# Introduction to Technical Programming

# Topic 4: Programming tools and utilities

* 1. **Building and running C/C++ applications**

Content:

* The C and C++ compiler
* Compiler artefacts
* Compiler design

*Learning* Outcomes*:*

*Students should be able to:*

* + 1. Define *the* term compiler
    2. Define the term source code
    3. Explain what a decompiler is used for
    4. Define the term interpreter
    5. Explain the difference between a compiler and an interpreter
    6. Explain what a binary is and when it is produced
    7. Explain the difference been C and C++ compiler
    8. List and Explain the basic three stage compiler design (Front Middle Back end)
  1. **C/C++ on the Desktop**

Content:

* C/C++ compiler environment on desktop PC
* IDE on desktop PC

*Learning Outcomes:*

*Students should be able to:*

* + 1. Install and configure C/C++ compiler on desktop PC
    2. Define the term IDE
    3. Explain what an IDE is used for
    4. Install and configure IDE on desktop PC
  1. C++ on the Raspberry Pi

Content:

* The GCC C/C++ compiler
* Geany on Raspberry Pi

Learning Outcomes:

Students should be able to:

4.3.1 Explain what the acronym GCC stands for

4.3.2 Explain what the GCC collection contains

4.3.3 List compilers included in the GCC

4.3.4 Explain the difference between GCC and MinGW

4.3.5 Install and configure GCC C++ compiler on Raspberry Pi

4.3.6 Install and configure Geany on Raspberry Pi

* 1. Debugging C/C++ applications

Content:

* Debugging
* GDB

Learning Outcomes:

Students should be able to:

4.4.1 Define the term debugging

4.4.2 Explain why an application needs to be debugged

4.4.3 List common debugging techniques (Interactive, Print, Remote)

4.4.4 Define the term breakpoint as it relates to debugging

4.4.5 Define the term stepping as it relates to debugging

4.4.6 Explain what the GDB tool is used for

4.4.7 Install and configure MinGW on desktop PC

4.4.8 Locate the GDB application in the MINGW installation

4.4.9 Add the MINGW bin directory to the operating system path environment variable

4.4.10 Compile source code wit the -g flag in IDE

* 1. Debugging C/C++ with GDBgui

Content:

* GDBgui
* C++ debugging

Learning Outcomes:

Students should be able to

4.5.1 Explain the concept of a debugger frontend

4.5.2 Install and configure gdbgiu on Desktop PC

4.5.3 Start gdbgui application

4.5.4 Load a compiled C/C++ binary (.exe)

4.5.5 Run the loaded binary

4.5.6 Step through the running binary line by line

4.5.7 Investigate (watch) the values of local variables

* 1. The Git Version Control System

Content:

Git concepts

4.6.1 Explain what Git is and what Git is used for

4.6.2 Discuss the three main goals of the Git VCS

* speed
* data integrity
* distributed

4.6.3 Explain the relationship between local Git directory and directory located on server

4.6.4 Discuss the software license used for Git

4.6.5 Name and explain the advantages and disadvantages to the software license used by Git

4.6.6 Explain why Git was created

4.6.7 Explain the relationship between Git and hosting providers like GitHub

4.6.8 List major open-source projects using Git

4.6.9 Install and configure Git on desktop PC

4.6.10 Install and configure Git on Raspberry Pi

4.6.11 List and explain common terminology associated with distributed VCS’s including:

* Branch
* Checkout
* Clone
* Stage
* Commit
* Conflict
* Head
* Repository
* Initialise

# Topic 4: Programming tools and utilities

After you have completed this module, you should be able to :

