



Computer Programming

UNIVERSITY OF HARGEISA

COLLAGE OF COMPUTING AND IT

DEPARTMENT OF COMPUTER SCIENCE

Suleiman Gargaare

MSc in Computer Science, MSc in Project Management & BSc in
Information Technology.

Lecturer

suleiman.gargaare@yahoo.com

Course Contents

- **Chapter 1 – Introduction**
 - Chapter 2 – Basic Elements in C++
 - Chapter 3 – Input and Output
 - Chapter 4 – Control Structure
 - Chapter 5 – Functions
 - Chapter 6 – Arrays & Strings
-



Computer Programming 1 with C++

Chapter 1

Introduction of Computers and
Programming Languages

Programming Languages Ranking!

For the first time in more than 20 years, the TIOBE programming languages index [has a new leader](#) of the pack: the Python programming language.

Oct 2021	Oct 2020	Change	Programming Language		Ratings	Change
1	3	▲		Python	11.27%	-0.00%
2	1	▼		C	11.16%	-5.79%
3	2	▼		Java	10.46%	-2.11%
4	4			C++	7.50%	+0.57%
5	5			C#	5.26%	+1.10%

Introduction

- In this course, you'll learn programming the right way from the beginning.
- We present multiple numbers of complete working programs and shows the outputs produced when those programs are run on a computer.
- We call this the “live-code approach.” All of these example programs, you can get and download from my Github account:

[Github.com/Gargaare](https://github.com/Gargaare)

- It's **software** (i.e., the instructions you write to command computers to perform **actions** and make **decisions**) that controls computers (often referred to as **hardware**).

- C++, an object-oriented programming language based on C, is of such interest today that we've included a detailed introduction to C++ and object-oriented programming in the later chapters.
- To keep up to date with C and C++ developments at Deitel & Associates, register for the *Deitel[®] Buzz Online*, at
 - www.deitel.com/newsletter/subscribe.html

Computers: Hardware & Software

- A **computer** is a device that can perform computations and make logical decisions billions of times faster than human beings can.
 - Many of today's personal computers can perform several billion additions per second.
 - Today's fastest **supercomputers** can perform thousands of trillions (quadrillions) of instructions per second!
-

- Computers process **data** under the control of sets of instructions called **computer programs**
- These programs guide the computer through orderly sets of actions specified by people called **computer programmers**.
- A computer consists of various devices referred to as hardware (e.g., the keyboard, screen, mouse, hard disk, memory, DVDs and processing units).
- The programs that run on a computer are referred to as software.
- Hardware costs have been declining dramatically in recent years, to the point that personal computers have become a commodity.

Computer Organization

- Every computer may be envisioned as divided into six **logical units** or sections:
 - **Input unit:** This “receiving” section obtains information (data and computer programs) from **input devices** and places it at the disposal of the other units so that it can be processed. Humans typically enter information into computers through keyboards and mouse devices. Information also can be entered in many other ways, including by speaking to your computer, scanning images and barcodes, reading from secondary storage devices (like hard drives, CD drives, DVD drives and USB drives—also called “thumb drives”).
-

- ❑ **Output unit:** This “ shipping” section takes information that the computer has processed and places it on various **output devices** to make it available for use outside the computer. Most information that is output from computers today is displayed on **screens**, printed on paper, **played on audio players** (such as Apple’s popular **iPods**), or used to control other devices. Computers also can output their information to networks, such as the Internet.

- ❑ **Memory unit:** This rapid-access, relatively low-capacity “warehouse” section retains information that has been entered through the input unit, making it immediately available for processing when needed. The memory unit also retains processed information until it can be placed on output devices by the output unit. Information in the memory unit is **volatile**—it’s typically lost when the computer’s power is turned off. The memory unit is often called either **memory** or **primary memory**.

- **Arithmetic and logic unit (ALU):** This “manufacturing” section performs calculations, such as addition, subtraction, multiplication and division. It also contains the decision mechanisms that allow the computer, for example, to compare two items from the memory unit to determine whether they’re equal. In today’s systems, the ALU is usually implemented as part of the next logical unit, the CPU.

Cont..

