Statements and expressions

Victor Eijkhout, Susan Lindsey

Fall 2023

last formatted: February 6, 2024

Basics

1. Two kinds of files

In programming you have two kinds of files:

- *source files*, which are understandable to you, and which you create with an editor such as `vi` or `emacs`; and
- *binary files*, which are understandable to the computer, and unreadable to you.

Your source files are translated to binary by a compiler, which 'compiles' your source file.

Exercise 1

Make a file `zero.cc` with the following lines:

```
// basic/null.cpp
#include <iostream>
using std::cout;

int main() {
  return 0;
}
```

and compile it. Intel compiler:

```
icpc -o zeroprogram zero.cc
```

Run this program (it gives no output):

```
./zeroprogram
```

2. Anatomy of the compile line

- `icpc` : compiler. Alternative: use `g++` or `clang++`
- `-o zeroprogram` : output into a binary name of your choosing
- `zero.cc` : your source file.

Exercise 2

Add this line:

```
// basic/hello.cpp
cout << "Hello world!" << '\n';
```

(copying from the pdf file is dangerous! please type it yourself)

Compile and run again. What is the output?

3. C++ versions

- The compiler by default uses `C++98`.
- This course explains `C++17` and `C++20`. You need tell your compiler about this.
- On `isp.tacc.utexas.edu` 'icpc' uses this by default.
- On your own machine you need to do

  ```
  g++ -std=c++17 [other options]
  ```

  or

  ```
  alias g++='g++ -std=c++17'
  ```

  (in your `.bashrc` file)

Review quiz 1

True or false?

1. The programmer only writes source files, no binaries.
   `/poll "A programmer writes sources, no binaries" "T" "F"`
2. The computer only executes binary files, no human-readable files.
   `/poll "Computer executes binaries, no readable files" "T" "F"`

4. Error types

Your code may suffer from the following types of error:

1. *Syntax* or *compile-time* errors: these arise if what you write is not according to the language specification. The compiler catches these errors, and it refuses to produce a binary file.
2. *Run-time* errors: these arise if your code is syntactically correct, and the compiler has produced an executable, but the program does not behave the way you intended or foresaw. Examples are divide-by-zero or indexing outside the bounds of an array.
3. Design errors: your program does not do what you think it does.

5. File names

File names can have extensions: the part after the dot. (The part before the dot is completely up to you.)

- `program.cpp` or `program.cc` or `program.cxx` are typical extensions for C++ sources.
- `program.cpp` has a possible possible confusion with 'C PreProcessor', but it seems to be the standard, so we will use it in this course.
- Using `program` without extension usually indicates an executable. (What happens if you leave the `-o myprogram` part off the compile line?)

Statements

6. Program statements

- A program contains statements, each terminated by a semicolon.
- 'Curly braces' can enclose multiple statements.
- A statement can correspond to some action when the program is executed.
- Some statements are definitions, of data or of possible actions.
- Comments are 'Note to self', short:

```
cout << "Hello world" << '\n'; // say hi!
```

and arbitrary:

```
cout << /* we are now going
           to say hello
        */ "Hello!" << /* with newline: */ '\n';
```

Exercise 3

Take the 'hello world' program you wrote above, and duplicate the hello-line. Compile and run.

Does it make a difference whether you have the two hellos on the same line of your file, or on different lines?

Experiment with other changes to the layout of your source. Find at least one change that leads to a compiler error. Can you relate the message to the error?

7. Fixed elements

You see that certain parts of your program are inviolable:

- There are keywords such as `return` or `cout`; you can not change their definition.
- Curly braces and parentheses need to be matched.
- There has to be a *main* keyword.
- The *iostream* and `std` are usually needed.

Exercise 4

Experiment with the `cout` statement. Replace the string by a number or a mathematical expression. Can you guess how to print more than one thing, for instance:

- the string `One third is`, and
- the result of the computation $1/3$,

with the same `cout` statement? Do you get anything unexpected?

Review quiz 2

True or false?