* Define *the* term compiler
* Define the term source code
* Explain what a decompiler is used for
* Define the term interpreter
* Explain the difference between a compiler and an interpreter
* Explain what a binary is and when it is produced
* Explain the difference been C and C++ compiler
* List and Explain the basic three stage compiler design (Front Middle Back end)
* Install and configure C/C++ compiler on desktop PC
* Define the term IDE
* Explain what an IDE is used for
* Install and configure IDE on desktop PC
* Explain what the acronym GCC stands for
* Explain what the GCC collection contains
* List compilers included in the GCC
* Explain the difference between GCC and MinGW
* Install and configure GCC C/C++ compiler on Raspberry Pi
* Install and configure Geany on Raspberry Pi
* Define the term debugging
* Explain why an application needs to be debugged
* List common debugging techniques (Interactive, Print, Remote)
* Define the term breakpoint as it relates to debugging
* Define the term stepping as it relates to debugging
* Explain what the GDB tool is used for
* Install and configure MinGW on desktop PC
* Locate the GDB application in the MINGW installation
* Add the MINGW bin directory to the operating system path environment variable
* Compile source code wit the -g flag in IDE
* Explain the concept of a debugger frontend
* Install and configure gdbgiu on Desktop PC
* Start gdbgui application
* Load a compiled C/C++ binary (.exe)
* Run the loaded binary
* Step through the running binary line by line
* Investigate (watch) the values of local variables
* Explain what Git is and what Git is used for
* Discuss the three main goals of the Git VCS
* Explain the relationship between local Git directory and directory located on server
* Discuss the software license used for Git
* Name and explain the advantages and disadvantages to the software license used by Git
* Explain why Git was created
* Explain the relationship between Git and hosting providers like GitHub
* List major open-source projects using Git
* Install and configure Git on desktop PC
* Install and configure Git on Raspberry Pi
* List and explain common terminology associated with distributed VCS’s

# Programming tools and utilities

## 4.1 Source Control

**Introduction**

In topic 1.2.1, we discussed about the different types of language translators. Translators convert programs written in high-level languages into machine code that a computer understands. To bring you up to speed, a language translator in software programming terms, is a generic term that could refer to a compiler, assembler, or interpreter. In this topic, we are going to explain more about the C or C++ compiler.

### 4.1.1 Revisiting version control

You will recall that we touched on version control in level 2. We mentioned that version control is used to track and control changes to source code. Version control systems are software tools that help software teams manage changes to source code over time. As part of this course, you will be required to develop applications or programs as individuals or in teams. If there is no system to track the changes done regularly, you might end up submitting a version which was developed earlier after several changes. In essence, keeping track of changes is difficult. We do know that you can do that manually by changing file names but still requires constant tracking of the names. Also rolling back will be a challenge. Version control comes in hand to deal with these issues. In this topic, we will do practical on using git technology as our version control software.

**Importance of version control systems**

* **Easy Modification of the codebase** -version control system makes it easier for continuous process of modifying programs.
* **Reverting Errors**- Emergency hot fixes, normal maintenance, upgrades, and new features with possibly conflicting development schedules can all benefit greatly from version control.
* **Collaboration**- Through version control, developers and programmers can collaborate easily. Version control enables team members to work independently and concurrently on the same document without interfering with each other's contributions.
* **Backup**- A distributed VCS acts as a backup.
* **Traceability**- A system called traceability shows all the revisions and modifications that have been performed throughout time.
* **Reduction Of Duplication and Errors-**Version control can reduce the duplication of multiple and/ or out-dated versions of any given document. As a result, it will lessen errors brought on by information that is shown inconsistently across many publications.

## 4.2 Building and running C/C++ applications

### 4.2.1 Define *the* term compiler

**VOCABULARY**

The compiler converts source code written in high-level language into machine code.

Because they must instantly transform higher-level code into lower-level machine language and save the executable object code to memory, compilers can be slow. A compiler generates machine code for a particular Instruction Set Architecture (ISA), which is unique to each processor. For instance, without a specific compiler, it is impossible to compile code for the x86 architecture and run it on the Million instructions per second (MIPS) architecture.

If the source code in a compiler is error-free, it will translate correctly to object code. If there are any mistakes in the source code, the compiler specifies the errors with line numbers at the end of the compilation. Before the compiler can correctly recompile the source code, the errors must be fixed. Examples of languages that uses compilers are C, C++, C#, Java,  Erlang, Haskell, Rust, and Go.

