CS111

Introduction to Computing Science

Today

- Variables
- Assignments
- Constants
- Comments

Variables

- A variable
  - is used to store information:
    - can contain one piece of information at a time.
  - has an identifier

The content of a variable

The name of a variable

The programmer picks a good name

- A good name describes the contents of the variable or what the variable will be used for

Variables

Parking garages store cars.

Variables

Each parking space is identified – like a variable's identifier

A each parking space in a garage "contains" a car – like a variable's current contents.

Variables

and

each space can contain only *one* car

and

***only* cars, not buses or trucks**

Variable Declarations

- Creating a variable is called **declaring a variable.**
- When declaring variables, the programmer specifies the **type** of information to be stored. (more on types later)
- The compiler will set aside that space in memory and give it the name we choose.

Variable Declarations

- Unlike a parking space, a variable is often given an initial value.
  - *Initialization* is putting a value into a variable when the variable is created.
  - Initialization is not required.

Variable Definitions

SYNTAX 2.1 Variable Definition

```
int cans_per_pack = 6;
```

Must obey the rules for valid names

Types introduced in this chapter are the number types `int` and `double` and the `string` type

A variable definition ends with a semicolon.

Supplying an initial value is optional, but it is usually a good idea.

Use a descriptive variable name.

Variable Definitions

Table 1 Variable Definitions in C++

Variable Name	Comment
`int cans = 6;`	Defines an integer variable and initializes it with 6.
`int total = cans + bottles;`	The initial value need not be a constant. (Of course, `cans` and `bottles` must have been previously defined.)
`int bottles = "10";`	**Error:** You cannot initialize a number with a string.
`int bottles;`	Defines an integer variable without initializing it. This can be a cause for errors—see Common Error 2.2 on page 37.
`int cans, bottles;`	Defines two integer variables in a single statement. In this book, we will define each variable in a separate statement.
`bottles = 1;`	**Caution:** The type is missing. This statement is not a definition but an assignment of a new value to an existing variable—see Section 2.1.4 on page 34.

Data Types

There are different **types** of data.

Every variable contains only data of one **type**.

Common **basic types**

- **`int`** Integers ….,-1,0,1,2, ….
- **`double`** Floating point numbers 1.0, 3.14, -2.1
- **`char`** characters a,b,c,…, A,B,C,…,!,@,#,$,...

More about data types later.

C++ Variables

Variables are containers for storing data values.

In C++, there are different **types** of variables (defined with different keywords), for example:

- int - stores integers (whole numbers), without decimals, such as 123 or -123

  Eg:

  ```
  int myNum = -10;    // Integer (whole number without decimals)
  ```

- double - stores floating point numbers, with decimals, such as 19.99 or -19.99

  Eg:

  ```
  double myFloatNum = 25.99;    // Floating point number (with decimals)
  ```

- char - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes

  Eg:

  ```
  char myLetter = 'D';       // Character
  ```

- string - stores text, such as "Hello World". String values are surrounded by double quotes

  ```
  string myText = "Hello";     // String (text)
  ```

- bool - stores values with two states: true or false

  ```
  bool myBoolean = true;       // Boolean (true or false)
  ```

Variable Names

- When you define a variable, you should pick a name that explains its purpose.
- For example, it is better to use a descriptive name, such as **`can_volume`**, than a terse name, such as **`cv`**.
- You do this for yourself, and your colleagues.

Variable Names

- In C++, there are a few strict rules for variable names.
- The compiler will enforce these rules.
- This means, if you do not follow them, your program will not compile.

Variable Names

1. Variable names must start with a letter or the underscore (_) character, and the remaining characters must be letters numbers, or underscores.
2. You cannot use special symbols. Spaces are not permitted inside names; you can use an underscore instead, as in can_volume.
3. Variable names are case-sensitive, that is, can_volume and can_Volume are different names.
   For that reason, it is a good idea to use only lowercase letters in variable names.
4. You cannot use reserved words such as double or return as names; these words are reserved exclusively for their special C++ meanings.

Variable Names

Table 3 Variable Names in C++

Variable Name	Comment
`can_volume1`	Variable names consist of letters, numbers, and the underscore character.
`x`	In mathematics, you use short variable names such as *x* or *y*. This is legal in C++, but not very common, because it can make programs harder to understand (see Programming Tip 2.1 on page 38).
⚠ `Can_volume`	**Caution:** Variable names are case-sensitive. This variable name is different from `can_volume`.
⊘ `6pack`	**Error:** Variable names cannot start with a number.
⊘ `can volume`	**Error:** Variable names cannot contain spaces.
⊘ `double`	**Error:** You cannot use a reserved word as a variable name.
⊘ `ltr/fl.oz`	**Error:** You cannot use symbols such as / or .