- If your program compiles correctly, it is correct.
  `/poll "If it compiles it is correct" "T" "F"`
- If you run your program and you get the right output, it is correct.
  `/poll "Program runs, no errors, therefore it is correct" "T" "F"`

Variables

8. What's a variable?

Programs usually contain data, which is stored in a variable.
A variable has

- a datatype,
- a name, and
- a value.

These are defined in a variable declaration and/or variable assignment.

9. Example variable lifetime

```
int i,j; // declaration
i = 5;   // set a value
i = 6;   // set a new value
j = i+1; // use the value of i
i = 8; // change the value of i
       // but this doesn't affect j:
       // it is still 7.
```

10. Variable names

- A variable name has to start with a letter;
- a name can contains letters and digits, but not most special characters, except for the underscore.
- For letters it matters whether you use upper or lowercase: the language is case sensitive.
- Words such as `main` or `return` are reserved words.
- Usually `i` and `j` are not the best variable names: use `row` and `column`, or other meaningful names, instead.
- While you can start a name with an underscore, there are some limitations on the use of the underscore: do not use two underscores in a row, and do not start a name with an underscore followed by a capital letter.

11. Declaration

A variable declaration establishes the name and the type of a variable:

```
int n_elements;
float value;
int row,col;
double re_part,im_part;
```

12. Where do declarations go?

Declarations can go pretty much anywhere in your program, but they need to come before the first use of the variable.

Note: it is legal to define a variable before the main program but such global variables are usually not a good idea. Please only declare variables *inside* main (or inside a function et cetera).

Review quiz 3

Which of the following are legal variable names?

1. `mainprogram`
   `/poll "Legal mainprogram?" "T" "F"`
2. `main`
   `/poll "Legal main?" "T" "F"`
3. `Main`
   `/poll "Legal Main?" "T" "F"`
4. `1forall`
   `/poll "Legal 1forall?" "T" "F"`
5. `one4all`
   `/poll "Legal one4all?" "T" "F"`
6. `one_for_all`
   `/poll "Legal one_for_all?" "T" "F"`
7. `onefor{all}`
   `/poll "Legal onefor{all}?" "T" "F"`

13. Datatypes

Variables come in different types;

- We call a variable of type `int`, `float`, `double` a numerical variable.
- *Complex numbers* will be discussed later.
- For characters: `char`. Strings are complicated; see later.
- Truth values: `bool`
- You can make your own types. Later.

Assignments

14. Assignment

Once you have declared a variable, you need to establish a value. This is done in an assignment statement. After the above declarations, the following are legitimate assignments:

```
n = 3;
x = 1.5 - n;
n1 = 7;
n2 = n1 * 3;
```

These are not math equations: the variable on the left hand side gets the value of the expression on the right hand side.

You see that you can assign both a simple value or an expression.

15. Special forms

Certain assignments with the same variable in both the left and right hand sides can be simplified:

```
x = x+2; y = y/3;
// can be written as
x += 2; y /= 3;
```

Integer add/subtract one:

```
i=i+1; j=j-1;
// rewritten as:
++i; --j;
// or
i++; j--;
```

Review quiz 4

Which of the following are legal? If they are, what is their meaning?

1. `n = n;`
   ```
   /poll "Legal 'n = n;'?" "T" "F"
   ```
2. `n = 2n;`
   ```
   /poll "Legal 'n = 2n;'?" "T" "F"
   ```
3. `n = n2;`
   ```
   /poll "Legal 'n = n2;'?" "T" "F"
   ```
4. `n = 2*k;`
   ```
   /poll "Legal 'n = 2*k;'?" "T" "F"
   ```
5. `n/2 = k;`
   ```
   /poll "Legal 'n/2 = k;'?" "T" "F"
   ```
6. `n /= k;`
   ```
   /poll "Legal 'n /= k;'?" "T" "F"
   ```

16. Initialization syntax

There are (at least) two ways of initializing a variable

```
int i = 5;
int j{6};
```

Note that writing

```
int i;
i = 7;
```

is not an initialization: it's a declaration followed by an assignment.

Review quiz 5

```
#include <iostream>
using std::cout;
int main() {
  int i;
  int j = i+1;
  cout << j << "\n";
  return 0;
}
```

What happens?

