CodeWave Academy

Introduction To computer science

By: CodeWave Academy

Overview

- The different between hardware & software
- What Is Programming and Program Development Life Cycle ?
- High-level & low-level & Assembly language
- What is compiler & how it is work?
- The different between compiler & interpreter & Assembler
- How can I write a c++ code ?
- Comments
- Your First C++ Program

Introduction

Hardware and **software** are two terms you've probably heard of at some point or another. The odds are high that you use both on a daily basis, whether it's with your smartphone or personal computer. Let's take a deeper look at what these two things are and why they're important.

Hardware is any element of a computer that's **physical**. This includes things like **monitors**, **keyboards**, and also the insides of devices, like **microchips** and **hard drives**.

Software is anything that **tells hardware what to do** and **how to do it**, including computer **programs** and **apps** on your phone. Video games, photo editors, and web browsers are just a few examples.

Hardware and **software** are different from each other, but they also **need one another in order to function**. Let's look at an example of this using a smartphone. In this case, the hardware would be the physical phone itself, and the software would be its operating system and apps.

operating system:

The operating system is **an interface between a user's program and the hardware** and provides a variety of services and supervisory functions. Among the most important functions are:

- Examples of operating systems in use to - Handling basic input and output operations

day are Linux, iOS, and Window

What Is Programming and Program Development Life Cycle ?

- **Programming is a process of problem solving**

- **Step 1: Analyze the problem:**

– Outline the problem and its requirements

– Design steps (algorithm) to solve the problem

Algorithm:

– Step-by-step problem-solving process

- **Step 2: Implement the algorithm**

– Implement the algorithm in code

– Verify that the algorithm works

- **Step 3: Maintenance**

– Use and modify the program if the problem domain changes

The Problem Analysis–Coding–Execution Cycle :

- Understand the Overall problem
- Understand problem requirements
  - Does program require user interaction?
  - Does program manipulate data?
  - What is the output?
- If the problem is complex, divide it into subproblems
  - Analyze each subproblem as above

The Language of a Computer :

We have a three types of language :

-High-level language

-Low-level language

-Assembly language

1. High-level language:

A **high-level language** is a programming language that is designed to make it easier for humans to understand and write. It is closer to natural language

The Language of a Computer :

2. Low-level language:

low level languages which are closer to hardware as compared to high-level languages

❑ Types of Low-Level Languages

Low level language are divided into two types.

The Language of a Computer :

i) Machine Language:

We know that machines follow the language of binary system, means **0 and 1**. Machine language is low level language which consists of binary codes which are directly operated by CPU Central Processing Unit. There every instruction are written in form of 0 and 1.

ii) Assembly Language:

Assembly Language is a way of writing computer programs that are very close to how the computer works. It have some symbols and codes that represent the basic operations that the computer can perform, which includes adding, moving, or comparing numbers.

The Evolution of Programming Languages:

100100 010001 //Load

100110 010010 //Multiply

100010 010011 //Store

Machine code	Assembly code	Description
001 1 000010	LOAD #2	Load the value 2 into the Accumulator
010 0 001101	STORE 13	Store the value of the Accumulator in memory location 13
001 1 000101	LOAD #5	Load the value 5 into the Accumulator
010 0 001110	STORE 14	Store the value of the Accumulator in memory location 14
001 0 001101	LOAD 13	Load the value of memory location 13 into the Accumulator
011 0 001110	ADD 14	Add the value of memory location 14 to the Accumulator
010 0 001111	STORE 15	Store the value of the Accumulator in memory location 15
111 0 000000	HALT	Stop execution

compiling process:

What is a compiler?

A compiler is a special program that translates a programming language's Source code into machine code