Number Types

A number written by a programmer is called a **number literal.**

There are rules for writing literal values:

Number Types

Number	Type	Comment
6	`int`	An integer has no fractional part.
–6	`int`	Integers can be negative.
0	`int`	Zero is an integer.
0.5	`double`	A number with a fractional part has type `double`.
1.0	`double`	An integer with a fractional part .0 has type `double`.
1E6	`double`	A number in exponential notation: 1×10^6 or 1000000. Numbers in exponential notation always have type `double`.
2.96E-2	`double`	Negative exponent: $2.96 \times 10^{-2} =$ 2.96 / 100 = 0.0296
100,000		**Error**: Do not use a comma as a decimal separator.
3 1/2		**Error**: Do not use fractions; use decimal notation: 3.5

Lessons so far

A variable declaration (aka variable definition)

- Starts with the type (`int, double, char`,...)
- Then come the variable name. Choose a good name.
- Then you can set an initial value. This is optional.

Lessons so far

Literals used in C++ have a type, too. There is for example a difference between

- the integer 2
- the floating point number 2.0
- the character ‘2’.

The Assignment Statement

- The contents in variables can “vary” over time (hence the name!).

- Variables can be changed by
  - assigning to them
    - The assignment statement
  - using the increment or decrement operator
  - inputting into them
    - The input statement

The Assignment Statement

- An *assignment statement*

  stores a new value in a variable,
  replacing the previously stored value.

The Assignment Statement

```
cans_per_pack = 8;
```

This assignment statement changes the value stored in **`cans_per_pack`** to be 8.

The previous value is replaced.

The Assignment Statement

SYNTAX 2.2 Assignment

```
double total = 0;
   .
   .
total = bottles * BOTTLE_VOLUME;
   .
   .
   .
total = total + cans * CAN_VOLUME;
```

This is an initialization of a new variable, NOT an assignment.

This is an assignment.

The name of a previously defined variable

The expression that replaces the previous value

The same name can occur on both sides.

The Assignment Statement

- There is an important difference between a variable definition and an assignment statement:

```
int cans_per_pack = 6; // Variable definition
...
cans_per_pack = 8; // Assignment statement
```

- The first statement is a *declaration* of **`cans_per_pack`**.
- The second statement is an *assignment statement.* An *existing* variable's contents are replaced.

The Assignment Statement

- The = in an assignment does ***not*** mean the left hand side is equal to the right hand side as it does in math.
- = is an instruction to do something:

  copy the value of the expression on the right ***into*** the variable on the left.

- Consider what it would mean, mathematically, to state:

counter = counter + 2 *;*

counter *EQUALS* counter + 1 ?

The Assignment Statement

```
counter = 11; // set counter to 11
counter = counter + 2; // increment
```

The Assignment Statement

```
counter = 11; // set counter to 11
counter = counter + 2; // increment
```

1. Look up what is currently in counter (11)

The Assignment Statement

```
counter = 11; // set counter to 11
counter = counter + 2; // increment
```

1. **Look up what is currently in counter (11)**
2. **Add 2 to that value (13)**

The Assignment Statement

```
counter = 11; // set counter to 11
counter = counter + 2; // increment
```

1. Look up what is currently in counter (11)
2. Add 2 to that value (13)
3. *copy* the result of the addition expression *into* the variable on the left, changing counter

The Assignment Statement

```
counter = 11; // set counter to 11
counter = counter + 2; // increment
```

1. Look up what is currently in counter (11)
2. Add 2 to that value (13)
3. *copy* the result of the addition expression *into* the variable on the left, changing counter

```
cout << counter << endl;
```

```
13 is shown
```

Constants

- Sometimes the programmer knows certain values just from analyzing the problem, for this kind of information, programmers use the reserved word **`const`**.
- The reserved word **`const`** is used to define a constant.
- A **`const`** is a variable whose contents cannot be changed and must be set when created.
  (Most programmers just call them constants, not variables.)
- Constants are commonly written using capital letters to distinguish them visually from regular variables:

```
const double BOTTLE_VOLUME = 2;
```

Constants

Another good reason for using constants:

```
double volume = bottles * 2;
```

What does that 2 mean?

Constants

If we use a constant there is no question:

```
double volume = bottles * BOTTLE_VOLUME;
```

Any questions?