**Advantages of compilers**

1. **Independence**-No other program or application is required to run the executable file of source codes.
2. **Optimisation**-The compiled program is well optimized and runs faster.
3. **Data Security**-The compiler generates executable files that can be executed on any other system.
4. **Speed-**Compiler are faster than interpreter

**Disadvantages of Compilers**

1. **Hardware Specific**
2. **Time consuming-** Compilation process takes time before an executable file is produces
3. **Extra memory**-Because source code object code file and executable files
4. **Debugging Difficulty**- All errors are shown at once.

### 4.2.2 Define the term source code

In general, programming statements written by a programmer using a text editor or visual programming tool and then saved in a file are referred to as source code. When the source code is compiled using a C++ compiler, the output, a compiled file, is referred to as object code.

It's common to refer to a computer program's source code and object code as its "before" and "after" versions respectively. Since there is only one form of the code, the words source code and object code do not apply to script (noncompiled or interpreted) programming languages like JavaScript. Figure 4.1 illustrates the conversion from source code to object code.

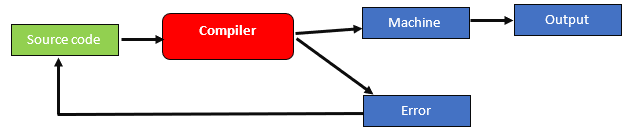


Figure 4. 1:Source code to object code

To write a source code, all you really need is a simple text editor like Notepad++, notepad or any other. This way, source code can be saved as plain text with the correct file name ending for the programming language. So, if you find a file with the ending “.cpp” on your hard drive, then it’s a text file, containing code in the C++ programming language.

Here is an example of C++ language source code:

**EXAMPLE 4.1**

#include <iostream>  
using namespace std;  
  
int main() {  
 cout << "Hello, World! \n" ;  
 cout << "This is a test program!" ;  
 return 0;  
}

It is obvious to everyone who isn't a computer programmer that the code on Example 4.1 has anything to do with the text being printed. "Hello World” and “This is a test program!" . When executed the output is as follows:

C:\ CLionProjects\untitled\cmake-build-debug\test.exe

Hello, World!

This is a test program!

Process finished with exit code 0

We are not going to explain the code for now, but we wanted to show what will be printed.

The object code file contains a sequence of machine-readable instructions that is processed by the CPU in a computer. Operating system or application software is usually in the form of compiled object code.

Here is a list of some popular C++ compilers/IDE

* [C++ Builder](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#1_C_Builder)
* [Microsoft Visual C++](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#2_Microsoft_Visual_C)
* [Eclipse IDE](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#3_Eclipse_IDE)
* [Codeblocks](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#4_Codeblocks)
* [Dev-C++](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#5_Dev-C)
* [NetBeans IDE](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#6_NetBeans_IDE)
* [Cygwin](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#7_Cygwin)
* [GCC](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#8_GCC)
* [MinGW](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#11_MinGW)
* [CodeLite](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#12_CodeLite)
* [Clang C++](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#14_Clang_C)
* [Clion](https://www.softwaretestinghelp.com/best-cpp-compiler-ide/#15_Clion)

**C++ Output**

In C++, the cout object belongs to the iostream object type. In the iostream header file, it is declared. It is used to display the output to the monitor, which is the typical output device. Using the insertion operator (), the information required to be displayed on the screen is added to the standard output stream (cout) using the insertion operator(<<). The **"c"** in cout refers to "character" and "out" means "output".

cout means "character output". The << operator can be used more than once with a combination of variables, strings, and manipulators (like endl). We are not going to go deeper into this for now. Example 4.1 demonstrate how to display text in quotes (string) using cout.