1. Compiler error
2. Output: `1`
3. Output is undefined
4. Error message during running the program.

```
/poll "What happens:" "Compiler error" "Output: 1" "Output undefined" "Runtime error"
```

17. Floating point constants

- Default: `double`
- Float: `3.14f` or `3.14F`
- Long double: `1.22l` or `1.22L`.

18. Warning: floating point arithmetic

Floating point arithmetic is full of pitfalls.

- Don't count on `3*(1./3)` being exactly 1.
- Not even associative.

19. Truth values

So far you have seen integer and real variables. There are also boolean values which represent truth values. There are only two values: `true` and `false`.

```
bool found{false};
found = true;
```

Input/Output

20. Terminal output

Terminal (console) output with `cout`:

```
float x = 5;
cout << "Here is the root: " << sqrt(x) << '\n';
```

Note the newline character.
Alternatively: `std::endl`, less efficient.

21. Terminal input

```
// basic/cin.cpp
string name; int age;
cout << "Your name?\n";
cin >> name;
cout << "age?\n";
cin >> age;
cout << age << " is a nice
    age, "
     << name << '\n';
```

```
> ./cin
Your name?
Victor
age?
18
18 is a nice age, Victor
> ./cin
Your name?
THX 1138
age?
1138 is a nice age, THX
```

22. Quick intro to strings

- Add the following at the top of your file:

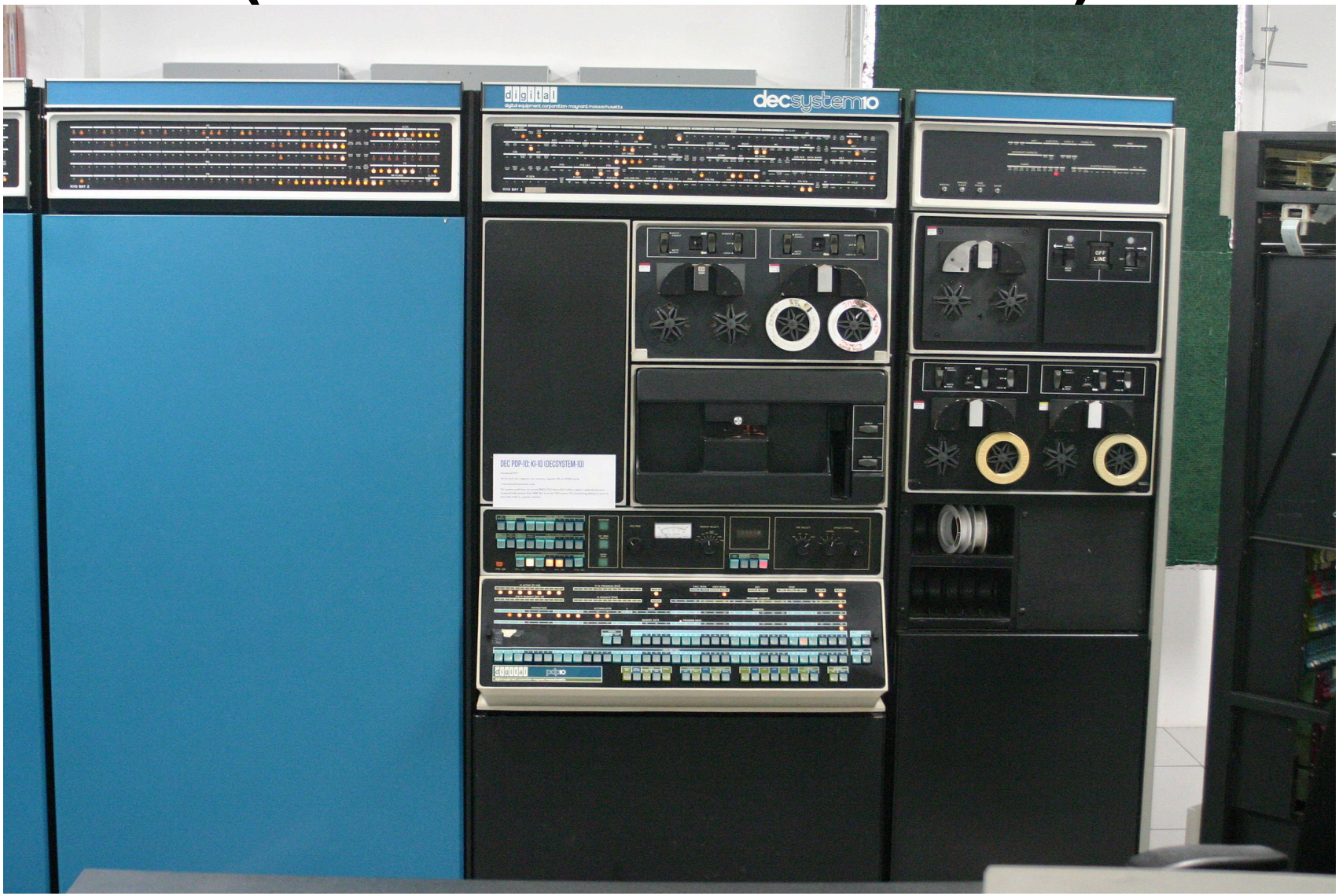
```
#include <string>
using std::string;
```

