Programming Essentials

(UOMU022023)

اساسيات البرمجة
2025-2024

Introduction

by
Dr Murtada Dohan
murtada.dohan@uomus.edu.iq

Agenda

- Programming Languages
- C++ Program
- Special Characters
- cout & cin

Programming Languages

- **Machine Language** consisting of binary code (0s and 1s). Computers understand only this. It is processor dependent.
- **Assembly Language** consisting of mnemonics (symbols) to make programming less tedious and faster. (e.g. Load A, Add B, Store C). An Assembler converts assembly language into machine language. It is also processor dependent.
- **High-level Language** consisting of English-like instructions to make programming simpler and faster. (e.g. C= A+B). A compiler/interpreter helps convert high- level language into machine language. It is not processor dependent.

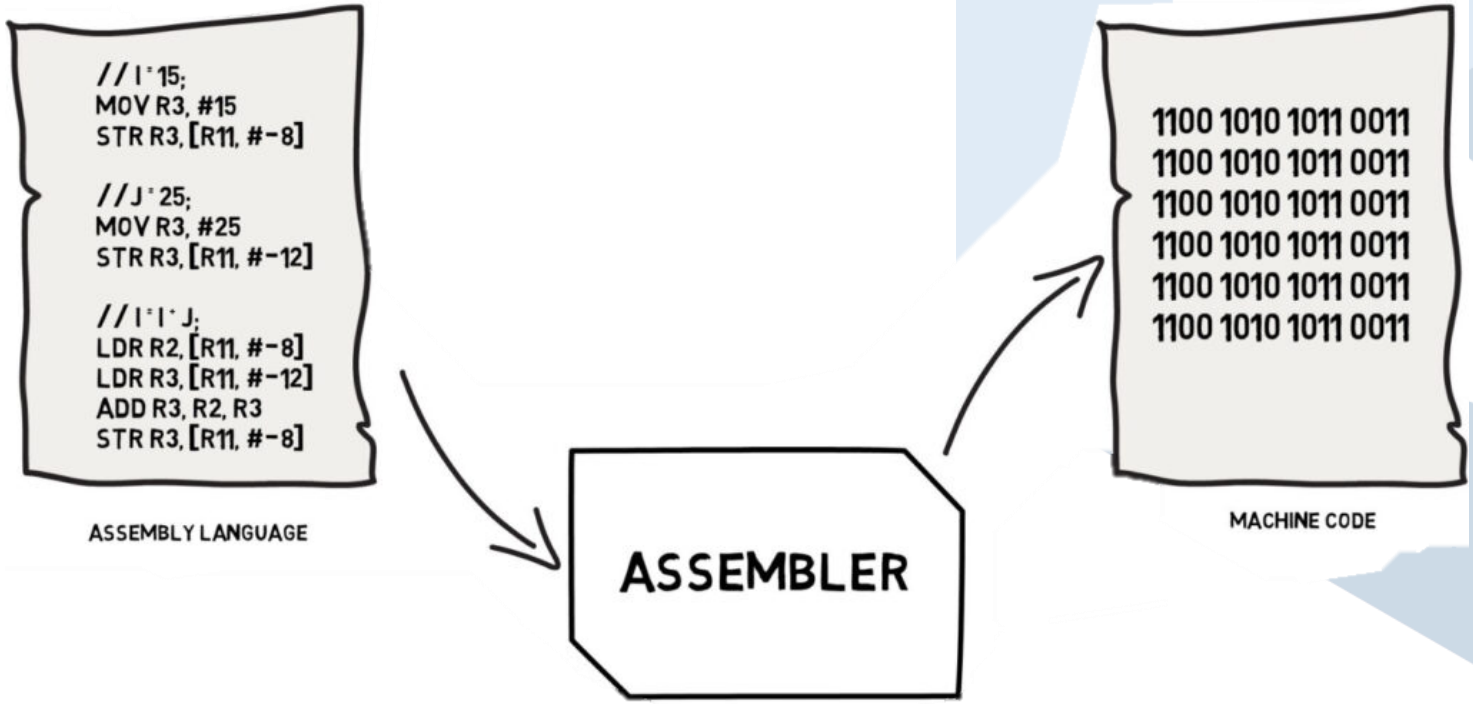
Machine Language

- Programming Language - Way to write instructions
- Machine Language: 0s and 1s (computer understands)
- Example: 10110011
- Hard for humans!

```
01010000 01110010 01101111 01100111 01110010 01100001 01101101 01101101
01101001 01101110 01100111 00100000 01001100 01100001 01101110 01100111
01110101 01100001 01100111 01100101 00100000 00101101 00100000 01010111
01100001 01111001 00100000 01110100 01101111 00100000 01110111 01110010
01101001 01110100 01100101 00100000 01101001 01101110 01110011 01110100
01110010 01110101 01100011 01110100 01101001 01101111 01101110 01110011
```

