Today

- Variable declaration
- Mathematical Operators
- Input and Output
- Lab
  - login to the CIRT UNIX machines
  - Start Emacs
  - Write HelloWorld.cpp
  - Compile HelloWorld.cpp

Current Assignments

- Homework 1 due in 6 days (June 16th)

  Variables, mathematical and logical operators, input/output, and the "if" operator.

  (After today and tomorrow's class you should be able to do all but problem 6 on the homework.)

- Project 1 Due in 13 days (June 23rd)

  Write a binomial root solver using the quadratic equation.

Program structure

preprocessor directives

int main()

{

declarations

statements

}

Program structure

```
//preprocessor directives
#include <iostream>
using namespace std;

int main()
{
//variable declarations
int height = 0;
int length = 0;
int area = 0;

//statements
area = height * length;
return 0;
}
```

Comments

- Comments help people read programs, but are ignored by the compiler.
- In C++ there are two types of comments.
  - Line comments begin with // and continue for the rest of the line. (Preferred)
  - Delimited comments begin with /* and end with */
  - /* */ comments are a carry over from C. They can be useful but are also dangerous.

```
/* This is a comment
/* This is a previous comment */ */
```

Preprocessor Commands

- Preprocessor commands provide instructions to the compiler.
- They always start with #
- Example: #include
  - Copies source code into the program from the specified file.
  - #include <iostream> contains class information for input and output.
  - #include <cmath> contains code for many common mathematical functions, such as square root, cosine, and exponentiation.

C++ Data Types

Type	Example values
bool	true
char	‘5’, ‘a’, ‘\n’
int	25
float	25.0
string	“a string value”

(the string data type is defined in <strings>

C++ Data Types

Type	Amount of memory reserved
bool	1 byte
char	typically 1 byte
float	typically 4 bytes
double	≥ float
long double	≥ double
short int	≤ int
int	short int ≤ int ≤ long int
long int	≥ int

There are others… long long, unsigned int, etc

Overflow and Underflow

- Overflow
  - answer too large to store
  - Example: if an “int” is 16 bits the maximum value it can hold will be 2^{16} or 65536

- Exponent overflow and underflow
  - floats consist of a digit portion and an exponent portion
  - If the exponent is to large and positive we call it exponent overflow, if too large and negative then it is underflow

Naming entities in C++

- Identifiers are used to name variables in C ++.
- Rules for construction of identifiers
  - Start with a letter or underscore _
  - Consist of letters digits and underscore
  - Can not be a reserved word.
  - Only first 31 characters used to distinguish it from other identifiers.
  - Case sensitive, r is a different entity from R

Variable Declarations

Declarations define memory locations, including type of data to be stored, identifer, and hopefully an initial value.

General Form:

data_type identifier_list;

Examples:

float length = 20.75, width = 11.5, volume;

int number_of_students = 40;

int number_of_students(40); also works but is much less common.

Symbolic Constants

- Used to name values which do not change during the execution of the program.
- Are always initialized at declaration.
- Used wherever an expression is allowed.

General Form:

const *data_type identifier* **=** *value***;**

Assignment Statements

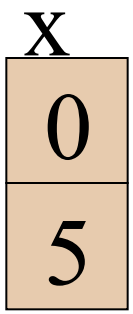
- Used to assign a value to a variable

General Form:

identifier **=** *expression*;

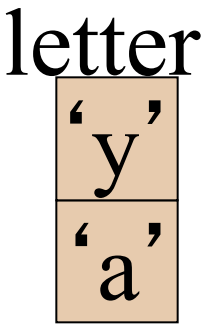
✓ Example 1

```
int x = 0;
x=5;
```



✓ Example 2

```
char letter = 'y';
letter = 'a';
```



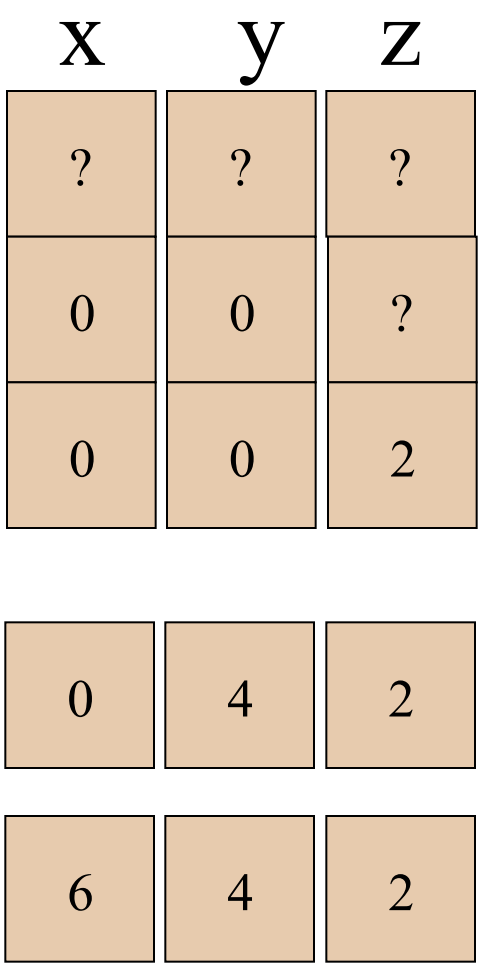
Assignment Statements

- Example 3

```
int x, y, z;
x=y=0;
z=2;

y = 4;

x = y + z;
```



Arithmetic Operators

- Addition **+**
- Subtraction **-**
- Multiplication *****
- Division **/**
- Modulus **%**
  - Modulus returns remainder of division after the second number is divided into the first
  - Examples

