C++ Programming

Y. Liow (July 21, 2025)

Contents

Chapter 1

Print statements and C-strings

OBJECTIVES

- Print strings and characters using `std::cout`
- Use the `\n`, `\t`, `\"`, `\'`, `\\` characters
- Use `std::endl` to force newline
- Write multiple print statements

Why the big, fat right margin? For your notes. Get busy.

In this set of notes, we learn to print strings and characters.

A quick advice to ease the pain of learning your first programming language: Type the program **exactly** as given. Even your spaces and blank lines must match the spaces and blank lines in my programs.

Let's begin ...

1.1 Hello world debug: hello-world.tex

Go ahead and run your first program:

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";

    return 0;
}
```

... commercial break ... go to notes on software tool(s) for writing and running a program ...

Practice doing a hello world program until you can do it in $<$ 2 minutes and without your notes.

Now let's go back to the C++ code. Right now, you should think of the stuff in bold as the program:

```
#include <iostream>

int main()
{
    *** YOUR "PROGRAM" GOES HERE ***

    return 0;
}
```

Therefore you can treat this as a template:

```
#include <iostream>

int main()
{


    return 0;
}
```

I will not explain things like "`#include <iostream>`" or "`int main()`" until later. (Technically, they are also part of the program.)

Try this

```
#include <iostream>

int main()
{
    std::cout << "Hello ... world! ... mom!\n";

    return 0;
}
```

Exercise 1.1.1. Write a program that prints the following: debug: exercises/240-0001/question.tex

```
hello columbia
```

You have 3 minutes. (Go to solution, page 3) □

Exercise 1.1.2. Write a program that gets the computer to greet you: debug: exercises/240-0002/question.tex

```
hello dr. liow, my name is c++.
```

(replace my name with yours ... you're probably not "`dr. liow`"). 2.5 minutes! (Go to solution, page 3) □

Exercise 1.1.3. Debug (i.e., correct) this program by hand and then verify by running it with your C++ compiler. 3 minutes! debug: exercises/240-0003/question.tex

```
#include (iostream)

int main()
(
    std;;cout < "Hello, world!\n";

    return 0;
)
```

(Go to solution, page 3) □

Solutions

Solution to Exercise 1.1.1.

debug: exercises/240-0001/answer.tex

```
#include <iostream>

int main()
{
    std::cout << "hello columbia\n";

    return 0;
}
```

□

Solution to Exercise 1.1.2.

debug: exercises/240-0002/answer.tex

Solution not provided. □

Solution to Exercise 1.1.3.

debug: exercises/240-0003/answer.tex

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";

    return 0;
}
```

□

1.2 Statement debug: statement.tex

Here's the first jargon. Look at our program again:

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";

    return 0;
}
```

The line in bold is a **statement**. In C++, statements must terminate with a **semi-colon**. Note that the semi-colon is part of the statement.

At this point you should think of a statement as something that will cause your computer to perform some operation(s) when you run the program.

If you like, you can think of a statement as a sentence and the semicolon as a period. When you're told to write a C++ statement, don't forget the semicolon!!! That would be like writing a sentence without a period (or question mark or exclamation mark) in an english essay.

Exercise 1.2.1. Modify your program by removing the first semicolon to get this: debug: exercises/240-0004/question.tex

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n"

    return 0;
}
```

Run it. Does it work? Fix it. Test to make sure it works (that means: run the program.) (Go to solution, page 6) □

Exercise 1.2.2. Replace the semicolons by periods. debug: exercises/240-0005/question.tex

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n".

    return 0.
}
```

Run it. Does it work? Fix it. (Go to solution, page 6) □

See what I mean by this advice:

> *A quick advice to ease the pain of learning your first programming language: Type the program* ***exactly*** *as given. Even the spaces and blank lines must match my programs.*

Computers are dumb (and picky about details.) We are the smart ones. So ... when you communicate with your computer, you have to be exact and explicit in your programs.

debug: exercises/240-0006/question.tex

Exercise 1.2.3. Now write a program that prints any message (example: "do you want green eggs and ham?") ... close your notes first. Peek at your notes only when you have problems. (Go to solution, page 6) □

Solutions

Solution to Exercise 1.2.1.

debug: exercises/240-0004/answer.tex

No. The semicolon is not optional. □

Solution to Exercise 1.2.2.

debug: exercises/240-0005/answer.tex

Solution not provided. □

Solution to Exercise 1.2.3.

debug: exercises/240-0006/answer.tex

```
#include <iostream>

int main()
{
    std::cout << "do you want green eggs and ham?\n";

    return 0;
}
```

□

1.3 C-strings debug: c-strings.tex

The stuff in quotes is called a **C-string** or just a **string**:

```
#include <iostream>

int main()
{
    std::cout << "hello world\n";

    return 0;
}
```

You can think of a string as textual data. (Soon I'll talk about numeric data for numeric computations. Got to have that for computer games, right?)

Double quotes are used to mark the beginning and ending of the string. So technically they are ***not*** part of the string. That's why when you run the above program, you do not see double-quotes.

Exercise 1.3.1. Does it work without the double-quotes? debug: exercises/240-0007/question.tex

