoeeeeeeeeeeeeee!";
        int eCounter = 0;
        int ind=0;

        while(ind <= whee.length()-1)
        {
                if(whee[ind]=='E' || whee[ind]=='e') //same as: if(whee.at(ind)=='e')
                {
                        eCounter++;
                }
                ind++;
        }

        cout << "There are " << eCounter << " e's." << endl;
}
```

```
        int a = 10;
        int b = a*a;
        int c = 25/(b-100);

        double d;
        double e = d*d;
        cout << e;

        int f = 1000;
        int g = f*f*f;
        int h = f*g;

        //undefined behavior!


{ stmt; stmt; stmt; }

compound statement
block
```

```
void makeUpperCase(string& s)
{
        for(int k=0; k!=s.size(); k++)
        {
                s[k]=toupper(s[k]);
        }
}
#include <iostream>
#include <string>
#include <cctype>
using namespace std;

bool isValidPhoneNumber(string pn);
string cleanNumber(string pn);

int main()
{
        cout << "Enter a phone number: ";
        string phone;
        getline(cin, phone);
        if(isValidPhoneNumber(phone))
                cout << "The digits in the number are " << cleanNumber(phone) << endl; //String is copied: "passed by value"
        else
                cout << "A phone number must contain 10 digits." << endl;
}

bool isValidPhoneNumber(string pn)
{
        int numberOfDigits=0;
        for(int k=0; k!=pn.size(); k++)
        {
                if(isdigit(pn[k]))
                        numberOfDigits++;
        }

        return numberOfDigits==10;
}

string cleanNumber(string pn)
{
        string phone="";

        for(int k=0; k!=pn.size(); k++)
        {
                if(isdigit(pn[k]))
                        phone += pn[k]; //concatenation (combine)
        }

        return phone;
}

#include <iostream>
#include <string>

using namespace std;

void flip (string& s)
{
        if(s.size()==0)
                return;
        int b=0;
        int e=s.size()-1;

        while(b!=e && b!=e-1)
        {
                char ch=s.at(b);
                s.at(b)=s.at(e);
                s.at(e)=ch;

                b++;
                e--;
        }
}
```

```
char c = 'A';
int k = 65;
char c2 = 65; //If ASCII is the encoding, this is 'A'
int k2 = 'A' //If ASCII is the encoding, this is 65
k++; //k is now 66
c++; //c is now 66; if ASCII is the encoding, this is 'B'
char d = '9'; //If ASCII is the encording, this is 57
char e = 9; //If ASCII is the encoding, this is '/t'
double x = 3.5;
cout << x; //calls the function for doubles; writes '3' '.' '5'
                            //If ASCII, this is 51      46       53
cout << k; //calls the function for ints; writes '6' '6'
                            //If ASCII, this is 54      54
cout << c; //calls the function for chars; writes 'B'
                            //If ASCII, this is 66
         code for ' 'is less than the code for any printable character
         code for 'A' is less than the code for 'B', 'B' is less than 'C', ...'Z'
         code for 'a' is less than the code for 'b', 'b' is less than 'c', ...'z'
         code for '0' is one less than code for '1', '1' is one less than '2', ...
We CANNOT assume that the codes for alphabetal letters are consecutive; this is only true in ASCII
(a is 1 less than b, which is 1 less than c, etc.)
```

```
"off-by-one-error" - screwing up a loop because the
index is wrong by 1 (OR "fencepost error")

i.e.:

int nTimes;
cin >> nTimes;

int n=0;
while(n<=nTimes)
{
        cout << "hello" << endl;
}

int n=1;

while (n<10)
;       //This program will keep running, since
the semi-colon counts as a "do nothing" under the
while loop; n is never incremented
{
        cout << "Hello" << endl;
        n++;
}
```

```
int longestRun(int a[], int n, int& value)
{
        int lastStreak=1;
        int maxStreak=1;
        int index=0;
        int lastVal = a[0];
        value = a[0];
        while (index<n-1)
        {
                lastStreak=1;
                lastVal = a[index];

                while(a[index]==a[index+1])
                {
                        index++;
                        lastStreak++;
                }
                if(lastStreak>maxStreak)
                {
                        maxStreak=lastStreak;
                        value=lastVal;
                }
                index++;
        }
        return maxStreak;
}
```

```
C strings

char s[100] = ""; //initiates an array with just a "zero-
byte"
char t[9] = {'H', 'e', 'l', 'l', 'o', '\0'}; //ends with
a zero-byte
        //ALTERNATIVE:
        char t[9] = "Hello"; //also adds a zero-byte

cout << t; //prints up-to, but NOT including, the zero-
byte: Hello
cin.getline(s, 100);

t[0] = 'J';

To find the string size in the array:

int strlen(const char a[])
{
        int k;
        for(k=0; a[k]!= '\0'; k++)
                ;
        return k;
}
```

```
#include <cstring> //INCLUDES the strlen function!
s = t;   //Error! Won't compile!
strcpy(s, t); // strcpy(destination, source);
strcpy(t, "asdfjlj jalksdjflasjdflajsdlfjasldfjlsdjf"); /causes a problem!

If at any time, we try to access t[9], we have caused undefined behavior.
```

```
int main()
{
        int data[15] = {5, 8, 8, 2, 7, 7,
7, 7, 8, 8, 8, 3, 3, 3, 3};

        int v;
        int len = longestRun(data, 15, v);
        len = longestRun(data, 5, v);
        len = longestRun(data, 2, v)
}
```