- ❑ **Central processing unit (CPU)**: This “administrative” section coordinates and supervises the operation of the other sections. The CPU tells the input unit when to read information into the memory unit, tells the ALU when information from the memory unit should be used in calculations and tells the output unit when to send information from the memory unit to certain output devices. Many of today’s computers have multiple CPUs and, hence, can perform many operations simultaneously—such computers are called **multiprocessors**. For example a dual-core processor has two CPUs and a quad-core processor has four CPUs.

Cont..

- **Secondary storage unit:** This is the long-term, high-capacity “warehousing” section. Programs or data not actively being used by the other units normally are placed on secondary storage devices (e.g., your hard drive) until they’re again needed, possibly hours, days, months or even years later. Therefore, information on secondary storage devices is said to be **persistent**—it is preserved even when the computer’s power is turned off. Secondary storage information takes much longer to access than information in primary memory, but the cost per unit of secondary storage is much less than that of primary memory. Examples of secondary storage devices include HDD, DVDs and flash drives (sometimes called memory sticks).

Personal, Distributed and Client/Server Computing

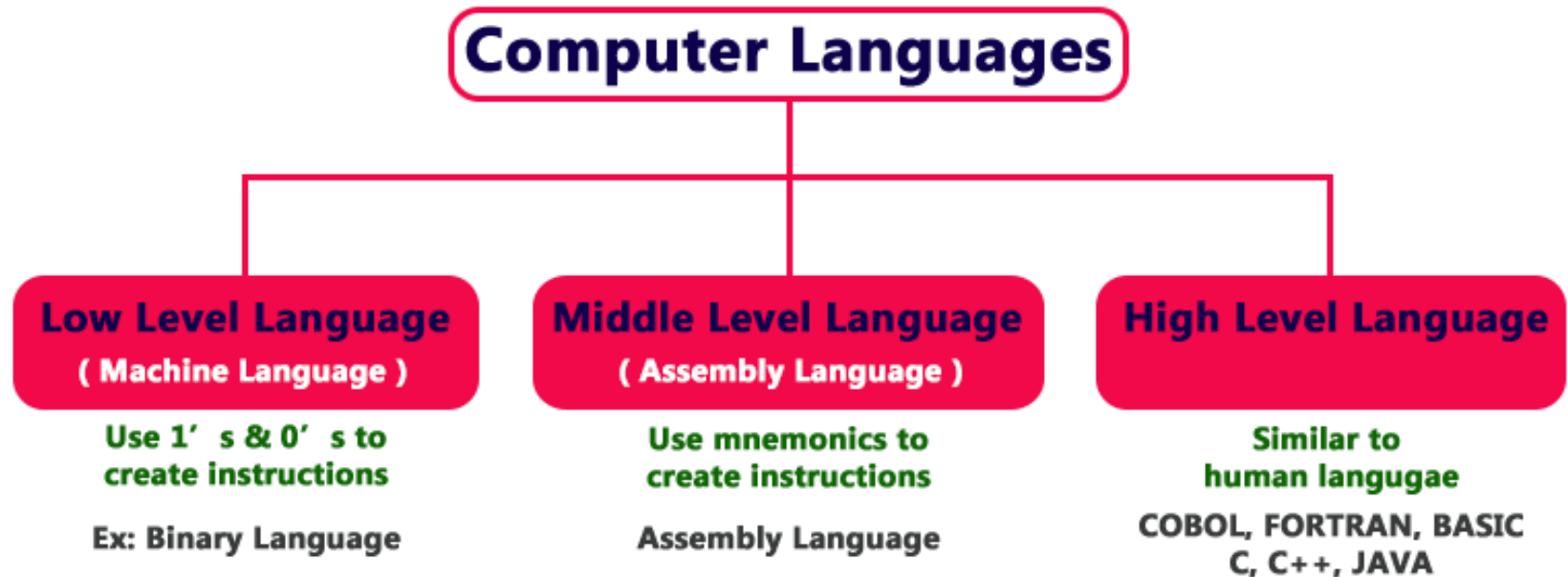
- In 1977, Apple Computer popularized **personal computing**.
- In 1981, IBM, the world's largest computer vendor, introduced the IBM Personal Computer (PC).
- These computers were “stand-alone” units—people transported disks back and forth between them to share information (this was often called “sneakernet”).
- These machines could be linked together in computer networks, sometimes over telephone lines and sometimes in **local area networks (LANs)** within an organization.