```
#include <iostream>

int main()
{
    std::cout << Hello, world!\n;

    return 0;
}
```

(Go to solution, page 10) □

In the string `"Hello, world!\n"`, `H` is a **character**. The next character is `e`. Etc. When we talk about characters we enclose them with single-quotes. So I should say character `'H'` rather than `H` or `"H"`. You can think of the character as the smallest unit of data in a string.

The string `"Hello, world!\n"` contains characters `'H'` and `'e'` and `'l'` and `'l'` and ... However a character such as `'H'` can contain one and only one unit of textual data. So you ***cannot*** say that `'Hello'` is a character.

Note that a string can contain as many as characters as you like: 10,

20, 50, 100, etc. There's actually a limit, but we won't be playing around with a string with 1,000,000 characters anyway for now – that would be a pain to type!!! Note that a string can contain no characters at all: `""`. Of course a string can contain exactly one character. For instance, here's a string with one character: `"H"`.

Note that `"H"` is a string with one character whereas `'H'` is a character. You just have to look at what quotes are used to tell if the thingy is a string or a character. That's all.

Exercise 1.3.2. Does it work with single-quotes? debug: exercises/240-0008/question.tex

```
#include <iostream>

int main()
{
    std::cout << 'Hello, world!\n';

    return 0;
}
```

(Go to solution, page 10) □

Exercise 1.3.3. Can you also print characters?? (You did think about this, right?) Try this: debug: exercises/240-0009/question.tex

```
#include <iostream>

int main()
{
    std::cout << '?';

    return 0;
}
```

(Go to solution, page 10) □

Exercise 1.3.4. Which of the following is a string, a character, or neither: debug: exercises/240-0010/question.tex

```
"I"
like
{eggs}
"and"
'ham'
```

'.'

(Go to solution, page 10) □

Solutions

Solution to Exercise 1.3.1.

debug: exercises/240-0007/answer.tex

No! □

Solution to Exercise 1.3.2.

debug: exercises/240-0008/answer.tex

No! □

Solution to Exercise 1.3.3.

debug: exercises/240-0009/answer.tex

Yes. □

Solution to Exercise 1.3.4.

debug: exercises/240-0010/answer.tex

```
"I"      string
like     neither
'green'  neither
{eggs}   neither
"and"    string
'ham'    neither
'.'      character
```

□

1.4 Case sensitivity

debug: case-sensitivity.tex

Is C++ case sensitive?

Exercise 1.4.1. Modify your program by changing `std` to `Std`:

debug: exercises/240-0011/question.tex

```
#include <iostream>

int main()
{
    Std::cout << "Hello, world!\n";

    return 0;
}
```

Run it. Does it work? Fix it. (Go to solution, page 12) □

Exercise 1.4.2. Modify your program by changing `return` to `RETURN`:

debug: exercises/240-0012/question.tex

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";

    RETURN 0;
}
```

Does it work? Fix it.

Now answer this question:

Is C++ case sensitive? Circle one: YES NO

(duh ... I'm not taking answers in class.)

(Go to solution, page 12) □

Solutions

Solution to Exercise 1.4.1.

debug: exercises/240-0011/answer.tex

No! □

Solution to Exercise 1.4.2.

debug: exercises/240-0012/answer.tex

Solution not provided. □

1.5 Whitespaces debug: whitespaces.tex

A whitespace is ... well ... a white space.

Spaces, tabs, and newlines are whitespaces.

Exercise 1.5.1. Modify your program by inserting some spaces: debug: exercises/240-0013/question.tex

```
#include <iostream>

int main()
{
    std::cout            << "Hello, world!\n";

    return 0;
}
```

Does it work? (Go to solution, page 18) □

Exercise 1.5.2. Modify your program by inserting some newlines: debug: exercises/240-0014/question.tex

```
#include <iostream>

int main()
{
    std::cout <<




            "Hello, world!\n";

    return 0;
}
```

Does it run? (Go to solution, page 18) □

In general you can insert whitespaces between “basic words” understood by C++. These “basic words” are called **tokens**. (Oooooo another big word.)

If you think of statements as sentences, semi-colons as periods, then you can think of tokens as words.

For instance the following are some tokens from the above program: `int`, `return` and even `{`.

We say that C++ **ignores whitespace**.

Exercise 1.5.3. Try this:

debug: exercises/240-0015/question.tex

```
#include <iostream>

int main()
{
    std :: cout << "Hello, world!\n";

    return 0;
}
```

Does it work? Now try this:

```
#include <iostream>

int main()
{
    std:     :cout << "Hello, world!\n";

    return 0;
}
```

Does it work? (Go to solution, page 18) □

This shows you that **::** is a token – you cannot break it down.

Exercise 1.5.4. Try this:

debug: exercises/240-0017/question.tex

```
#include <iostream>

int main()
{
    std    ::    cout << "Hello, world!\n";

    return 0;
}
```

Does it work? (Go to solution, page 18) □

Try this:

```
#include <iostream>

int main()
{
    std::cout << "Hello,          world\n";

    return 0;
}
```

The whitespaces between tokens are removed when your computer runs your C++ program. So, to the computer, it doesn't matter how much whitespace you insert between tokens – it still works.