**Outputting numbers in C++**

To print the numbers, we use the same cout object but without using quotation marks. Look at the example 4.2

**EXAMPLE 4.2**

#include <iostream>

using namespace std;

int main() {

int number\_1=22;

int number\_2=19;

cout<<number\_1;

cout<<number\_2;

return 0;

}

OUTPUT

Graphical user interface, application

Description automatically generated

When you look at our program, you'll notice that the numbers 22 and 19 are not in quotation marks, so when we want to print them, we use the names of the memory spaces where these two values are stored, which in our case are number\_1 and number\_2 for values 22 and 19, respectively.

Our only challenge is now that the two numbers are printed joined together (concatenated) We can rewrite the output statements as follows:

cout<<number\_1<< endl;

cout<<number\_2<< endl;

A new line is added using the endl manipulator. For this reason, each output is shown in a separate line.

The << operator can be used more than once if we want to print different variables, strings and so on in a single statement. For example:

cout<<"The first number is : " << number\_1<< endl;

cout<<"The second number is : "<<number\_2<< endl;

**Output**

The first number is : 22

The second number is : 19

**Inputting data to the C++ program**

When using a conventional input device, like the keyboard, C++'s cin function accepts structured input. The >> operator and the cin object are used to take input. Let us look at example 4.3 which asks the user to input a number.

**EXAMPLE 4.3**

#include <iostream>

using namespace std;

//main function

int main() {

int num;

//displaying output

cout << "Enter an integer: ";

//accepting input

cin >> num; // Taking input

//displaying out

cout << "The number is: " << num;

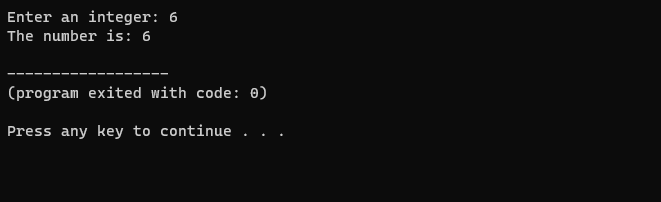
return 0;

}

**Output**



Type in any number for instance 6. The final output will look as follows:



In the program, we used cin >> num; to take input from the user. The input is stored in the memory location called num. We use the >> operator with cin to take input.

**Accepting multiple inputs**

Let’s assume in our previous code we wanted to accept two numbers. What it means is these numbers must be stored in two memory locations and when we accept each number will be stored in its own memory location as follows:

**EXAMPLE 4.4**

#include <iostream>

using namespace std;

//main function

int main() {

int num, num2;

//displaying output

cout << "Enter an integer: ";

//accepting input

cin >> num >>num2; // Taking input

//displaying out

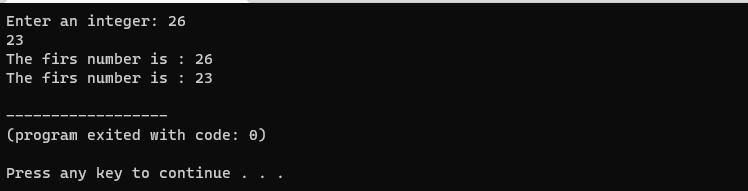
cout << "The firs number is : " << num << endl;

cout << "The firs number is : " <<num2;

return 0;

}

**Output**



### 4.2.3 Explain what a decompiler is used for

In software programming, a decompiler converts executable programs into computer code that can be understood by programmers.It performs the operations of a compiler, which translates source code into an executable format, but in reverse. A decompiler’s target is the user , whereas the compiler’s target is the machine.

**NOTE**

To convert a programming source code into a working program, it must first be compiled -- i.e., converted into a series of binary bits or digits (that is, 1s and 0s) that can be understood by the computer. This operation can be reversed by decompiling the final program (which is why decompiling is described as a type of **reverse-engineering)**.

Decompilation can be used unethically to copy source code for reuse or adaption without the owner's consent. A decompiler can be useful in some cases for the following purposes:

* Recovery of lost source code to archive or maintain the code
* Debugging programs
* Antivirus capability to find vulnerabilities in the program

Some common examples of decompilers includes:

* IDA Pro
* Hex-Rays Decompiler
* CFF Explorer
* Hiew

### 4.2.4 Define the term interpreter

**VOCABULARY**

An interpreter program executes other programs directly, running through program code and executing it line-by-line.