- This led to the phenomenon of **distributed computing**.
- Information is shared easily across computer networks, where computers called **servers** (file servers, database servers, web servers, etc.) offer capabilities that may be used by **client** computers distributed throughout the network, hence the term **client/server computing**.
- **C** is widely used for writing software for operating systems, for computer networking and for distributed client/server applications.

The Internet and the World Wide Web

- With the introduction of the **World Wide Web**—which allows computer users to locate and view multimedia-based documents on almost any subject over the Internet—the Internet has exploded into the world's premier communication mechanism.
 - Today's applications can be written to communicate among the world's computers.
-

Computer Languages



Machine Languages, Assembly Languages and High-Level Languages

- Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate **translation** steps.
- Any computer can directly understand only its own **machine language**.
- Machine language is the “natural language” of a computer and as such is defined by its hardware design.

- Machine language is often referred to as object code.
- Machine languages generally consist of strings of numbers (ultimately reduced to 1s and 0s) that instruct computers to perform their most elementary operations one at a time.
- Machine languages are **machine dependent** (i.e., a particular machine language can be used on only one type of computer).

- Such languages are cumbersome for humans, as illustrated by the following section of an early machine-language program that adds overtime pay to base pay and stores the result in gross pay:
 - +1300042774
+1400593419
+1200274027
- Instead of using the strings of numbers that computers could directly understand, programmers began using English-like abbreviations to represent elementary operations.
- These abbreviations formed the basis of **assembly languages**.

- **Translator programs** called **assemblers** were developed to convert early assembly-language programs to machine language at computer speeds.
- The following section of an assembly-language program also adds overtime pay to base pay and stores the result in gross pay:
 - load basepay
 add overpay
 store grosspay
- Although such code is clearer to humans, it's incomprehensible to computers until translated to machine language.

- Computer usage increased rapidly with the advent of assembly languages, but programmers still had to use many instructions to accomplish even the simplest tasks.
- To speed the programming process, **high-level languages** were developed in which single statements could be written to accomplish substantial tasks.
- Translator programs called **compilers** convert high-level language programs into machine language.
- High-level languages allow programmers to write instructions that look almost like everyday English and contain commonly used mathematical notations.

- A payroll program written in a high-level language might contain a statement such as
 - `grossPay = basePay + overTimePay;`
- C, C++, Microsoft's .NET languages (e.g., Visual Basic, Visual C++ and Visual C#) and Java are among the most widely used high-level programming languages.
- **Interpreter** programs were developed to execute high-level language programs directly (without the delay of compilation), although slower than compiled programs run.

History of C Language

- **C** evolved from two previous languages, **BCPL** and **B**.
 - **BCPL** was developed in **1967** by **Martin Richards** as a language for writing operating-systems software and compilers.
 - **Ken Thompson** modeled many features in his **B** language after their counterparts in **BCPL**, and in 1970 he used **B** to create early versions of the **UNIX** operating system at Bell Laboratories.
-

Cont..

- The **C** language was evolved from **B** by **Dennis Ritchie** at **Bell Laboratories** and was originally implemented on a DEC PDP-11 computer in **1972**.
- **C** initially became widely known as the development language of the **UNIX operating system**.
- Today, virtually all new major operating systems are written in **C++**.
- C is available for most computers.
- C is mostly hardware independent.

- By the late **1970s**, **C** had evolved into what is now referred to as “**traditional C**.” The publication in 1978 of Kernighan and Ritchie’s book, *The C Programming Language*, drew wide attention to the language.
- The rapid expansion of **C** over various types of computers (sometimes called **hardware platforms**) led to many variations that were similar but often incompatible.
- In **1989**, the **C standard** was approved; this standard was updated in **1999**.
- **C99** is a revised standard for the C programming language that refines & expands the capabilities of C.