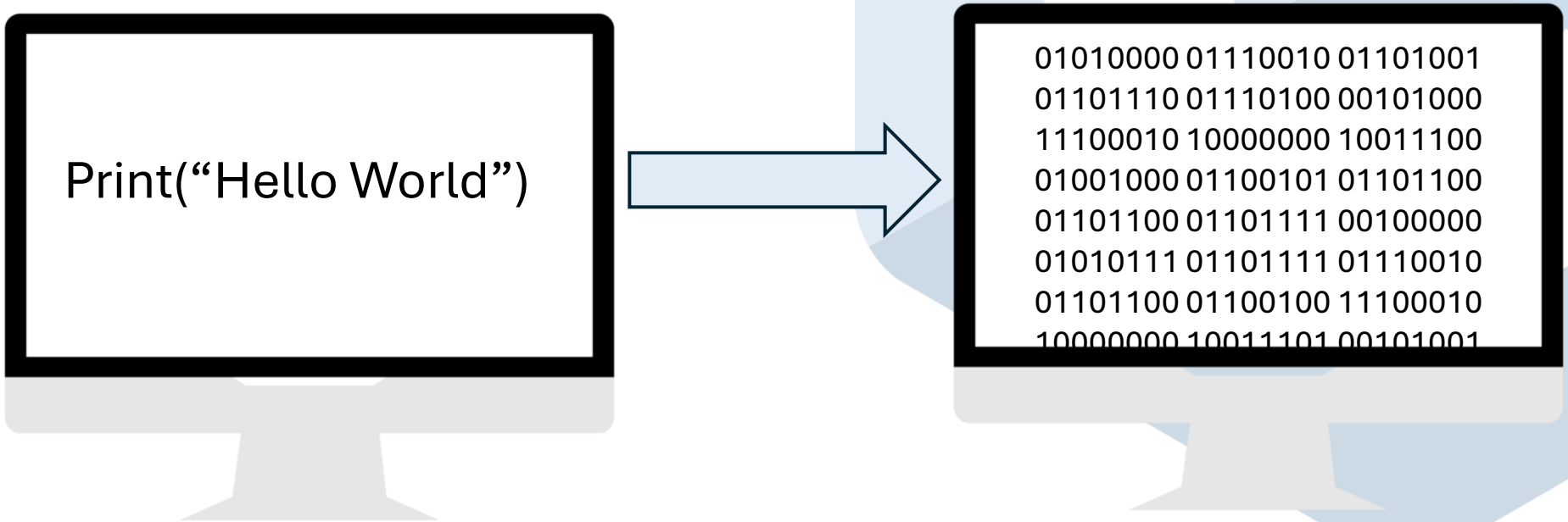
Assembly Language

- Uses words like "ADD" instead of 0s and 1s
- Easier than machine language
- Still close to how computers work



High-level languages

- Like English: print("Hello")
- Examples: C++, Python, Java
- Easy for humans, needs translation for computers



The Parts of a C++ Program

```
// sample C++ program
#include <iostream>
using namespace std;
int main()
{
      cout << "Hello, there!";
      return 0;
}
```