An interpreter is a program that also translates a high-level language into a low-level language, but it does so while the program is running. You write the program in a text editor or something similar, then tell the interpreter to run it. It runs the program one line at a time, translating each line as it goes: it translates the first line and runs it, then the second line and runs it, and so on. Python and BASIC are good examples of an interpreted language.

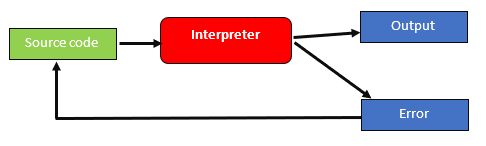


Figure 4. 2:How interpreters work

**Advantages Of Interpreter**

1. **Cross platform**- We can directly share the source code in interpreted languages, which can run on any system without any system incompatibility issues.
2. **Easier to debug**- Interpreters make code debugging easier because they read the code line by line and return the error message immediately.
3. **Less Memory** and steps- Interpreters change code into output files.
4. **Execution control**- Because an interpreter reads code line by line, you can halt execution and edit the code at any point, which a compiled language does not allow.

**Disadvantages Of Interpreter**

1. **Slower**
2. **Dependencies file required-**The interpreter is often slower than the compiler because it reads, analyses, and converts the code line by line.
3. **Less Secure-** Interpreters do not generate executable files, so to share the program with others, we must share our source code, which is insecure and private.

### 4.2.5 Difference between a compiler and an interpreter

Table 4. 1: Comparison of a compiler vs an interpreter

| **Compiler** | **Interpreter** |
| --- | --- |
| A compiler translates the entire source code in a single run. | An interpreter translates the entire source code line by line. |
| Compilers are faster than an interpreter. | Interpreters are slower than the compiler. |
| CPU utilization is more. | CPU utilization is less as compared to the compiler. |
| Both syntactic and semantic errors can be checked simultaneously. | Only syntactic errors are checked. |
| The compiler is larger. | Interpreters are often smaller than compilers. |
| The localization of errors is difficult. | The localization of error is easier than the compiler. |
| The compiler is used by the language such as C, C++. | An interpreter is used by languages such as Java and Python. |

### 4.2.6 Explain what a binary is and when it is produced

Data in a binary file consists of a series of eight-bit bytes that are sequential and contain data in a binary format. The material needs to be interpreted by a software program or hardware processor that is aware of how it is formatted and how to read the data in advance. Among the many different file kinds that can be stored as binary files are executables, libraries, images, databases, archives, and many more.

A header identifying the kind of file is frequently present in binary files.. Application developers and other software developers commonly use binary files. An executable program is generally identified as a binary file with an extension such as .bin or .exe. Binary files are often described as executables or as compiled applications. Files of this type are also known as object code.

### 4.2.7 Key differences between C and C++ compiler

Applications, games, databases, operating systems, and games are commonly developed in C and C++. Although C and C++ sound similar, their features and usage differ. The C programming language supports objects and classes. C++, on the other hand, is an enhanced version of C that supports object-oriented programming. C is a procedural-oriented language and C++ is an object-oriented language, which is the most important difference between them. Table 4.2 demonstrates the key differences between C and C++ programming languages.

Table 4. 2: Differences between C and C++ programming

|  |  |  |
| --- | --- | --- |
| **Key** | **C** | **C++** |
| Developer | C developed by Dennis Richie between 1969-1973 at AT&T Bell Labs. | C++ developed by Bjarne Stroustrup in 1979. |
| Object Oriented Programming | C does not support OOPS concepts like polymorphism, encapsulation, and inheritance. C is procedural. | C++ is a object oriented programming and supports OOPS concepts like polymorphism, encapsulation, and inheritance. C++ supports both procedural and OOP. |
| Function | C is function driven language. | C++ is Object driven language. |
| File extension | The file extension of a program in C language is .c. | The file extension of a C++ program is .cpp. |
| Header | C uses <stdio.h> header file for input/output operations. | uses <iostream> header file for input/output operations. |

### 4.2.8 Basic three stage compiler design (Front Middle Back end)

A compiler typically has six states, and the high-level language source code that it receives is processed by each state individually to produce machine-understandable code or object code as an output. The three main stages of the compiler are backend, middle end and front end.

