Simple Scalar Data Types (32-bit compilers)

Commonly used

`int`	**Integer** in the range -2,147,483,648 to 2,147,483,647	
`double`	Floating-point **decimal** in the range $\pm 1.7 \times 10^{308}$ to $\pm 1.7 \times 10^{308}$	15-digit precision
`char`	Character (think: **letter**)	
`bool`	**Boolean**: true (1) or false (0)	
`string`	A series of char variables (think: **word**)	

Variable names should be descriptive of their use. They should consist of letters and/or numbers. They are case sensitive and must start with a letter. They cannot contain a space but can use an underscore (_).

Mathematical Operators

(some operators omitted)

+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Unary increment
--	Unary decrement

Logical Operators

&&	and
||	or
!	not

Order of Operations

(some operators omitted)

Higher	!, ++, --
2	*, /, %
3	+, -
4	==, !=
5	&&
6	||
Lower	=

Input and Output

The `cin` and `cout` objects provide high-level input and output in C++, respectively. `cin`, which verifies that input is of the correct type, is used with the extraction operator (>>):

```
cin >> x >> y >> z >> ch;
```

`cout` provides output functionality by means of the insertion (<<) operator:

```
cout << "This is a number: " << n;
```

Formatting Output and Input with the iomanip Library

*Requires the following line at the beginning of your code `#include <iomanip>`

Function	Computes
`setbase(int base)`	Sets *basefield* to hex, dec or oct depending on *base* parameter.
`setfill(char ch)`	Fill the white space with character `ch`
`setprecision(int n)`	Set decimal precision to `n` places
`setw(int w)`	Sets a value to be used as the *field width* (w) for the next insertion operation.
`resetiosflags(flag)`	Unsets the format flags specified by parameter.
`setiosflags(flag)`	Sets the format flags specified by parameter.

ios::base flags options for setiosflags and resetiosflags

Function	Computes
`ios_base::boolalpha`	input/output **bool** objects as alphabetic names (**true**, **false**).
`ios_base::fixed`	output floating point values in fixed-point notation.
`ios_base::left`	the output is filled at the end enlarging the output up to the *field width*.
`ios_base::right`	the output is filled at the beginning enlarging the output up to the *field width*.
`ios_base::scientific`	output floating-point values in scientific notation.
`ios_base::showpoint`	output floating-point values including always the decimal point.
`ios_base::showpos`	output non-negative numeric preceded by a plus sign (+).
`ios_base::uppercase`	output uppercase letters replacing certain lowercase letters.

Examples

Output is unformatted by default but may be modified by using the manipulators found in `iomanip`:

```
cout << setiosflags(ios_base::fixed)       // do not use E notation
     << setiosflags(ios_base::showpoint)   // always show decimal pt
     << setprecision(2);                 // round to two decimal places
```

Once used, the manipulators above "stick" for the rest of the program—that is, they apply to all numeric output thereafter unless the `resetiosflags` manipulator is called.

To output a number right aligned in a certain size field, use the `setw` manipulator. They apply to only the next output.

```
cout << setw(10)
     << number1
     << number2;
```

Ten columns are allocated for the next output by the `setw(10)` manipulator. Number 1 would appear right aligned within those 10 columns and number2 would appear immediately to the right of number1 (no spaces between them).