comment

preprocessor directive

which namespace to use

beginning of function named main

beginning of block for main

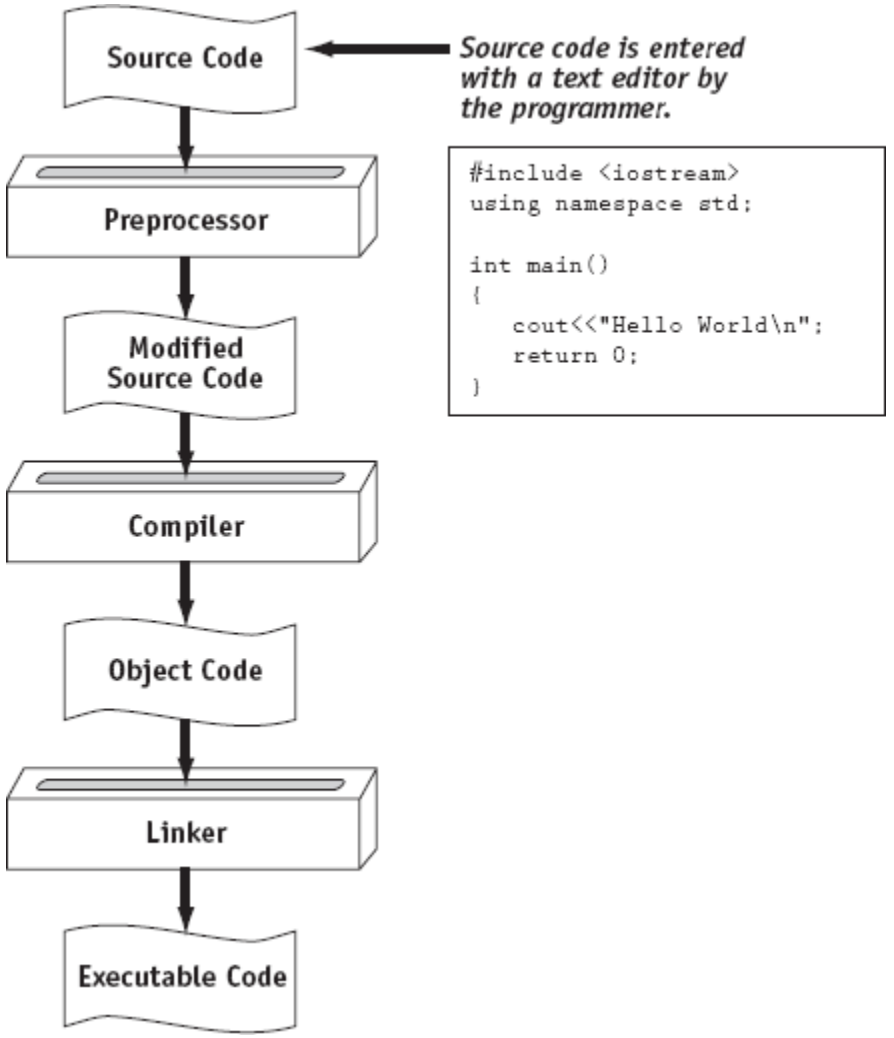
output statement

string literal

send 0to operating system

end of block for main

High-Level Program to an Executable File




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

int main()
{
   cout<<"Hello World\n";
   return 0;
}
```

Hello World Program

```
//*************************************
// This program prints Hello World!
//*************************************
#include <iostream>
using namespace std;
int main ()
{
   cout << "Hello World!"; return 0;
}
```

Special Characters

Character	Name	Meaning
//	Double slash	Beginning of a comment
#	Pound sign	Beginning of preprocessor directive
< >	Open/close brackets	Enclose filename in #include
( )	Open/close parentheses	Used when naming a function
{ }	Open/close brace	Encloses a group of statements
" "	Open/close quotation marks	Encloses string of characters
;	Semicolon	End of a programming statement

The `cout` Object

- Displays output on the computer screen
- You use the stream insertion operator << to send output to cout:

```
cout << "Programming is fun!";
```

The cout Object

- Can be used to send more than one item to cout:

```
cout << "Hello " << "there!";
```

Or:

```
cout << "Hello ";
cout << "there!";
```

The `cout` Object

- This produces one line of output:

```
cout << "Programming is ";
cout << "fun!";
```

The endl Manipulator

- You can use the **endl** manipulator to start a new line of output. This will produce two lines of output:

```
cout << "Programming is" << endl;
cout << "fun!";
```

The endl Manipulator

```
cout << "Programming is" << endl;
cout << "fun!";
```

The `endl` Manipulator

- You do NOT put quotation marks around **endl**

- The last character in **endl**is a lowercase L, not the number 1.

`endl` ⟵ This is a lowercase L

The \n Escape Sequence

- You can also use the **\n**escape sequence to start a new line of output. This will produce two lines of output:

```
cout << "Programming is\n"; cout <<
"fun!";
```

Notice that the \nis INSIDE the string.

The \n Escape Sequence

```
cout << "Programming is\n";
cout << "fun!";
```

The `cin` Object

- Standard input object
- Like `cout`, requires `iostream` file
- Used to read input from keyboard
- Information retrieved from cinwith >>
- Input is stored in one or more variables

The `cin` Object

Program 3-1

```
1  // This program asks the user to enter the length and width of
2  // a rectangle. It calculates the rectangle's area and displays
3  // the value on the screen.
4  #include <iostream>
5  using namespace std;
6
7  int main()
8  {
9     int length, width, area;
10
11    cout << "This program calculates the area of a ";
12    cout << "rectangle.\n";
13    cout << "What is the length of the rectangle? ";
14    cin >> length;
15    cout << "What is the width of the rectangle? ";
16    cin >> width;
17    area = length * width;
18    cout << "The area of the rectangle is " << area << ".\n";
19    return 0;
20 }
```

Program Output with Example Input Shown in Bold

```
This program calculates the area of a rectangle.
What is the length of the rectangle? 10 [Enter]
What is the width of the rectangle? 20 [Enter]
The area of the rectangle is 200.
```

The `cin` Object

- **`cin`** converts data to the type that matches the variable:

```
int height;
cout << "How tall is the room? ";
cin >> height;
```

Let’s try C++

Install Visual Studio and familiarise yourself with its interface.