C++ Standard Library

- As we will learn in the coming Chapters, C++ programs consist of modules or pieces called **functions**.
- You can program all the functions you need to form a C++ program, but most C programmers take advantage of a rich collection of existing functions called the **C++ Standard Library**.
- Visit the following website for the complete C++ Standard Library documentation, including the C features:
 - <https://en.cppreference.com/w/cpp/header>

- When programming in C++ you'll typically use the following building blocks:
 - ❖ C++ Standard Library functions
 - ❖ Functions you create yourself
 - ❖ Functions other people have created and made available to you.
 - ❖ Variables
 - ❖ Data types
 - ❖ Loops
 - ❖ Arrays

C++ History

- **C++** was developed by Bjarne Stroustrup at Bell Laboratories.
- It has its roots in **C**, providing a number of features that “spruce up” the C language.
- More important, it provides capabilities for **object-oriented programming**.
- **Objects** are essentially reusable software **components** that model items in the real world.
- Using a modular, object-oriented design and implementation approach can make software development groups much more productive than is possible with previous programming techniques.

Processing a C++ Program

```
#include <iostream>
using namespace std;
int main()
{
    cout << "My first C++ program." << endl;
    return 0;
}
```

Sample Run:

My first C++ program.

- To execute a C++ program:
 - ❖ Use an editor to create a source program in C++
 - ❖ Preprocessor directives begin with # and are processed by a the preprocessor
 - ❖ Use the compiler to:
 - Check that the program obeys the rules
 - Translate into machine language (object program)

- To execute a C++ program (cont'd.):
 - Linker:
 - Combines object program with other programs provided by the SDK to create executable code
 - Loader:
 - Loads executable program into main memory
 - The last step is to execute the program

Cont..

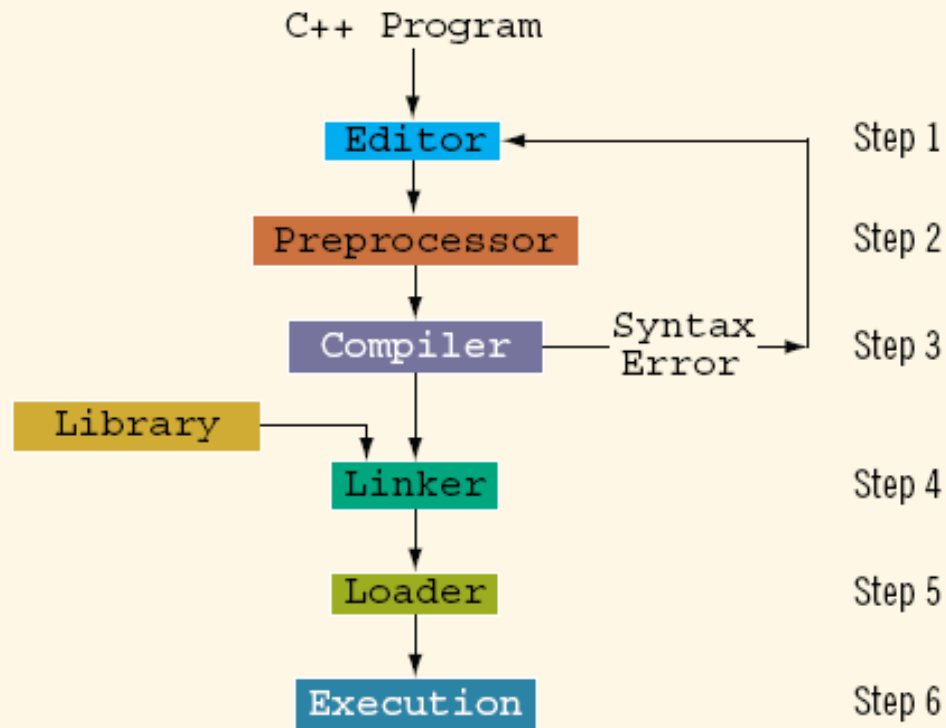


FIGURE 1-3 Processing a C++ program

Programming with the Problem Analysis– Coding–Execution Cycle

- Programming is a process of problem solving
- One problem-solving technique:
 - Analyze the problem
 - Outline the problem requirements
 - Design steps (algorithm) to solve the problem
- Algorithm:
 - Step-by-step problem-solving process
 - Solution achieved in finite amount of time