**Front End Phase**- Phases or parts of phases in the front end depend on the source language and are independent of the target machine. The main activity involves analysis of source code into intermediate code. These generally consist of lexical analysis, semantic analysis, syntactic analysis, symbol table creation, and intermediate code generation.

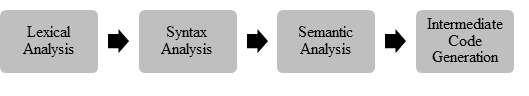


Figure 4. 3: Front End Phase

**Lexical Analysis** -Convert a program into sequence of tokens.

**Syntax Analysis** - Recover the structure portrayed by utilizing arrangement of tokens from past scanner.

**Semantic Analysis-**Ensures program has a well-defined meaning.

**IR Generation-**Compilers generate an explicit low-level representation that should be simple to provide and understood by the target machine.

**Code Generation-** The optimised intermediate code is converted by the compiler to machine code specific for the target machine.

**Middle End**

Independent of the desired CPU architecture, the middle end optimizes the intermediate representation (IR). Middle end performs target-independent optimizations such as removal of useless or unreachable code, constant propagation, hoisting code from out of a loop, loop unrolling, etc.

**Back End**

The back end of compilers includes the parts that are independent of the source language and dependent only on the target computer. A number of back-end tasks include code generation, code optimization, error handling, and symbol table maintenance.

Figure 4. 4: Back End compiler phase

In summary, we have learnt about compilers and their duty being to change source code in low level machine code at once. Compliers consists of front, middle and back end. The middle takes the output of the front end, transforms it somehow, then feeds that to the back-end. The [front end converts the source code to an intermediate representation](https://opensource.com/article/18/4/introduction-python-bytecode) which is not language specific. The backend converts its input to machine instructions for the target actual or virtual machine.

# FORMATIVE ASSESSMENT 4.1 INDIVIDUAL TASK

4.1.1 Define the term compiler. (2)

4.1.2 List THREE examples of compiled programming languages. (3)

4.1.3 Explain what back-end of a compiler entails. (3)

4.1.4 List 3 advantages and 3 disadvantages of compilers. (6)

4.1.5 Define the term interpreter. (2)

4.1.6 With the aid of diagrams, differentiate a compiler and interpreter. (12)

**Total :28 Marks**

## 4.3 C/C++ on the Desktop

**Introduction**

In section 4.2.2, we listed some of the C++ compilers and some come bundled with the IDE. For windows platforms, the common two options are CygWin and MingW.

The [CygWin](http://www.cygwin.com/) package provides a Portable Operating System Interface (Posix) layer in Windows, allowing much Unix software to be compiled and run without modification. When you install CygWin, you get a good command shell (bash), which may be useful. The g++ compiler can also be selected from the CygWin setup utility.

A disadvantage of CygWin is that the compiler produces code that relies on the Posix layer, which in turn is provided by a CygWin DLL. Binary executables produced by the CygWin port of g++ can only be run on other machines that have a copy of that Dynamic-link library (DLL).

A different version of the g++ compiler that does not require a unique DLL is offered by MinGW... You may have used this already, as this is the g++ typically shipped with Code::Blocks and other basic Windows Integrated Development Environment (IDEs).

### 4.3.1 Install and configure C/C++ compiler on desktop PC

Steps to Install C++ Compiler

1. To install C++ compiler, search for Code::Blocks in your browser.
2. On the downloads page, select download the binary release
3. Move to the Microsoft downloads section. In our case we will select the Code::Blocks IDE and the compiler. At the time of writing this book, the latest version was codeblocks-20.03mingw-setup.exe. MingW is our C++ compiler.
4. Open and run the downloaded file and it will install the IDE plus the compiler
5. You will see the following screen.

