

PCP — Lecture 02

Fall 2020 August 17, 2020

Last Time - Presentation

- Lab component and class component. Attendance is not mandatory.
- Grading: quizzes, projects, and exams. Lab and homework are not due.
- Everything is incremental: cannot succeed if missed a lecture or a key concept.
- To get help, contact Crystal Anderson or Dr. Aubert through email, teams, discord, ...

1 Intro: Principles of Computer Programming:

1.1 Principles

1. Commons principles at the core of every programming language.
2. Always been and always will be (some studies since ancient history!).
3. Ties with Mathematics.

1.2 Computer

1. Particular, contingent.
2. Evolving fast (parallel computing, quantum computing, etc.).
3. Ties with Physics (circuits).

1.3 Programming

Basic building blocks that are used to construct *any* program (= software = app = application).

2 Generalities

2.1 On Computers

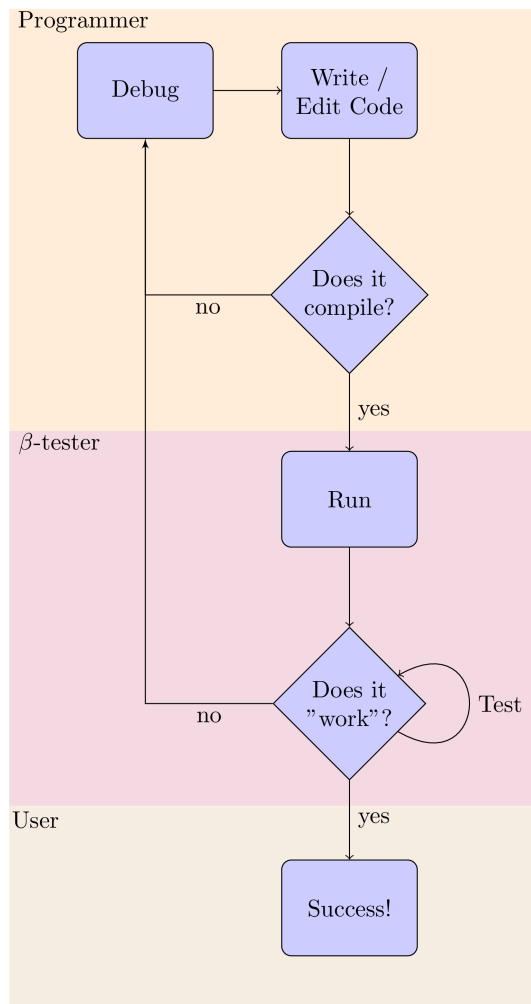
- Not interested in the hardware component in this lecture: we will treat software as platform-independent
- There are three levels of languages: Machine-code (read by computers, not humans), Assembly, High-Level Language (read by humans, not computers).
- Assembler and compiler have the same “target language” (machine code), but different “source languages” (assembly language for assembler, high-level language for compiler).
- Compiler and code are particular to one language.

2.2 On Software

- Interaction with software: **G**raphical **U**ser **I**nterface / **C**ommand-**L**ine **I**nterface, and **I**nterface / **O**utput
- A single software can use multiple programming languages (taking benefits of their efficient characteristics).

3 Specificity of C#

3.1 Workflow



Some of the specificity of C# is that:

1. Everything will be done in Visual Studio (VS).
2. Compiling is called "Build" and running is called "Debugging" or "Start without Debugging"

3.2 An object-oriented language

There are multiple paradigms (cf. https://en.wikipedia.org/wiki/Programming_paradigm), i.e. “families”, of languages. Object-oriented language is one of them, and Java, C#, Python, ... are in that same family. The key concepts (that we will study at length) are:

- Class = Blueprint, cookie-cutter, birthday card invitation template (with blanks), ...
- Object = Actual house, cookie, birthday card with a name and an address, ...
- Method (Action that you can perform uniformly on objects) = Painting the house, decorating the cookie, sending the card, ...