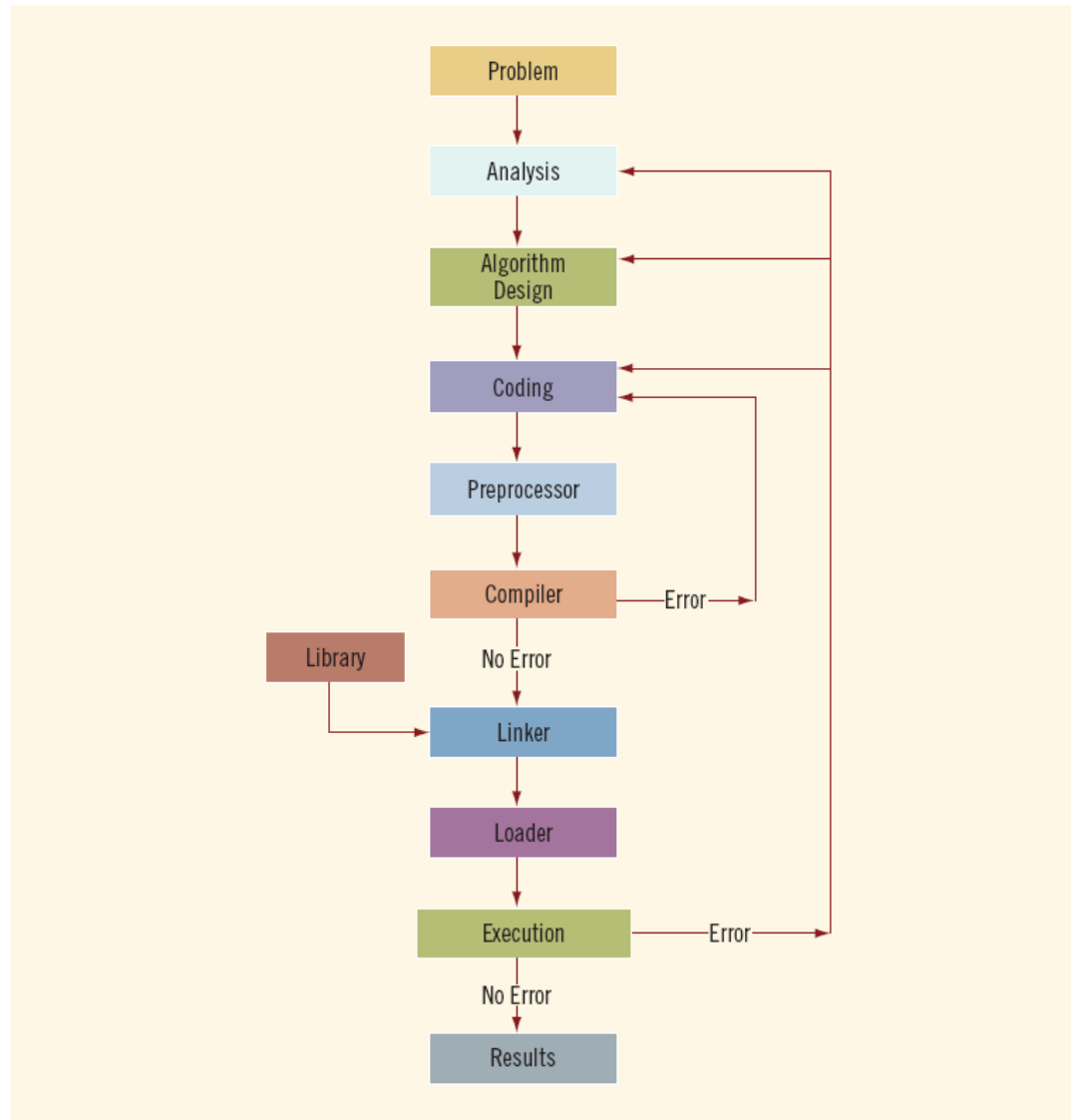


FIGURE 1-4 Problem analysis-coding-execution cycle

- Run code through compiler
- If compiler generates errors
 - Look at code and remove errors
 - Run code again through compiler
- If there are no syntax errors
 - Compiler generates equivalent machine code
- Linker links machine code with system resources

- Once compiled and linked, loader can place program into main memory for execution
- The final step is to execute the program
- Compiler guarantees that the program follows the rules of the language
 - Does not guarantee that the program will run correctly

Example

- Design an algorithm to find the perimeter and area of a rectangle
- The perimeter and area of the rectangle are given by the following formulas:

`perimeter = 2 * (length + width)`

`area = length * width`

■ Algorithm:

- ❑ Get length of the rectangle
- ❑ Get width of the rectangle
- ❑ Find the perimeter using the following equation:

$$\text{perimeter} = 2 * (\text{length} + \text{width})$$

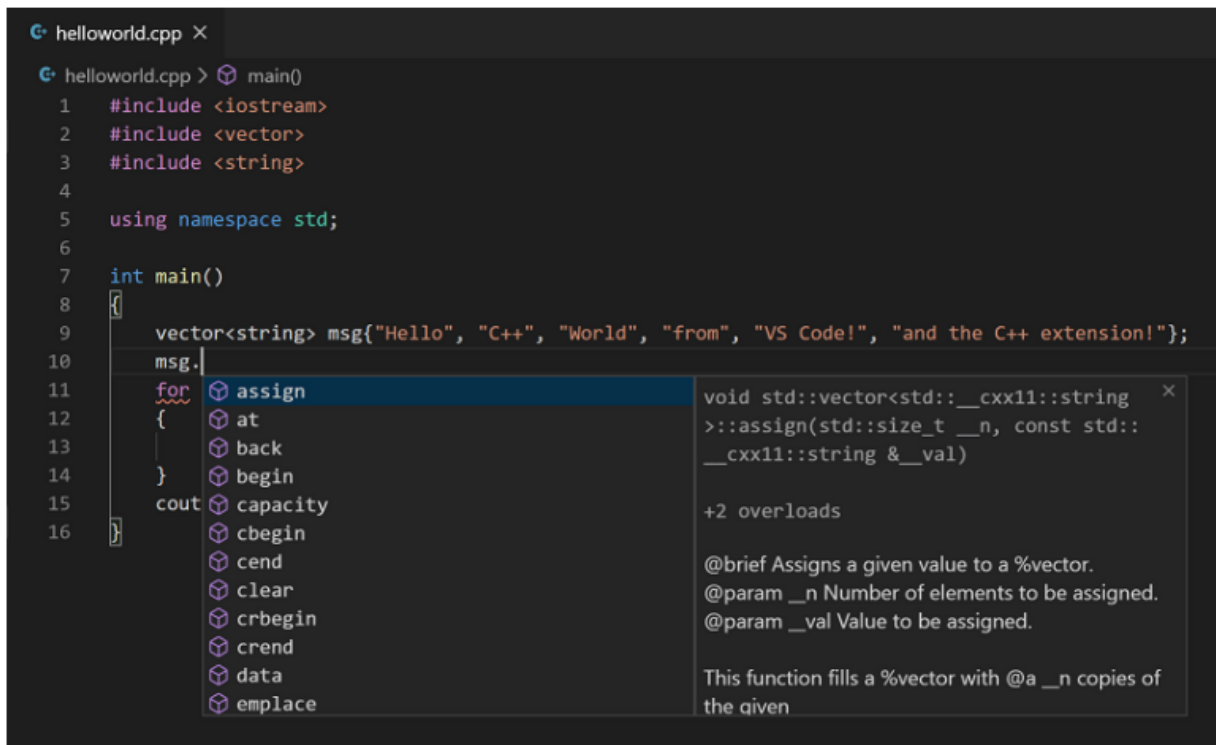
- ❑ Find the area using the following equation:

$$\text{area} = \text{length} * \text{width}$$

C++ Editors

Visual Studio Code (VS Code)

Visual Studio Code (VS Code) is an open-source, cross-platform source code editor created by Microsoft.



```
helloworld.cpp X
helloworld.cpp > main()
1  #include <iostream>
2  #include <vector>
3  #include <string>
4
5  using namespace std;
6
7  int main()
8  {
9      vector<string> msg{"Hello", "C++", "World", "from", "VS Code!", "and the C++ extension!"};
10     msg.
11     for { assign
12         { at
13             back
14         } begin
15     cout capacity
16     cbegin
17     cend
18     clear
19     crbegin
20     crend
21     data
22     emplace
```

void std::vector<std::__cxx11::string>::assign(std::size_t __n, const std::__cxx11::string &__val)

