Computer Programming 1

Concepts and Notions

Outline

- Basic concepts on coding/programming
- Programming paradigms
- Introduction to C++

Programming

- Coding is writing lines of code in the programming language for creating a computer program (software).
- Programming is a complex activity which includes coding, but also other activities, e.g., problem analysis, design of the solution to solve the problem in terms of definition of the algorithm and data structures, implementation….

  Several phases:
  - Planning
  - Design
  - Implementation
  - Testing
  - Deployment
  - Maintenance
- To write code, you start from a mental schema or structure, pass though a pseudocode to describe the logic of the algorithm, and implement the program

Programming

- Coding is writing lines of code in the programming language for creating a computer program (software).
- Programming is a complex activity which includes coding, but also other activities, e.g., problem analysis, design of the solution to solve the problem in terms of definition of the algorithm and data structures, implementation….

  Several phases:
  - Planning
  - Design
  - Implementation
  - Testing
  - Deployment
  - Maintenance

Understand the problem and the context

- To write code, you start from a mental schema or structure, pass though a pseudocode to describe the logic of the algorithm, and implement the program

Programming

- Coding is writing lines of code in the programming language for creating a computer program (software).
- Programming is a complex activity which includes coding, but also other activities, e.g., problem analysis, design of the solution to solve the problem in terms of definition of the algorithm and data structures, implementation….

  Several phases:
  - Planning
  - Design
  - Implementation
  - Testing
  - Deployment
  - Maintenance

Understand the problem and the context

Identify and design the solution – design the algorithm

- To write code, you start from a mental schema or structure, pass though a pseudocode to describe the logic of the algorithm, and implement the program

Programming

- Coding is writing lines of code in the programming language for creating a computer program (software).
- Programming is a complex activity which includes coding, but also other activities, e.g., problem analysis, design of the solution to solve the problem in terms of definition of the algorithm and data structures, implementation….

  Several phases:
  - Planning — *Understand the problem and the context*
  - Design — *Identify and design the solution – design the algorithm*
  - Implementation — *Develop the solution – implement the algorithm*
  - Testing
  - Deployment
  - Maintenance
- To write code, you start from a mental schema or structure, pass though a pseudocode to describe the logic of the algorithm, and implement the program

Programming

- Coding is writing lines of code in the programming language for creating a computer program (software).
- Programming is a complex activity which includes coding, but also other activities, e.g., problem analysis, design of the solution to solve the problem in terms of definition of the algorithm and data structures, implementation….

  Several phases:

  - Planning — *Understand the problem and the context*
  - Design — *Identify and design the solution – design the algorithm*
  - Implementation — *Develop the solution – implement the algorithm*
  - Testing — *Verify and validate the developed solution*
  - Deployment
  - Maintenance

- To write code, you start from a mental schema or structure, pass though a pseudocode to describe the logic of the algorithm, and implement the program

Programming

- Coding is writing lines of code in the programming language for creating a computer program (software).
- Programming is a complex activity which includes coding, but also other activities, e.g., problem analysis, design of the solution to solve the problem in terms of definition of the algorithm and data structures, implementation….

  Several phases:

  - Planning — *Understand the problem and the context*
  - Design — *Identify and design the solution – design the algorithm*
  - Implementation — *Develop the solution – implement the algorithm*
  - Testing — *Verify and validate the developed solution*
  - Deployment — *Provide the developed solution – ready to be used*
  - Maintenance
- To write code, you start from a mental schema or structure, pass though a pseudocode to describe the logic of the algorithm, and implement the program

Programming

- Coding is writing lines of code in the programming language for creating a computer program (software).
- Programming is a complex activity which includes coding, but also other activities, e.g., problem analysis, design of the solution to solve the problem in terms of definition of the algorithm and data structures, implementation….

  Several phases:

• Planning	*Understand the problem and the context*
• Design	*Identify and design the solution – design the algorithm*
• Implementation	*Develop the solution – implement the algorithm*
• Testing	*Verify and validate the developed solution*
• Deployment	*Provide the developed solution – ready to be used*
• Maintenance	*Maintain and evolve the developed solution – updates*

- To write code, you start from a mental schema or structure, pass though a pseudocode to describe the logic of the algorithm, and implement the program

Programming

- Coding is writing lines of code in the programming language for creating a computer program (software).
- Programming is a complex activity which includes coding, but also other activities, e.g., problem analysis, design of the solution to solve the problem in terms of definition of the algorithm and data structures, implementation….

  Several phases:
  - Planning
  - Design
  - Implementation
  - Testing
  - Deployment
  - Maintenance

Do we always need to write our code from scratch?

