Nathan Tung
CS31 Lec2 DisB

```
#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);
}
```

```
//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)

//      double x = 3.1 + 14/5; //x is 5.1, not 5.9
```

```
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

comparison operators
        <, <=, ==, >, <=, !=

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

Nathan Tung
CS31 Lec2 DisB

```
int main()
{
        string whee = "Wheeeeeeeeeeeeeeeeeeeeeoooeoooeoeoeoeoeoeoeoeoeeeeeeeeeeeeee!";
        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;
}


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;
}
```

```
        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
```

```
"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++;
}
```