University Of Anbar

College Of CS & IT

Computer Science

Programming in CPP

First Stage

First Semester

2025-2026

Lecture One

Dr. Ruqayah Ayad Al-ani

Lecture1: Introduction to Programming Languages (C++)

1. Introduction to Programming Languages

Programming languages are formal systems that allow humans to communicate instructions to computers (Deitel & Deitel, 2024). These languages provide a structured way to write commands that the machine can execute to perform specific tasks.
A program is essentially a sequence of instructions expressed in a particular programming language, which is then translated into machine code that the computer understands (ISO/IEC, 2020).

Like human languages, programming languages have grammar and vocabulary. In computing, grammar refers to **syntax**, which defines the structure of valid programs, and **semantics**, which determines their meaning (Stroustrup, 2013). Learning these rules is the foundation of becoming a proficient programmer.

Example in C++:
```cpp
#include <iostream>
using namespace std;
int main() {
   cout << "Hello, World!" << endl; // Output to screen
   return 0;
}
```
This simple program outputs "Hello, World!" to the screen, demonstrating the basic syntax of a C++ program.

2. Purpose and Classification of Programming Languages

Programming languages can be classified into categories (Deitel & Deitel, 2024):
- **Low-level languages**: Closer to machine code, like Assembly and C, allowing direct hardware control.

- **High-level languages**: More abstract and user-friendly, such as Python, Java, and C++.

- **Domain-specific languages**: Designed for specialized purposes, e.g., SQL for databases.

Example classification:
Low-level: Assembly, C
High-level: C++, Java, Python
Domain-specific: SQL, MATLAB

3. Historical Development of C++

C++ was developed in the early 1980s by Bjarne Stroustrup at Bell Labs to combine the efficiency of C with object-oriented features (Stroustrup, 2013).
Originally called "C with Classes," it introduced features such as encapsulation, inheritance, and polymorphism (Meyers, 2014).

Over time, C++ evolved through standardization:
- **C++98/03**: First standardization.
- **C++11/14**: Modern features like `auto`, lambda expressions, smart pointers.
- **C++17/20**: Further refinements, modules, concepts.

Example of OOP in C++:

```
#include <iostream>
using namespace std;
```

```
class Car {
public:
    string brand;
    int year;
    void display() {
        cout << brand << " (" << year << ")" << endl;
    }
};

int main() {
    Car c1;
    c1.brand = "Toyota";
    c1.year = 2023;
    c1.display();
    return 0;
}
```

4. Key Features of C++

C++ offers features that make it versatile and powerful (Stroustrup, 2013; ISO/IEC, 2020):
- **Object-Oriented Programming**: Encapsulation, inheritance, polymorphism.
- **High Performance**: Near-hardware execution speed.

- **Standard Template Library (STL)**: Predefined classes/functions for algorithms and data structures.

- **Portability**: Compile across platforms with minimal changes.

Example using STL:

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
   vector<int> nums = {1, 2, 3};
   nums.push_back(4);
   for (int n : nums) cout << n << " ";
   return 0;
}
```

5. Core Concepts in C++

When learning C++, core topics include (Deitel & Deitel, 2024):

- Variables and data types
- Operators
- Control structures
- Functions
- Arrays and pointers
- Classes and objects

Example of control structure:

```
#include <iostream>
using namespace std;
int main() {
   int score = 85;
   if (score >= 90) cout << "Grade: A";
   else if (score >= 80) cout << "Grade: B";
   else cout << "Grade: C";
   return 0;
}
```

6. Programming Workflow in C++

A typical workflow involves (Meyers, 2014):
1. **Planning**: Identify the problem and design a solution.
2. **Coding**: Write the program in C++.
3. **Compiling**: Use a compiler (e.g., g++) to produce executable code.
4. **Testing**: Verify correctness.
5. **Debugging**: Identify and fix errors.

7. Importance of Learning C++

C++ develops problem-solving skills, logical thinking, and efficiency (Stroustrup, 2013). It is used in game engines, embedded systems, simulations, and high-frequency trading platforms (Meyers, 2014).

Example: Game development uses C++ for graphics rendering and real-time processing.

8. Conclusion

Programming languages are essential for all software development. C++ stands out for combining efficiency, flexibility, and modern features. Mastering it opens opportunities in diverse, high-demand fields.

References

- Deitel, P., & Deitel, H. (2024). “C++ How to Program: An Objects-Natural Approach” (11th ed.). Pearson. ISBN 9780138101640.
- ISO/IEC. (2020). “ISO/IEC 14882:2020 Information technology — Programming languages — C++”. International Organization for Standardization.
- Meyers, S. (2014). “Effective Modern C++: 42 Specific Ways to Improve Your Use of C++11 and C++14”. O’Reilly Media. ISBN 9781491903995.
- Stroustrup, B. (2013). “The C++ Programming Language” (4th ed.). Addison-Wesley.