No.
Code reuse!
(not part of this course)

- To write code, you start from a mental schema or structure, pass though a pseudocode to describe the logic of the algorithm, and implement the program

Coding is not programming

	Coding	Programming
Scope	It is a process in which a set of instructions are converted into a language that is comprehensible for a computer	Other than coding, it concerns the definition of the requirements, problem solving, pseudocode writing, algorithm thinking, optimizing, testing, executable code creation, maintenance
Skill	As a coder, a good knowledge of syntax and semantics of the selected programming language is required	As a programmer, additional skills are required with respect to the ones of the coder, e.g., capability of level thinking and of problem analysis

If Programming is the process of writing a whole book.
Then Coding is just about writing a single chapter of the book.

https://hackr.io/blog/coding-vs-programming-difference-you-should-know

Coding is not programming

	Coding	Programming
Scope	It is a process in which a set of instructions are converted into a language that is comprehensible for a computer	Other than coding, it concerns the definition of the requirements, problem solving, **pseudocode writing**, **algorithm thinking**, optimizing, testing, executable code creation, maintenance
Skill	As a coder, a good knowledge of syntax and semantics of the selected programming language is required	As a programmer, additional skills are required with respect to the ones of the coder, e.g., capability of level thinking and of problem analysis

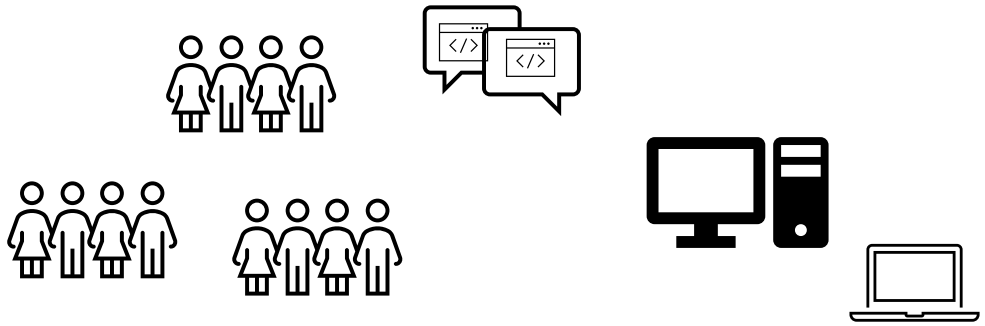
If Programming is the process of writing a whole book.
Then Coding is just about writing a single chapter of the book.

https://hackr.io/blog/coding-vs-programming-difference-you-should-know

What is a programming language? (1)

- A programming language is a set of rules that provides a way to:
  - specify an algorithm to the computer
  - communicate to a computer what it has to execute

A programming language is a rule-based notational system used to describe computation in **machine-readable** and **human-readable** form



What is a programming language? (2)

- English is a natural language
  - It has words, symbols and (grammatical) rules
  - A programming language also has words, symbols and (grammatical) rules

Language = Syntax + Semantics

Syntax

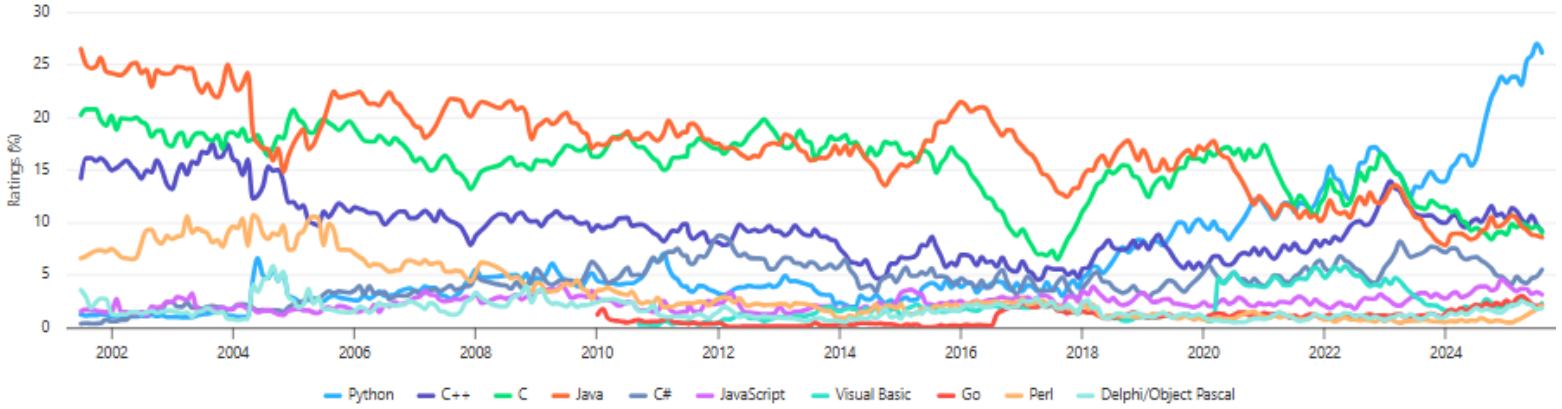
- It describes **composition and structure of a program**
- It is the set of grammatical rules
- It is a collection of rules that specifies the structure/form of the code