However spaces in a **string** are **not** removed before the program runs. The space characters `' '` (there are 10 in the above string) are actually characters within the string.

Note that although you can insert whitespaces, in general, good spacing of a program makes it easier to read. Therefore you must follow the style of spacing shown in my notes. In particular, I **don't** want to see monstrous programs like this (although the program does work):

```
#include <iostream>

int                          main(){std
::
                     cout<<

"Hello, world!\n"; return 0; }
```

or this:

```
#include <iostream>
int main(){std::cout<<"Hello, world!\n";return 0;}
```

And don't try to be cute this like ...

```
#include <iostream>
int
 main
  ()
   {
    std
     ::
      cout
       <<
        "Hello, world!\n";
         return 0;
          }
```

The above examples are excellent candidates for the F grade. The correct coding style produces this:

```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";

    return 0;
}
```

The left trailing spaces of a statement is called the **indentation** of the statement:

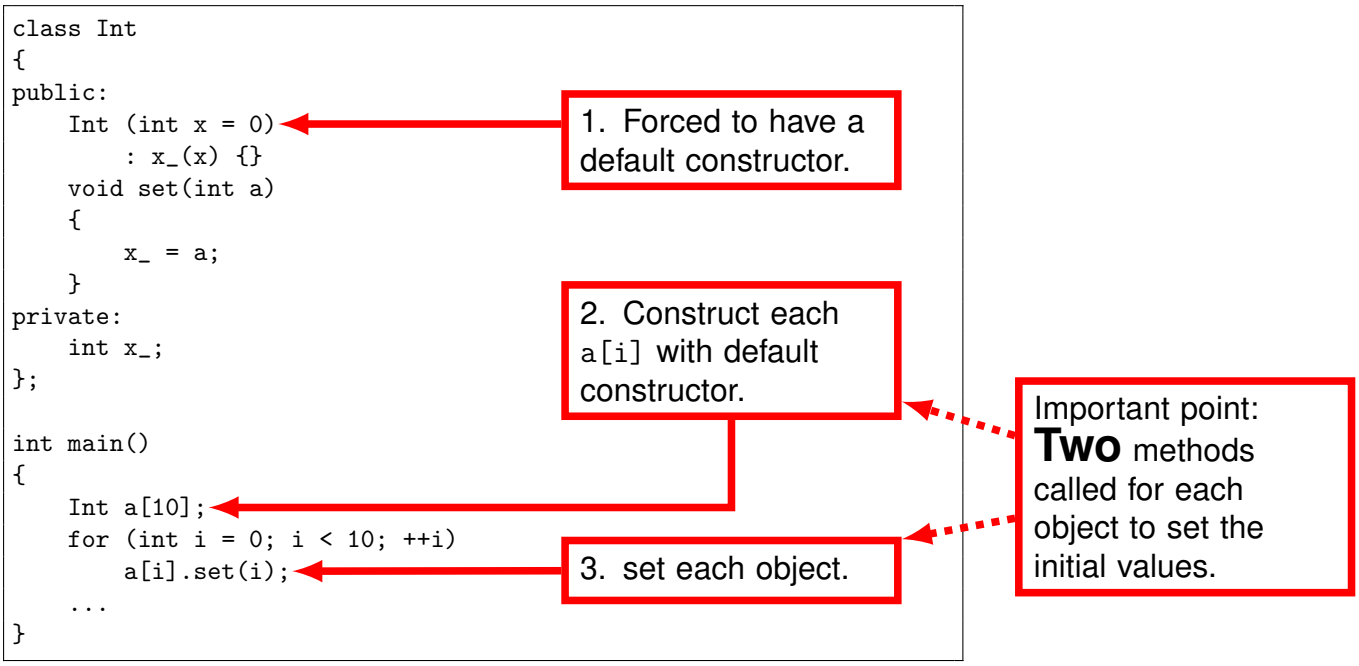
```
#include <iostream>

int main()
{
    std::cout << "Hello, world!\n";
    return 0;
}
```

Indentation

It's a common programming style to use 4 spaces for indentation.

1.6 Nasser


class Int
{
public:
Int (int x = 0)
: x_(x) {}
void set(int a)
{
x_ = a;
}
private:
int x_;
};
int main()
{
Int a[10];
for (int i = 0; i < 10; ++i)
a[i].set(i);
...
}
1. Forced to have a default constructor.
2. Construct each a[i] with default constructor.
3. set each object.
Important point: Two methods called for each object to set the initial values.

Solutions

Solution to Exercise 1.5.1.

debug: exercises/240-0013/answer.tex

Solution not provided. □

Solution to Exercise 1.5.2.

debug: exercises/240-0014/answer.tex

Solution not provided. □

Solution to Exercise 1.5.3.

debug: exercises/240-0015/answer.tex

No! Therefore the two semicolons are part of the same word. □

Solution to Exercise 1.5.4.

debug: exercises/240-0017/answer.tex

Yes. This means that `std`, `::`, `cout` are separate tokens in the C++ language. □