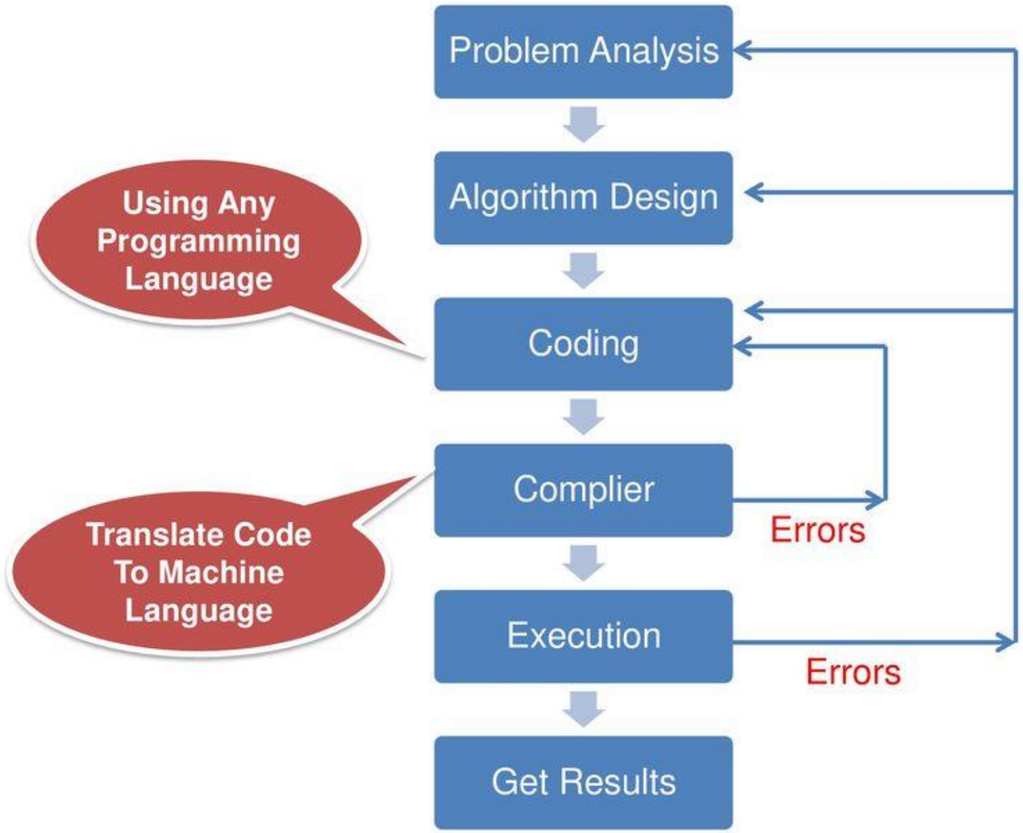
The source code is typically written in a high-level, human-readable language such as C# or C++

Compilers that translate source code to machine code target specific operating systems

•**Compiler**: A compiler translates code from a high-level programming language into machine code before the program runs.

•**Interpreter**: An interpreter translates code written in a high-level programming language into machine code line-by-line as the code runs.

The Problem Analysis–Coding–Execution Cycle



main.cpp [test1] - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Management

Projects Files FSymbols

- Workspace
  - test1
    - Sources
      - main.cpp

main.cpp

```cpp
#include <iostream>

using namespace std;

int main()
{
  std::  cout << "Hello world!" << endl;

}
```

Logs & others

Code::Blocks | Search results | Cccc | Build log | Build messages | CppCheck/Vera++ | CppCheck/Vera++ messages | Cscope | Debugger | Doxyl

```
Opening C:\c++ acadmy\test1\test1.cbp
Done.
NativeParser::DoFullParsing took: 0.494 seconds.
ProjectManager::SetProject took: 0.679 seconds.
ProjectManager::LoadProject took: 0.710 seconds.
NativeParser::CreateParser: Finish creating a new parser for project 'test1'
NativeParser::OnParserEnd: Project 'test1' parsing stage done!
```

C:\c++ acadmy\test1\main.cpp | C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 10, Col 1, Pos 109 | Insert | Read/Write | default

25°C غائم غالبًا ENG 01:51 ص ٢٠٢٤/٠٨/٣٠

❑ What are libraries?

A library is a collection of pre-written code that you can use to perform specific tasks.

❑ Why c++?

#include <iostream> => preprocessor statement

#include => preprocessor directive

- The preprocessor directive in c++ begin with ‘ **#** ’ that tells the processor to modify code before compiling We have more of preprocessor directive and used to link **header file** with source code

Search about it …..

#include <header_file**>** **or** **#include “**header_file**”**

Iostream => library

- Comment?

- Write the first program in c++.
  - Cout<<“hello world”;

Marketing Strategy

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam laoreet risus fringilla, egestas elit a, consequat risus augue. Phasellus sollicitudin felis mi, risus quis egestas ex ornare sed quis adipiscing consectetur laoreet.

01

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam laoreet risus fringilla, egestas elit a, consequat augue. Phasellus sollicitudin felis mi, quis egestas ex ornare sed quis adipiscing.

02

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam laoreet risus fringilla, egestas elit a, consequat augue. Phasellus sollicitudin felis mi, quis egestas ex ornare sed quis adipiscing.

03

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam laoreet risus fringilla, egestas elit a, consequat augue. Phasellus sollicitudin felis mi, quis egestas ex ornare sed quis adipiscing.

04

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam laoreet risus fringilla, egestas elit a, consequat augue. Phasellus sollicitudin felis mi, quis egestas ex ornare sed quis adipiscing.

THANK YOU!