Semantic

- It describes the **meaning of a program**
- It is a collection of rules that specifies the interpretation of the code and the meaning associated to words, symbols, and code

Why so many programming languages?

- Evolution history
- Evolution of needs
- Evolution of the technology
- Different objectives
- Different paradigms
- Different programming languages are designed for different types of program
- …

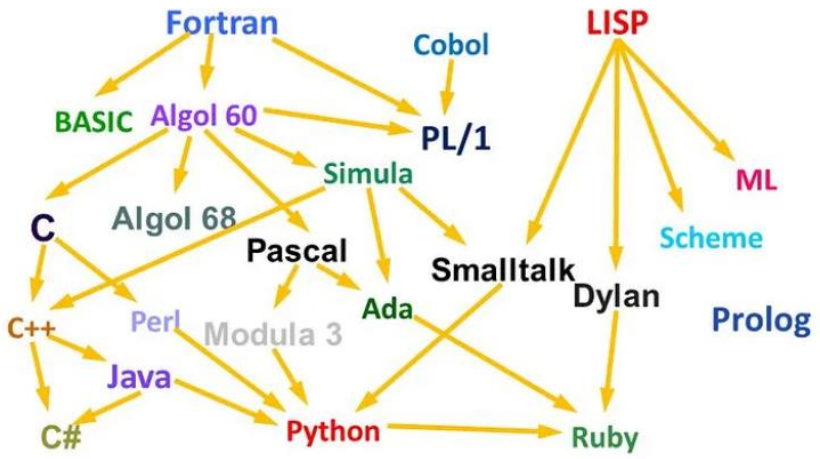


Aug 2025	Aug 2024	Change	Programming Language	Ratings	Change
1	1		Python	26.14%	+8.10%
2	2		C++	9.18%	-0.86%
3	3		C	9.03%	-0.15%
4	4		Java	8.59%	-0.58%
5	5		C#	5.52%	-0.87%
6	6		JavaScript	3.15%	-0.76%
7	8	^	Visual Basic	2.33%	+0.15%
8	9	^	Go	2.11%	+0.08%
9	25	^^	Perl	2.08%	+1.17%
10	12	^	Delphi/Object Pascal	1.82%	+0.19%
11	10	v	Fortran	1.75%	-0.03%
12	7	vv	SQL	1.72%	-0.49%
13	30	^^	Ada	1.52%	+0.91%
14	19	^^	R	1.37%	+0.26%
15	13	v	PHP	1.27%	-0.19%
16	11	vv	MATLAB	1.19%	-0.53%
17	20	^	Scratch	1.15%	+0.06%
18	14	vv	Rust	1.13%	-0.15%
19	18	v	Kotlin	1.10%	-0.04%
20	17	v	Assembly language	1.03%	-0.19%

https://www.tiobe.com/tiobe-index

Types of languages

- Several different taxonomies and criteria to classify
  - *intended domain of use:* general-purpose programming vs domain-specific languages
  - *applied programming paradigm*: declarative (based on declaration of "truth") vs imperative (based on list of instructions) languages
  - *abstraction level*: high-level (close to the human language), low-level (close to the machine language, e.g., assembly), executable machine code (see picture on the right-bottom)
  - *execution type*: compiled (translated to machine code by a compiler) interpreted (an interpret translates instruction-by-instruction during the execution), just-in-time compiled languages (code segments are compiled at runtime by needs)
  - *type of typing*: strongly (type must be explicitly converted) vs weakly (automatic conversion of type as needed) typed



History of Programming Languages by U.Lewis

```
class Triangle {
  ...
  float surface()
    return b*h/2;
  }
```

High-level

```
LOAD r1,b
LOAD r2,h
MUL r1,r2
DIV r1,#2
RET
```

Low-level

```
0001001001000101
0010010011101100
10101101001...
```

Executable machine code

Programming paradigms

- Procedural programming (C, Pascal, Perl)
- Object-oriented programming (C++, Java, Python, C#)
- Logic programming (Prolog, Haskell)
- Functional programming (Lisp, ML)
- Scripting programming (JS, PHP, Perl, Python)
- Command Language programming (sh, csh, bash)
- Event-driven programming (Javascript, Perl)
- Markup-language programming (HTML, XML)
- …

Other examples exist!!