Constants

And still another good reason for using constants:

```
double bottle_volume = bottles * 2;
double can_volume = cans * 2;
```

What does *that* 2 mean?

— *WHICH 2?*

That 2

is called a "***magic number***"

(so is that one)

because it would require magic to know what 2 means.

It is not good programming practice to use magic numbers. Use constants.

Constants

- Rules of thumb on Magic Numbers
  - Avoid using literal numbers like 2.0 or 45.
  - Define a constant instead and use that constant.
  - Exceptions are using 0 (zero) or 1.
    - It is usually obvious what these stand for.
    - You can use them on your program.
    - No need to define a constant for 0 or 1.

Constants

And it can get even worse …

Suppose that the number 2 appears hundreds of times throughout a five-hundred-line program?

Now we need to change the BOTTLE_VOLUME to 2.23 (because we are now using a bottle with a different shape)

How to change ***only*** some of those magic numbers 2's?

Constants

Constants to the rescue!

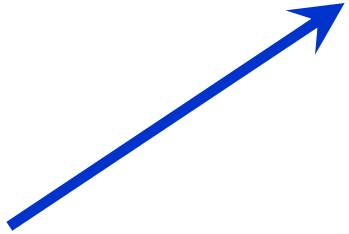
```
const double BOTTLE_VOLUME = 2.23;
const double CAN_VOLUME = 2;

...

double bottle_volume = bottles *
  BOTTLE_VOLUME;
double can_volume = cans * CAN_VOLUME;
```

(Look, no magic numbers!)

Comments

- *Comments*
  are explanations for human readers of your code (other programmers).
- The compiler ignores comments completely.

```
double can_volume = 0.355; // Liters in a 12-ounce can
```



Comment

Comments

```
double can_volume = 0.355; // Liters in a 12-ounce can
```

```
This just in…

The number of liters
in a twelve ounce can
is 355 one hundredths

This newsbreak brought
to you by Cay's Cans Corp.
```

Comments

Comments can be written in two styles:

- Single line:

```
double can_volume = 0.355; // Liters in a 12-ounce
   can
```

The compiler ignores everything after `//` to the end of line

- Multiline for longer comments:

```
/*
   This program computes the volume (in liters)
   of a six-pack of soda cans.
*/
```

Common Error –Undefined Variables

- You must define a variable before you use it for the first time.
- For example, the following sequence of statements would not be legal:

```
double can_volume = 12 * liter_per_ounce;
double liter_per_ounce = 0.0296;
```

What is the value of `can_volume` at the end?

Common Error –Undefined Variables

What is the value of `can_volume` at the end?

```
double can_volume = 12 * liter_per_ounce;
double liter_per_ounce = 0.0296;
```

? ?

- Statements are compiled in top to bottom order.
- When the compiler reaches the first statement, it does not know that **`liter_per_ounce`** will be defined in the next line, and it reports an error.

Common Error –Uninitialized Variables

- Initializing a variable is not required, but there is always a value in every variable, even uninitialized ones.
- Some value will be there, the flotsam left over from some previous calculation or simply the random value there when the transistors in RAM were first turned on.

```
int bottles; // Forgot to initialize
int bottle_volume = bottles * 2;
                                // Result is unpredictable
```

What value would be output from the following statement?

```
cout << bottle_volume << endl;
```

Notice All the Issues Covered So Far

```
/*
This program computes the volume (in liters) of a six-pack of soda
cans and the total volume of a six-pack and a two-liter bottle.
*/

int main()
{
   int cans_per_pack = 6;
   const double CAN_VOLUME = 0.355; // Liters in a 12-ounce can
   double total_volume = cans_per_pack * CAN_VOLUME;

   cout << "A six-pack of 12-ounce cans contains "
      << total_volume << " liters." << endl;

   const double BOTTLE_VOLUME = 2; // Two-liter bottle

   total_volume = total_volume + BOTTLE_VOLUME;

   cout << "A six-pack and a two-liter bottle contain "
      << total_volume << " liters." << endl;

   return 0;
}
```

Quiz

What is **standard output**?

What is **standard input**?

Quiz

What is an escape sequence?

Quiz

How do you print to standard output:

```
The emoticon for `looking around' is \\(<.<|>.>)//
```

Quiz

- What is the difference between *variables* and *constants*

Quiz

What would be a good variable name for the number of students in a tutorial?

Quiz

What would be a good name for the constant: maximal number of students in a tutorial?

Quiz

What is a magic number? Should you have them?

Quiz

What happens if you use a variable that has not been initialized?

Quiz

What happens if you use a variable that has not been declared?

Quiz

What do you use comments for?