3.3 History

- C# started in 2000, inspired by C, C++ and Java. Launched by Windows.
- C# is in the TOP 5 of the most used programming languages [Source 1](#), [Source 2](#)
- Its popularity seems to be eroding [Source](#)
- Solid, stable, widespread, used by [Unity's API](#).

4 A Simple Program

```

1  // Hello!
2
3  using System;
4  class Welcome
5  {
6      static void Main()
7      {
8          Console.WriteLine("Welcome to PCP!");
9      }
10 }
11 /*
12  This is a multi-line
13  comment
14  */

```

Only one line actually *does* something, (`Console.WriteLine("Welcome to PCP!");`), the rest is “overhead”: for now, let’s take that to be a magical incantation, we’ll expound on that soon.

“Line by line” reading of the code:

- There are two *statements* in this code: `Console.WriteLine("Welcome to PCP!");` and `using System;`. They both end with `;`.
- There are two kinds of comments (for the programmer and ignored by the computer): inline comment (`//...`) and delimited, or multi-line, comments (`/* ... */`),
- **Using** directive. Namespace (= API, Library, i.e., set of classes)
- The class has a *header* (l. 4), a *body* (l. 5-10), and a *name* (`Welcome`).
- The method also has a *header* (l. 6), a *body* (l.7-9), and a *name* (`Main`).
- The indentation (spacing) and the colors matter to the programmer, but not to the compiler.

5 Rules and Conventions

5.1 Rules

General rules for the syntax of C#:

1. C# is case-sensitive.
2. Every statement ends with a ;.
3. Braces and parenthesis must be matched.

Oddly enough, C# does not care (almost at all) about spaces, but the programmer does: this is the difference between rules (use exact case, for instance) and conventions (like indentation, class name starting with an upper-case letter, ...). Rules are required by the compilers, conventions are recommended by the programmers (and your instructors!).

There are also rules when we will be writing classes:

- You can have as many methods as you want per class.
- The **Main** method is the entry point of a C# application.

5.2 Conventions

Some conventions include:

- Have only one class per file.
- The **.cs** file has the same name as the class inside it.

Those are actually rules in Java!

6 Reserved Words and Identifiers

- Some words are reserved (they are called *keywords*): they have a really particular meaning for C# and we can use them only in one way. **using**, **class**, **static**, **void**, **namespace**, ... (**Main** is not a keyword).
- Some words are fixed by the programmers: those are the identifiers (**Welcome**). Class name, variable name, method name, ...
- Some words were fixed by other programmers (**System**, **Console**, **WriteLine**): if I want to use what they programmed for us, I have to use the name they picked. They also picked identifiers and gathered them in a *namespace* (=API, collection of classes we can use).

6.1 Rules For Identifiers

“Rules”: for the compiler.

An identifier...

1. Must not be a reserved word.
2. Must contain only letters, $a \rightarrow Z$, digits, $0 \rightarrow 9$, underscore, **_**.
3. Does not begin with digits.
4. Does not contain any spaces.

6.2 Conventions For Identifiers

“Conventions”: for the programmers, common to many programming languages (some of Java’s rules are C# conventions).

1. Identifiers should be descriptive.
2. Class names starts with an upper-case letter.
3. Pick names that are easy to memorize and type (CLASs is probably not a good choice).

7 Multiple Statements and Escape Sequences

*Starting here, your instructor will draw at the board “Output window”, and will stop writing all the code, to focus on the **Main** method or notion at stake.*

In our body, we can have multiple statements:

```
1 Console.WriteLine("Hey");  
2 Console.WriteLine("Mom");
```

In addition to `Console.WriteLine`, we can also use `Console.Write`: the difference is that the former adds a new line after displaying the message, while the latter only displays the message, without adding a new line.

We can also use escape sequences for special characters:

- `\n` - start a new line (newline character)
- `\a` - play a sound (alert)
- `\t` - horizontal tab

All of these use the escape character (`\`).