Other classifications and taxonomies exist!!

Programming paradigms

- Procedural programming (C, Pascal, Perl)
- Object-oriented programming (C++, Java, Python, C#)
- Logic programming (Prolog, Haskell)
- Functional programming (Lisp, ML)
- Scripting programming (JS, PHP, Perl, Python)
- Command Language programming (sh, csh, bash)
- Event-driven programming (Javascript, Perl)
- Markup-language programming (HTML, XML)
- …

Programming paradigms

- Procedural programming (C, Pascal, Perl)

- It is procedure/problem-oriented
- A program is a sequence of commands used to modify the state
- A program is structured in instructions and functions
- Provide fine-grade control of capability and efficiency (in terms of compilation and execution time)
- It is imperative programming
- Use case: often used for system programs and embedded systems

```c
#include <stdio.h>
int main() {
   // printf() displays the string inside quotation
   printf("Hello, World!");
   return 0;
}
```

```
Hello, World!
```

Programming paradigms

- Object-oriented programming (C++, Java, Python, C#)

- It is based on “objects”, i.e., units that contain data in the form of fields and code in the form of procedures
- It offers features like abstraction, encapsulation, polymorphism, inheritance, and classes
- High code reusability
- It is imperative programming
- Use case: large system and application software

```java
class Main {

  public static void main(String[] args) {

    System.out.println("Enter two numbers");
    int first = 10;
    int second = 20;

    System.out.println(first + " " + second);

    // add two numbers
    int sum = first + second;
    System.out.println("The sum is: " + sum);
  }
}
```

```
Enter two numbers
10 20
The sum is: 30
```

Programming paradigms

- Logic programming (Prolog, Haskell)

- It is largely based on formal logic
- A program is a set of sentences in logical form, expressing facts (true predicates), rules (in the form of clauses) and questions (about a problem domain)
- The language employs restrictions on what the machine must consider doing
- It is declarative programming
- Use case: specific tasks that involve symbolic or non-numeric computation (intelligent database retrieval)

```
/*Facts*/
food(pizza).    /* pizza is a food */
food(burger).   /* burger is a food */
food(sandwich). /* sandwich is a food */
food(milk).     /* milk is a food */
lunch(sandwich).    /* sandwich is a lunch */
lunch(milk).        /* milk is a lunch */
dinner(pizza).      /* pizza is a dinner */
dinner(burger).

/* Rules */
meal(X) :- food(X).
/* Every food is a meal OR Anything is a
 *  meal if it is a food */

/** <examples>
//Queries
?- food(pizza).  // Is pizza a food?
?- meal(X), lunch(X). // Which food is meal and lunch?
?- dinner(sandwich). // Is sandwich a dinner?
*/
```

```
meal(X), lunch(X).
X = sandwich
X = milk

?- meal(X), lunch(X).
Examples▲ History▲ Solutions▲   table results  Run!
```

https://athena.ecs.csus.edu/~mei/logicp/prolog/Prolog_Example_1.pdf

Programming paradigms

- Functional programming (Lisp, ML)

- Programs are constructed by applying and composing functions
- It is built on the concept of mathematical functions which uses conditional expressions and recursion to perform the calculation
- It is declarative programming
- It is possible to use functional-style programming in non-functional programming languages (e.g., C++, Python, Java, JS)
- Use case: problems where several operations have to be performed on the same set of data

Language: Common Lisp Layout: Vertical

```
; set value 1 to 3
(setq val1 3)
; set value 2 to 6
(setq val2 6)

;addition operation
(print (+ val1 val2))
```

Absolute running time: 0.17 sec, cpu time: 0.02 sec, memory peak: 9 Mb

```
9
```

https://rextester.com/l/common_lisp_online_compiler

Programming paradigms

- Scripting programming (JS, PHP, Perl, Python)

- Interpreted
- Often, weakly and dynamically typed
- Development speed
- Reduced runtime speed and maintainability
- Typically, can run multi-platform but in an execution environment
- Use case: data elaboration, web programming

```
# This program adds two numbers

num1 = 1.5
num2 = 6.3

# Add two numbers
sum = num1 + num2

# Display the sum
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