    5%2 = 1, 2 goes into 5 twice with a remainder of 1

    2%5 = 2, 5 goes into 2 zero times with a remainder of 2

Integer Division

- Division between two integers results in an integer:
- The result is truncated, not rounded

  5/3 is equal to 1

  3/6 is equal to 0
- Division with an integer and a floating point number results in a floating point number:
  - 5/3.0 = 1.6666
  - 3.0/6 = 0.5

Priority of Operators

1. Parentheses — Inner most first
2. Unary operators — Right to left
   (+ -)
3. Binary operators — Left to right
   (* / %)
4. Binary operators — Left to right
   (+ -)

Increment and Decrement Operators

- Increment Operator **++**
  - post increment **x++;**
  - pre increment **++x;**
- Decrement Operator **- -**
  - post decrement **x- -;**
  - pre decrement **- -x;**
- For examples assume k=5 prior to executing the statement.
  - m= ++k; both m and k become 6
  - n = k- -; n becomes 5 and k becomes 4

Abbreviated Assignment Operator

operator	example	equivalent statement
+=	**x+=2;**	x=x+2;
-=	**x-=2;**	x=x-2;
*=	**x*=y;**	x=x*y;
/=	**x/=y;**	x=x/y;
%=	**x%=y;**	x=x%y;

Precedence of Arithmetic and Assignment Operators		
Precedence	**Operator**	**Associativity**
1	Parentheses: **()**	Innermost first
2	Unary operators **+ - ++ -- (type)**	Right to left
3	Binary operators *** / %**	Left ot right
4	Binary operators **+ -**	Left ot right
5	Assignment operators **= += -= *= /= %=**	Right to left

Simple I/O - **cin**

✓**cin**

- is an istream object
- streams input from standard input
- uses the >> (input operator)

General Form:

cin >> *identifier* >> *identifier***;**

Note: Data entered from the keyboard must be compatible with the data type of the variable.

Simple Output - cout

- **cout**
  - is an ostream object
  - streams output to standard output
  - uses the << (output) operator

General Form:

cout << *expression* **<<** *expression*;

Note: An expression is any C++ expression (string constant, identifier, formula or function call)

```
//Example1 for input and output
#include <iostream>
#include <string>
using namespace std;
int main()
{
    int i, j;
    double x;
    string units = " cm";
    cin >> i >> j;
    cin >> x;
    cout << "output \n";
    cout << i << ',' << j << ',' <<
    endl          << x << units << endl;
    return 0;
}  // Input stream:
1,2,3,4
```

```
output
1,2,
4.5 cm

_
```

```
//Example 2 of input and output
#include <iostream>
using namespace std;
int main()
{       int i, j;
        double x, y;
        cin >> i >> j >> x >> y;
        cout << "First output " << endl;
        cout << i << ',' << j << ',' << x << ',' << y << endl;
        cin >> x >> y >> i >> j;
        cout << "Second output" << endl;
        cout << i << ',' << j << ',' << x << ',' << y << endl;
        return 0;
}  //Input stream is:
1 2
3.4 5
2 3 3.4 7
```

```
First output
1,2,3.4,5
Second output
3,2,2,3
_
```

Manipulators and methods

- endl – places newline character in stream ***and flushes the buffer.***
- setf() and unsetf()

Flag	Meaning
ios:: showpoint	display the decimal point
ios::fixed	fixed decimal notation
ios::scientific	scientific notation
ios::right	right justification
ios::left	left justification

✓ Manipulators in <iomanip>

- setprecision(n)
- setw(n)

Functions in <cmath>

abs(x)	computes absolute value of x
sqrt(x)	computes square root of x, where x >=0
pow(x,y)	computes x^y
ceil(x)	nearest integer larger than x
floor(x)	nearest integer smaller than x
exp(x)	computes e^x
log(x)	computes ln x, where x >0
log10(x)	computes $\log_{10}x$, where x>0
sin(x)	sine of x, where x is in radians
cos(x)	cosine of x, where x is in radians
tan(x)	tangent of x, where x is in radians

Characters and input

- **>> discards leading whitespace**
- Example:

code	*user input*
int x;	
char y;	
cin >> x >> y;	*39 c*
cin >> x;	*42*

Values in memory