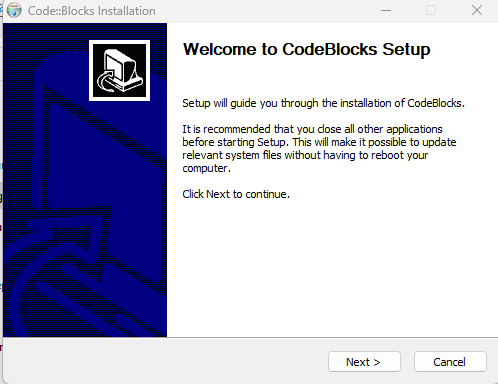


Figure 4. 5: CodeBlocks installation screen

Agree to the licence. It will also indicate the default compiler as GNU GCC compiler.

1. You will need to be very carefully with the path of MingW otherwise in some instances you run the program and it doesn’t pick up the compiler.
2. If this happens open Code::Blocks, click on compiler and select the Toolchain executables. Identify the path of your MingW and you will be good to go. Congratulations your C++ compiler is now installed

**Verify if compiler exists**

Go to the folder with MingW and type the following command: g++ --version

The output will be as follows:

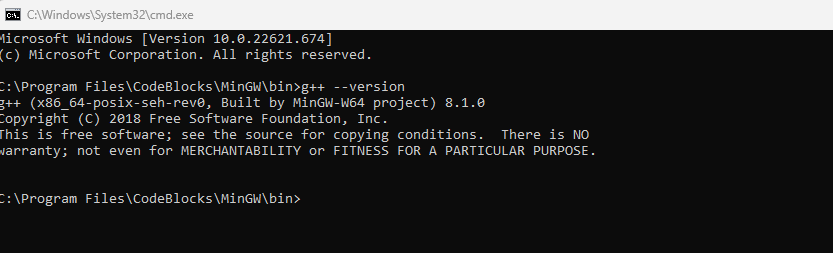


Figure 4. 6: Verifying the compiler

Type the following commands also:

* gdb --version
* make –version

### 4.3.2 Define the term IDE

**VOCABULARY**

An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. IDE normally consists of:

* compiler
* Source code editor
* A debugger

### 4.3.3 Use of IDE

A source code editor, build automation tools, and a debugger are the typical components of an IDE. Most contemporary IDEs offer intelligent code completion. Compiler, interpreter, or both may be included in some IDEs.  There are occasions when a version control system or different technologies to make creating a graphical user interface (GUI) easier are included. For usage in object-oriented software development, many contemporary IDEs additionally include a class browser, an object browser, and a class hierarchy diagram. Designed to encompass all programming tasks in one application, one of the main benefits of an IDE is that they offer a central interface with all the tools a developer needs, including:

* **Code editor:** Designed for writing and editing source code, these editors are distinguished from text editors because work to either simplify or enhance the process of writing and editing of code for developers
* **Compiler:** Compilers transform source code that is written in a human readable/writable language in a form that computers can execute.
* **Debugger:** Debuggers are used during testing and can help developers debug their application programs.
* **Build automation tools:**These can help automate developer tasks that are more common to save time.

### 4.3.4 Install and configure IDE on desktop PC

We have already demonstrated how we can install Code::Blocks which is an example of an IDE. As mentioned, another IDE which can also be use is VSCode, Eclipse, Netbeans, Geany, Komodo IDE and CLion. There are several IDEs out there and usage depends on user or organisation preferences. These IDE allow you to run languages such as C, C++, Java, Fontran and many others.

**Benefits of using IDEs**

* Faster set up
* Faster development tasks
* Continual learning
* Standardisation

We are going to demonstrate how to create a Hello World using Code::Blocks IDE.