```
The sum of 1.5 and 6.3 is 7.8
```

Programming paradigms

- It encapsulates lists of commands in files
- Often, it is used to automate/batch processes
- Available in all Operating Systems (e.g., Linux)
- Tailored on the hosting environment (OS)
- Use case: automation of tasks and process execution

- Command Language programming (sh, csh, bash)

```
#!/bin/bash
file='book.txt'
while read line; do
echo $line
done < $file
```

```
$ bash read_file.sh
```

```
ubuntu@ubuntu-Vi:~/code$ bash read_file.sh
1. Pro AngularJS
2. Learning JQquery
3. PHP Programming
4. CodeIgniter 3
```

https://linuxhint.com/30_bash_script_examples/#t23

Programming paradigms

- The program flow is driven by events such as user actions (mouse clicks, key presses), sensor outputs, message passing from other programs or threads.
- A program is generally a loop that listens for events and then triggers a callback function when one of those events is detected
- Use case: define graphical user interfaces and programs centered on reacting to certain actions or events

- Event-driven programming (Javascript, Perl)

```perl
#!/usr/bin/perl
# your code goes here
use strict;
use warnings;

print "Please type in the names of the programming languages you know: ";
my @names = <STDIN>;

chomp @names;
print "Hello, I see you know " . scalar(@names) . "\n";
```

Success #stdin #stdout 0.01s 5432KB

stdin

```
Perl
C++
Java
Python
```

stdout

```
Please type in the names of the programming languages you know: Hello, I see you know 4
```

https://ideone.com

Programming paradigms

- Text-encoding system in which a document consists of two type of text: (a) content and (b) special symbols controlling the text structure and formatting
- Symbols are interpreted by the computer to control the structure of the document (i.e., document rendering/display)
- Use case: HTML the markup language for Web document displayed by the browser

- Markup-language programming (HTML, XML)

```html
<!DOCTYPE html>
<html>
<body>
 <h2>First Page</h2>
 <p>Link list</p>
  <ul>
   <li>
     <a href="https://link_address_1"> Link 1 </a>
   </li>
   <li>
     <a href="https://link_address_2"> Link 2 </a>
   </li>
  </ul>
</body>
</html>
```

First Page

Link list

- Link 1
- Link 2

Programming paradigms

- Low-level programming language
- Just one step before the machine code
- It is converted into machine code by an assembler
- It is extremely fast
- Machine-dependent
- Difficult to program
- Use case: code for device with limited resources

- … Assembly programming

```
; ---------------------------------------------------------------------------
; Writes "Hello, World" to the console using only system calls. Runs on 64-bit Linux only.
; To assemble and run:
;
;     nasm -felf64 hello.asm && ld hello.o && ./a.out
; ---------------------------------------------------------------------------

          global    _start

          section   .text
_start:   mov       rax, 1                  ; system call for write
          mov       rdi, 1                  ; file handle 1 is stdout
          mov       rsi, message            ; address of string to output
          mov       rdx, 13                 ; number of bytes
          syscall                           ; invoke operating system to do the write
          mov       rax, 60                 ; system call for exit
          xor       rdi, rdi                ; exit code 0
          syscall                           ; invoke operating system to exit

          section   .data
message:  db        "Hello, World", 10      ; note the newline at the end
```

https://cs.lmu.edu/~ray/notes/x86assembly

Programming paradigms

- Optimized for mathematical operations
- Optimized to work with large set of mathematical data
- Programs are flow of operations on data
- Often, provided with large sets of mathematical and statistical libraries
- Use case: data analytics and data science

- … Mathematical programming (R, Matlab)

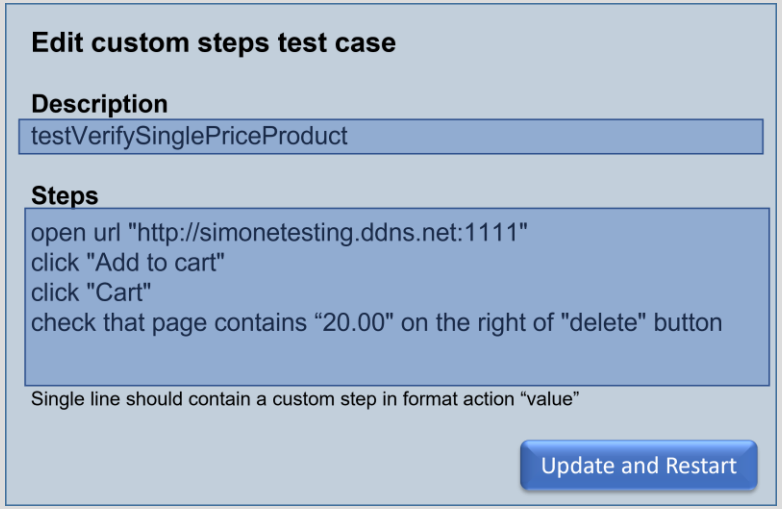
```r
num     = as.integer(readline(prompt = "Enter a number: "))
# num  = 15
isPrime = 0
if (num > 1) {
    isPrime = 1
    for (i in 2: (num - 1)) {
        if ((num %% i) == 0) {
            isPrime = 0
            break
        }
    }
}
if (num == 2) isPrime = 1
if (isPrime == 1) {
    print(paste(num, "is a prime number"))
} else {
    print(paste(num, "is not a prime number"))
}
```

