

C++ Intro Catchup

Victor Eijkhout, Susan Lindsey

Fall 2022

last formatted: August 31, 2022

Basics

1.

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:

```
#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.

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

Input/Output and strings

3.

You have already seen cout:

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

4.

```
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
```


5.

- 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.

Exercise 2

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?

Conditionals

6. If-then-else

A conditional is a test: 'if something is true, then do this, otherwise maybe do something else'. The C++ syntax is

```
if ( something ) {  
    // do something;  
} else {  
    // do otherwise;  
}
```

- The 'else' part is optional
- You can leave out braces in case of single statement.

7. Complicated conditionals

Chain:

```
if ( /* some test */ ) {  
    ...  
} else if ( /* other test */ ) {  
    ...  
}
```

Nest:

```
if ( /* some test */ ) {  
    if ( /* other test */ ) {  
        ...  
    } else {  
        ...  
    }  
}
```

8. Local variables in conditionals

The curly brackets in a conditional allow you to define local variables:

```
if ( something ) {  
    int i;  
    .... do something with i  
}  
// the variable 'i' has gone away.
```

Good practice: only define variable where needed.

Braces induce a scope.

Exercise 3

Read in a positive integer. If it's a multiple of three print 'Fizz!'; if it's a multiple of five print 'Buzz!'. If it is a multiple of both three and five print 'Fizzbuzz!'. Otherwise print nothing.

Note:

- Capitalization.
- Exclamation mark.
- Your program should display at most one line of output.

For loops

9.

The loop variable is usually an integer:

```
for ( int index=0; index<max_index; index=index+1) {  
    ...  
}
```

But other types are allowed too:

```
for ( float x=0.0; x<10.0; x+=delta ) {  
    ...  
}
```

Beware the stopping test for non-integral variables!

10. Nested loops

Traversing a matrix

(we will discuss actual matrix data structures later):

```
for (int row=0; row<m; row++)  
    for (int col=0; col<n; col++)  
        ...
```

This is called 'loop nest', with

row: outer loop

col: inner loop.

11. Indefinite looping

Sometimes you want to iterate some statements not a predetermined number of times, but until a certain condition is met. There are two ways to do this.

First of all, you can use a 'for' loop and leave the upperbound unspecified:

```
for (int var=low; ; var=var+1) { ... }
```

12. Break out of a loop

This loop would run forever, so you need a different way to end it. For this, use the `break` statement:

```
for (int var=low; ; var=var+1) {  
    statement;  
    if (some_test) break;  
    statement;  
}
```

Exercise 4

The integer sequence

$$u_{n+1} = \begin{cases} u_n/2 & \text{if } u_n \text{ is even} \\ 3u_n + 1 & \text{if } u_n \text{ is odd} \end{cases}$$

leads to the Collatz conjecture: no matter the starting guess u_1 , the sequence $n \mapsto u_n$ will always terminate at 1.

$$5 \rightarrow 16 \rightarrow 8 \rightarrow 4 \rightarrow 2 \rightarrow 1$$

$$7 \rightarrow 22 \rightarrow 11 \rightarrow 34 \rightarrow 17 \rightarrow 52 \rightarrow 26 \rightarrow 13 \rightarrow 40 \rightarrow 20 \rightarrow 10 \rightarrow 5 \dots$$

(What happens if you keep iterating after reaching 1?)

Try all starting values $u_1 = 1, \dots, 1000$ to find the values that lead to the longest sequence: every time you find a sequence that is longer than the previous maximum, print out the starting number.