Hang onto this worksheet. It will be due no later than October 30, on paper, either in class or under my door.

November 2 - Exam 2

Identifier - a name for a variable, a constant, a function, or a class

Scope - the portion of a program in which an identifier can be used.

1) file scope - applied to global variables and constants, also to function names. The identifier is in scope from its declaration until the end of the file.

2) block scope - an identifier that is in scope from its declaration until the end of the block

   Includes:
   - local variables inside functions
   - parameters of functions
   - variables declared inside loops or other structures

   A block is a portion of code enclosed in { }

3) function prototype scope - applies only to the names of parameters listed in a prototype

   ```
   void printSomething ( int n, int x );
   ```

   n and x are only in scope for this one line of code

4) In C++ we have class scope. Applies to the data members and member functions of a class.

The class definition goes into a file with .h extension.

```
class Student {

    public:
         Student( );
         Student( string, string, int );
         string getFirstName( ) const;
         string getLastName( ) const;
         int getIdNumber( ) const;
         void setFirstName( string );
         void setLastName( string );
         void setIdNumber( int );

// const after the parameter list prevents the
// function from modifying the data members
// of the calling object


   private:
         string firstName;
         string lastName;
         int idNumber;
};
```

By convention we would store this in a file called Student.h or student.h

In another file, typically Student.cpp, we write the function definitions.

```
#include "Student.h"

// constructors
Student::Student( ) {
     lastName = "";
     firstName = "";
     idNumber = 0;
}

Student::Student( string ln, string fn, int id ) {

    // we are overloading the constructor function
    // writing two or more versions with the same
    // name but different parameters

    //  function signature = function name + parameter list

     lastName = ln;
     firstName = fn;
     idNumber = id;
}

// this is not the best idea…

Student::Student ( string lastName, string firstName, int idNumber ) {

   // the 3 parameters have block scope

   // parameters have precedence over the
   // the data members.  Parameters occlude
   // the data members.

   // in C++ this is a pointer to the calling object

   this -> lastName = lastName;
   this -> firstName = firstName;
   this -> idNumber = idNumber;
}

-----------------------------------------

// test the Student class

#include "Student.h"
#include <iostream>
#include <iomanip>
using namespace std;
int main( ) {

    // instantiate a Student object using the
    // the default constructor
    Student john;

    // instantiate a Student object using the
    // other constructor
    Student max( "Power", "Max", 8245 );

    max.setFirstName( "Maxwell" );

}
```

C++ has destructors. A destructor is called automatically when an object goes out of scope.

The compiler provides a default destructor for you.

The name is ~ClassName. ~ is called tilde

A destructor has no parameters, and no return type, not even void.

We need to write a destructor if the data members include some kind of dynamically allocated memory. (for example… you've used malloc)

Prototype for Student destructor.

```
~Student();
```

Function definition for Student destructor.

```
Student::~Student( ) {

    // whatever we want the destructor to do
    // print the memory address of the object that
    // is going out of scope

    cout << "Destroying object at " << this << endl;
}
```

```
for (int i = 0; i < 10; i++) {

    // i has block scope

} // end for

// i is out of scope
```