```
Enter a number: 15
[1] "15 is not a prime number"

Enter a number: 13
[1] "13 is a prime number"
```

Programming paradigms

- Programming languages that attempt to use human languages
- A program is a structured document with content, sections and subsections containing natural-language sentences
  - constraints are usually accepted
- Require:
  a. visual of graphical interfaces to develop the natural-language source code;
  b. use of ontologies
  c. use of natural language processing techniques;
  d. an AI/machine-learning engine to convert the natural language into executable tasks
- Use case: program synthesis



- … Natural language programming

Programming paradigms

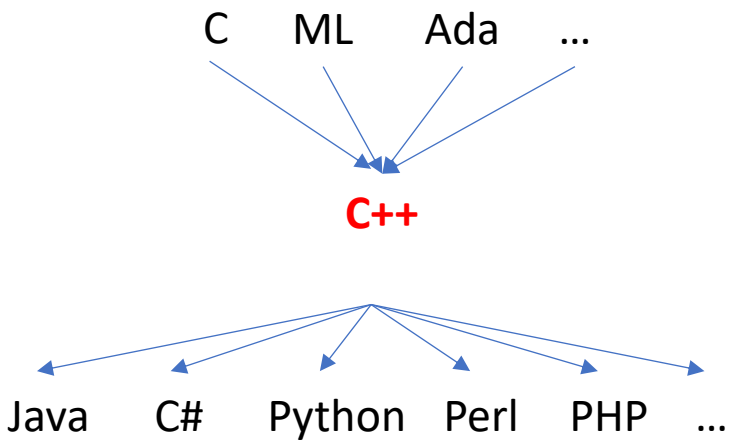
- Procedural programming (C, Pascal, Perl)
- Object-oriented programmin
- Logic programming (Prolog
- Functional programming (L
- Scripting languages (JS, PH
- Command Languages (sh, c
- Text-processing languages
- Markup-languages (HTML,
- …

- Most modern languages are multi-paradigm
- However, in most of the cases, programming language has intrinsic characteristics and a main expected domain and use
- Knowledge of the most appropriate language to adopt for a given task in a given domain can make the difference

Which programming language in the course?

- In the course we will use **C++**

- General purpose programming language
  - System/dektop, embedding, network … programming
- Multi-paradigm language
  - Procedural, functional, object-oriented
- It is a standardized language (by ISO/IEC)
  - C++98 1998, C++03 2003, C++11 2011, C++14 2014, C++17 2017, C++20 2020
- Developed in the Bell Labs (1983)
  - Evolution of C (→ "C++")



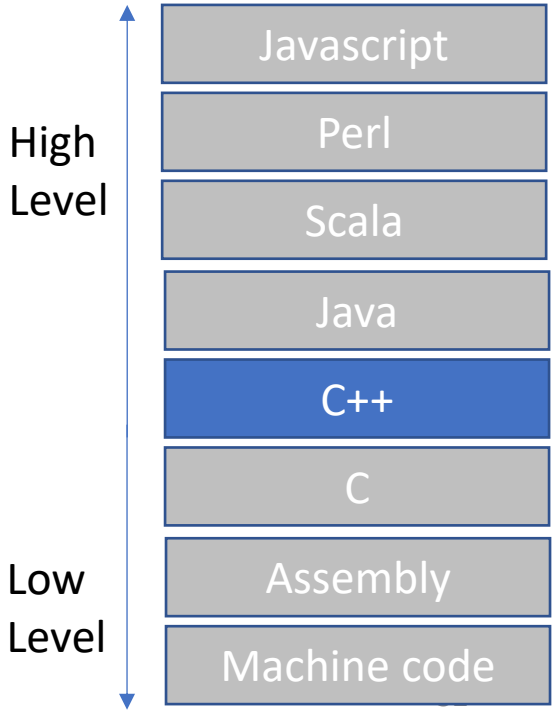
About the name
1. B (BCPL) : Basic Combined Programming Language
2. After B → C
3. After C → C++

Why C++ on this course?

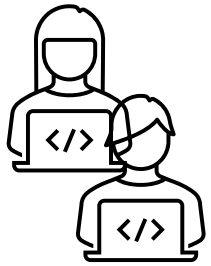
1. Learning about computers and compilers
   - C++ helps in understanding the internal architecture of a computer, how computer stores and retrieves information
   - C++ helps in understanding how compilers work
   - C++ helps in understanding the memory management and pointers
   - C++ helps in learning about compile time and load time
   - C++ helps in learning about program optimization, dynamic libraries and generic programming, etc.

Aug 2025	Aug 2024	Change	Programming Language	Ratings	Change
1	1		Python	26.14%	+8.10%
2	2		C++	9.18%	-0.86%