- Declare string variables as

```
string name;
```

- And you can now `cin` and `cout` them.

(Just what *is* a console?)



Exercise 5

Write a program that asks for the user's first name, uses `cin` to read that, and prints something like `Hello, Susan!` in response.

What happens if you enter first and last name?

Expressions

23. Arithmetic expressions

- Expression looks pretty much like in math.
  With integers: `2+3`
  with reals: `3.2/7`
- Use parentheses to group `25.1*(37+42/3.)`
- Careful with types.
- There is no 'power' operator: library functions.
- Modulus: `%`

24. Math library calls

Math functions are in `cmath`:

```
#include <cmath>
.....
x = pow(3,.5);
```

For squaring, usually better to write `x*x` than `pow(x,2)`.

Boolean expressions

We'll do that in the lecture on conditionals.

Exercise 6

Write a program that :

- displays the message `Type a number`,
- accepts an integer number from you (use `cin`),
- makes another variable that is three times that integer plus one,
- and then prints out the second variable.

Turn it in!

- If you have compiled your program, do:

  `coe_3np1 yourprogram.cc`

  where '`yourprogram.cc`' stands for the name of your source file.
- Is it reporting that your program is correct? If so, do:

  `coe_3np1 -s yourprogram.cc`

  where the `-s` flag stands for 'submit'.

Note: this will send your file to the instructors with a **time stamp**. If you submit again after the deadline, you will be recorded as a late submission.

25. Conversion and casting

Real to integer: round down:

```
double x,y; x = .... ; y = .... ;
int i; i =  x+y:
```

Dangerous:

```
int i,j; i = ... ; j = ... ;
double x ; x = 1+i/j;
```

The fraction is executed as integer division.
For floating point result do:

```
(1.*i)/j
```

Optional exercise 7

Write a program that asks for two integer numbers `n1,n2`.

- Assign the integer ratio n_1/n_2 to an integer variable.
- Can you use this variable to compute the modulus

$$n_1 \mod n_2$$

  (without using the % modulus operator!)
  Print out the value you get.
- Also print out the result from using the modulus operator:%.
- Investigate the behavior of your program for negative inputs.
  Do you get what you were expecting?

Optional exercise 8

Write two programs, one that reads a temperature in Centigrade and converts to Fahrenheit, and one that does the opposite conversion.

$$C = (F - 32) \cdot 5/9, \qquad F = 9/5\, C + 32$$

Check your program for the freezing and boiling point of water. (Do you know the temperature where Celsius and Fahrenheit are the same?)

Can you use Unix pipes to make one accept the output of the other?

Review quiz 6

True of false?

1. Within a certain range, all integers are available as values of an integer variable.
2. Within a certain range, all real numbers are available as values of a float variable.
3. `5(7+2)` is equivalent to 45.
4. `1--1` is equivalent to zero.
5. `int i = 5/3.;` The variable `i` is 2.
6. `float x = 2/3;` The variable `x` is approximately `0.6667`.