Simple Program Format

/***********************************************************************************/
  //
  // Header comments
  //
  // Written by: Bob Rinker
  // Date: 14 - June - 2004
  //
  **********************************************************************************/
#include <iostream>
  // other includes go here
using namespace std;

int main()
{
    // start of the body of the program

    // variable declarations

    // executable statements

} // End main

General C Program Rules

C statements are "free-form":
- Statements can begin and end in any line or column
- Statements can span multiple lines
- Multiple statements can be placed on a single line
- Blank lines are ignored

Comments are ignored by the compiler
- Multiple line comments are enclosed within /* --- */
- Single line comments start with // and continue to end of line

C statements are terminated with ; (semicolon)

White space can be placed around tokens
- White space: blanks, tabs, newlines
- Token: constants, variables, keywords, special characters

C is case sensitive!
- A, B, A, b, a are all different
A Simple First Program

/****************************************************************************
 *  This program does some simple calculations
 *  Written by: Bob Rinker
 *  ******************************************************/
#include <iostream>
using namespace std;

int main() { // Alternate format
  int a, b, c, d;
  cout << "Input values for a and b: ";
  cin >> a >> b;
  c = a + b;
  d = a - b;
  cout << "You input " << a << " for a and " << b << " for b " << endl << " The sum is " << c << " and the difference is " << d << endl;
} // END main

First Program Output

Input values for a and b: 2 4
You input 2 for a and 4 for b
The sum is 6 and the difference is -2
Keywords in C

Keywords are composed of lower case letters. Keywords are reserved – they cannot be used as variable or function names.

### C and C++
- auto
- break
- case
- char
- continue
- const
- default
- do
- double
- else
- enum
- extern
- float
- for
- goto
- if
- int
- long
- register
- return
- short
- signed
- sizeof
- static
- struct
- switch
- typedef
- union
- void
- volatile
- while

### C++ Only
- asm
- catch
- class
- delete
- friend
- inline
- new
- operator
- private
- protected
- public
- template
- this
- throw
- try
- virtual

---

C Data Types

- **Int** – Integer (whole numbers).
  
  - 123
  - -27
  - 0x123 (Hexadecimal = base 16)

- **float** – Fractional values. Sometimes called "real"
  
  - 123.0
  - -27.
  - 10.75
  - 0x8271.0 (representable on all systems)
  - 0.0000000000000000000013
  - 1.3e-20 ("exponential notation")

- **char** – A single char, different from a string
  
  - 'x'
  - 'Y'
Some Other Types

C has some additional types that are different in size from the basic types:

- long — (or long int) — usually longer (uses more bits) than a standard int, can store larger numbers than int
- short — (or short int) — usually smaller than standard int, saves some memory space but can’t store large numbers
- double — a larger version of float, can usually store larger numbers, more accuracy than float, but uses more space

The C standard doesn’t specify sizes of types — it varies with computer/compiler type.

Variable Declarations in C

All variables must be declared before being used in a program.

The declaration can also be used to define (initialize) the variable.

```c
int lower, upper, step;
float fahr, celsius;
int i = 0;
char set = ‘e’, ans;
```

*Style point — declarations of all variables usually done at the beginning of the body of a program.*
Variable Names

Variable names have the following rules:
- must start with a letter (some systems also allow $)
- other characters can be upper/lower alpha, numbers, and a few other characters (__, $)
- explicitly cannot contain operators or white space
- some systems impose a maximum length, others only use a certain number of chars and ignore the rest.

<table>
<thead>
<tr>
<th>LEGAL</th>
<th>NOT LEGAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>abc</td>
<td>1bc</td>
</tr>
<tr>
<td>Abc</td>
<td>.com</td>
</tr>
<tr>
<td>abc123</td>
<td>a&amp;b</td>
</tr>
<tr>
<td>sum_of_items</td>
<td>sum of items</td>
</tr>
<tr>
<td>SumOfItems</td>
<td></td>
</tr>
</tbody>
</table>

Constants in C

It is sometimes useful to declare constant values in our programs:

    const int MAX = 10;
    const float PI=3.1415;
    const char YES = 'y';

_style point – it is conventional to use capital letters for constants declared in a program._
Arithmetic Operators

There are many operators in C. These are the ones called arithmetic operators.