3	3		C	9.03%	-0.15%
4	4		Java	8.59%	-0.58%
5	5		C#	5.52%	-0.87%

2. Learning other programming languages
   - C++ helps in picking up other languages faster
   - After learning C++, it will be much easier to learn other programming languages
3. Type of developed applications
   - C++ lets build lightweight, high-performance, and simple solutions
   - C++ work perfectly with Operating System and network APIs
4. Diffusion of developed applications
   - C++ is everywhere (OS, web, embedded systems, …)
   - C++ is used to develop games, desktop apps, operating systems, browsers, and so on because of its power, flexibility, and its performance.



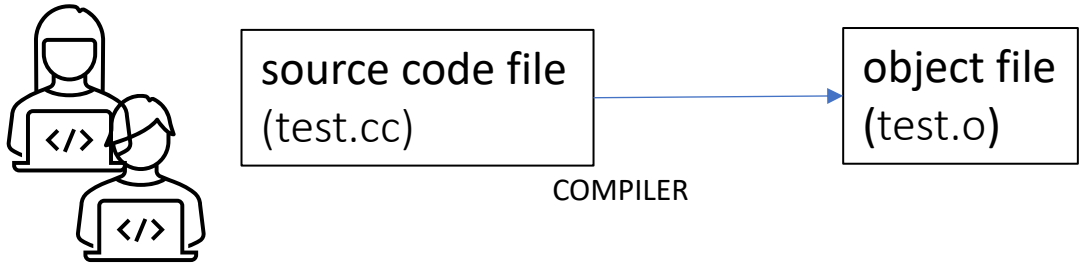
Software program development – flow (1/4)



source code file
(test.cc)

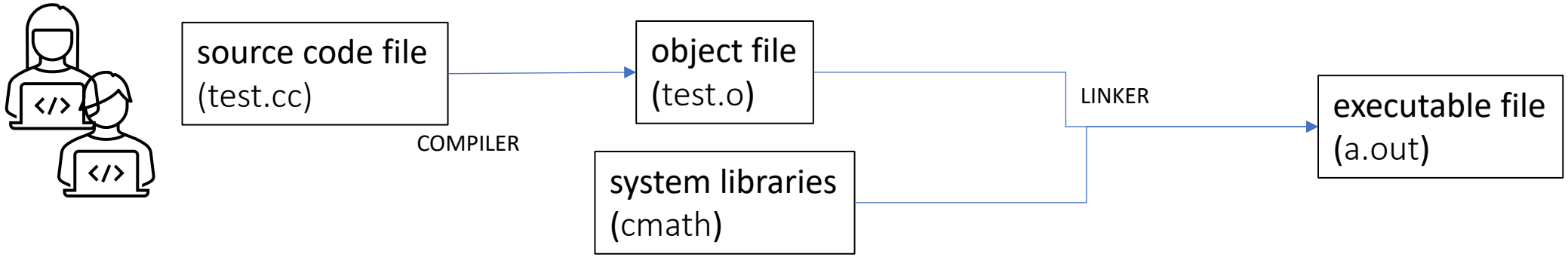
- One or more text files, called source code of the program

Software program development – flow (2/4)



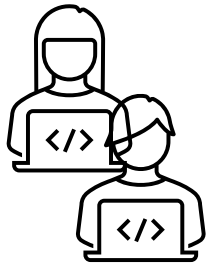
- One or more text files, called source code of the program
- The source code is converted into object file/s (not human-readable) from the compiler
  - E.g., g++ -c test.cc

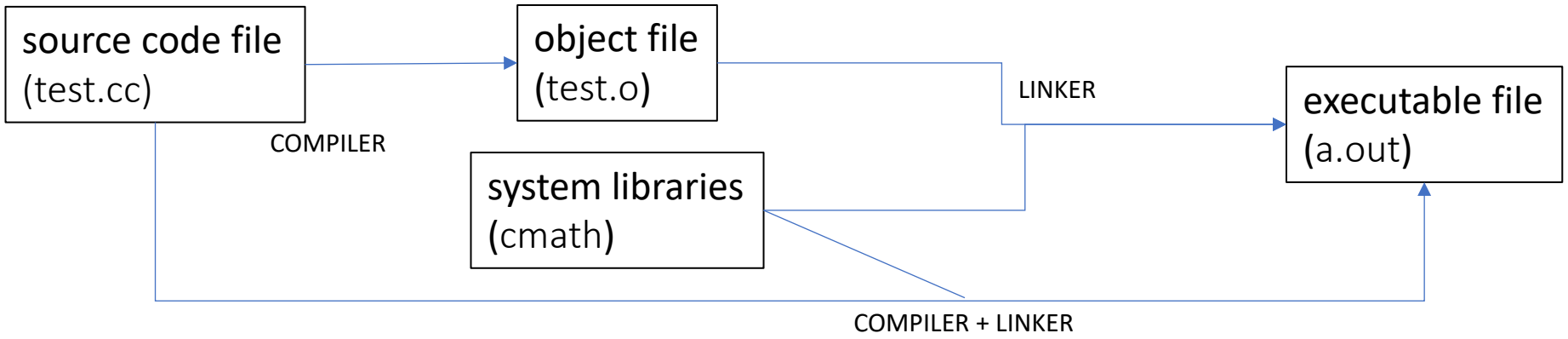
Software program development – flow (3/4)



- One or more text files, called source code of the program
- The source code is converted into object file/s (not human-readable) from the compiler
  - E.g., g++ -c test.cc
- Object file/s is/are linked to system libraries by the linker, thus generating the executable file (default name: a.out, -o <name> can be used to change the name)
  - E.g., g++ test.o
  - E.g., g++ test.o -o test
  - These files are unreadable for humans, but they are understandable and executable for computers

Software program development – flow (4/4)





- One or more text files, called source code of the program
- The source code is converted into object file/s (not human-readable) from the compiler
  - E.g., g++ -c test.cc
- Object file/s is/are linked to system libraries by the linker, thus generating the executable file (default name: a.out, -o <name> can be used to change the name)
  - E.g., g++ test.o
  - E.g., g++ test.o -o test
  - These files are unreadable for humans, but they are understandable and executable for computers
- All together compilation and linking:
  - E.g., g++ test.cc

Template for a C++ program

- Example of a common structure for a C++ program
  - {../L01_01_EXAMPLE_BASE/template.cc}