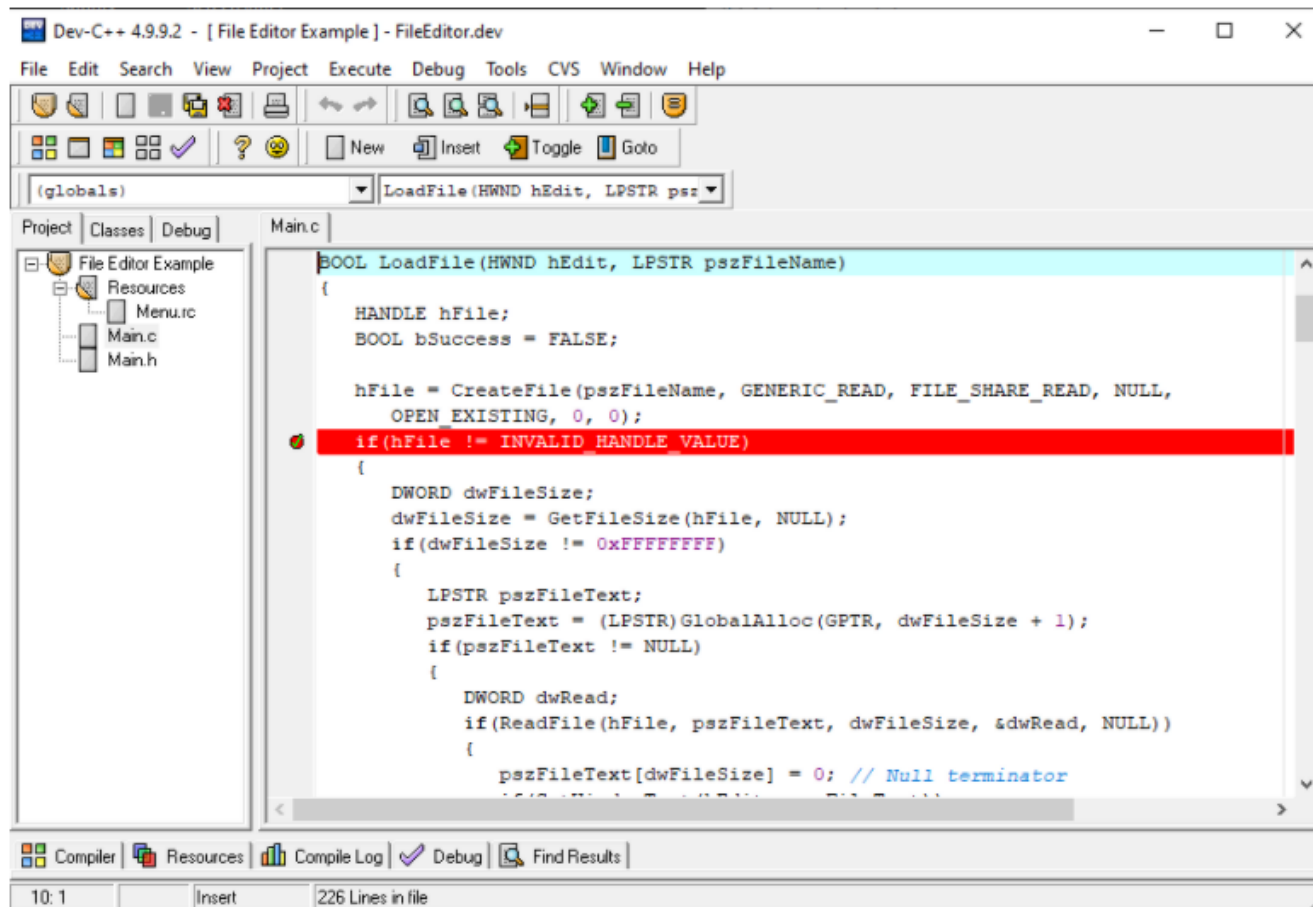
+2 overloads

@brief Assigns a given value to a %vector.
@param __n Number of elements to be assigned.
@param __val Value to be assigned.

This function fills a %vector with @a __n copies of the given

Cont..

Dev-C++



Real-World Applications of C++

Operating Systems

Apple OS



Microsoft Windows OS



Browsers

Mozilla Firefox



Google Applications



Thank You!