+ Addition
- Subtraction
* Multiplication
/ Division
% Modulus (remainder – integer only)

Arithmetic Expressions

Expressions are composed of constants, variables and operators. C evaluates expressions.

```c
int a = 10, b = 20;
float x = 2.0, y = 0.5;

a + b
x + 3.0
a+b*3
x / y
a / b
```
Operator Hierarchy (Simplified)

The compiler performs arithmetic operations in a specific order:

- (negation) is done first
- * (multiplication), / (division), % (mod) are done next, left to right
- Then + (addition), - (subtraction) are done next, left to right
- Then = (assignment) is done, right to left

All operators we will learn will be added to this hierarchy.

Changing the Hierarchy

Parentheses can be used to change the order of operations. Expressions in () are evaluated first

\[
\begin{align*}
\text{1st} & \quad a + b * c \\
\text{2nd} & \quad (a + b) * c \\
\text{3rd} & \quad ((3a + 1) * (y + 4x)) / (3 + z)
\end{align*}
\]

"When in doubt, use parentheses"
int Arithmetic

Ints can only store whole numbers. When performing calculations with ints, any fractional results are truncated!

\[
\begin{align*}
  4 \div 3 & \times 5 \\
  1 & \div 5 \\
  1 \div 2 & = 28 \\
  0 & = 0
\end{align*}
\]

Compare the last result with this:

\[
\begin{align*}
  1.0 \div 2.0 & = 28.0 \\
  0.5 & = 14.0
\end{align*}
\]

Be careful! Choose your types carefully!

---

Mixed Type Expressions

C determines the result type from the types of the operands. It uses int arithmetic if the operands are ints, float arithmetic if the operands are floats.

For mixed expressions, it uses the following rules:

- If either operand is double, convert the other to double
- Otherwise, if either operand is float, convert the other to float
- Otherwise, convert char and short to int
- Then, if either operand is long, convert the other to long
Assignment

Assignment is designated by the = sign (i.e., = is an operator)

The variable on the left is assigned the value of the expression on the right.

```c
int a = 10, b = 20;
float x = 2.0, y = 0.5;
int c, d;
float z, v, w;
c = a + b;
z = x + 3.0;
d = (a+b)*3;
v = x / y;
w = a / b;
a = a + 1;
```

Library Functions

Some operations are impossible or very difficult to calculate using just operators. Some of these operations can be performed using library functions.

\[
\text{roots} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
\]

\[
r1 = (-b + \sqrt{\text{pow}(b,2) - 4.0 * a * c}) / (2.0 * a);
\]

\[
r2 = (-b - \sqrt{\text{pow}(b,2) - 4.0 * a * c}) / (2.0 * a);
\]

You must #include the correct library header file. For the above example, you need to #include `<cmath>`
Character Values

Computers store characters as small numbers (bytes). The most common encoding for chars is called ASCII.

Some "control" (non printing) chars have special designations (can also be used in strings):

'\t' - tab character
'\n' - newline
'"' - the ' character
'\'' - the " character
'\' - the backslash

'\x41' - the char whose ASCII value is 41 hex ('A')

"\x7 Make \x7 up!!! \x7" - print this string!
Assignment Operators

Shortcuts for writing assignments:

```c
a += 10;  // Same as a = a + 10;
b -= 3;   // Same as b = b - 3;
x *= y;  // Same as x = x * y;
r /= 5;  // Same as r = r / 5;
```

---

C++ Input/Output

C++ uses the "iostream class" to do I/O

Strictly speaking, I/O is not a part of the C++ language, but rather is implemented as functions and operators that allow us to do I/O.

Original C used some different functions (printf and scanf) to do I/O.
"Standard Input/Output"

The cin and cout functions are connected to "standard input" and "standard output." Normally, this means the keyboard and screen, but is actually operating system dependent.

UNIX (and DOS) allows "redirection" of the standard input and output:

```plaintext
e.out < infile

e.out > outfile
```

---

The iostream class

- cin, cout, and endl are functions ("methods") of the iostream class.

- >> (for cin) and << (for cout) are operators defined by the iostream class.

- Classes exist to do many useful things.

- Eventually, we will learn how to write our own classes.