```
using namespace std ;                    //declarative regions for the (standard) program identifiers
#include <iostream>                      //library call
int main ( )
{
   return 0;                             //instruction for the program output
}
```

https://devdocs.io/cpp/
https://en.cppreference.com/w/
https://en.cppreference.com/w/cpp/header

Template for a C++ program

- Example of a C++ program with output a predefined string
  - {../L01_01_EXAMPLE_BASE/hello.cc}

```
using namespace std ;                    //declarative regions for the (standard) program identifiers
#include <iostream>                      //library call
int main ( )                             // main function
{
   cout << "Hello world \n";             //instruction for the program output
   return 0;
}
```

- Program variant with "endl"
  - {../L01_01_EXAMPLE_BASE/hello2.cc}

Template for a C++ program

- Example of a C++ program with output a predefined string
  - {../L01_01_EXAMPLE_BASE/hello.cc}

```
using namespace std ;                    //declarative regions for the (standard) program identifiers
#include <iostream>                      //library call
int main ( )                             // main function
{
   cout << "Hello world \n";             //instruction for the program output
   return 0;
}
```

```
$ ls
hello.cc  hello2.cc
$ g++ hello.cc
$ ls
a.out hello.cc  hello2.cc
$ ./a.out
Hello World
$
```

Template for a C++ program

- Example of a C++ program with output a predefined string
  - {../L01_01_EXAMPLE_BASE/hello.cc}

```
using namespace std ;                    //declarative regions for the (standard) program identifiers
#include <iostream>                      //library call
int main ( )                             // main function
{
   cout << "Hello world \n";             //instruction for the program output
   return 0;
}
```

```
$ ls
hello.cc  hello2.cc
$ g++ hello.cc
$ ls
a.out hello.cc  hello2.cc
$ ./a.out
Hello World$
```

Program variant:

- {../L01_01_EXAMPLE_BASE/hello2.cc}