1. To start a project, select Code::Blocks icon from the desktop or just search for Code.

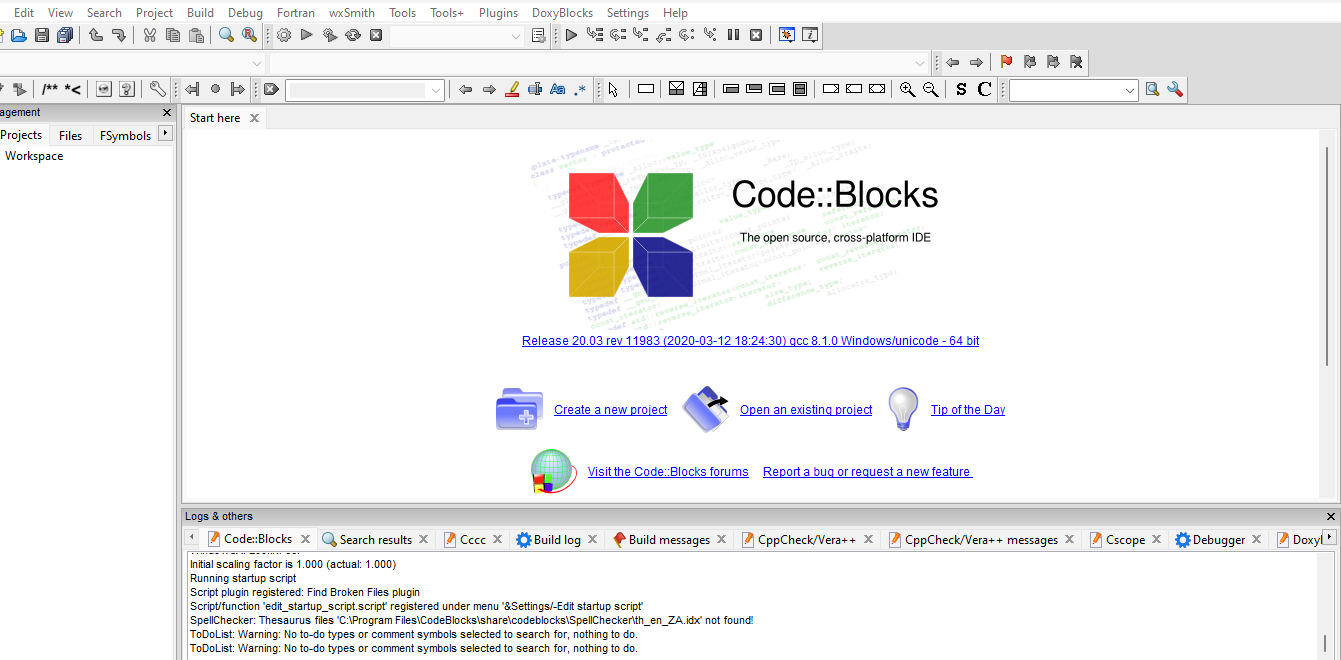


Figure 4. 7 : Code::Blocks Screen

1. Select create a new project and you will get a screen like the one on Figure 4.7 and select console application.

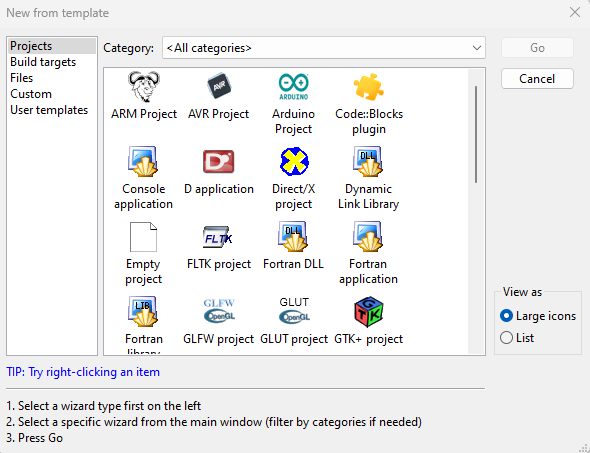


Figure 4. 8: Code::Blocks Template Window

1. Select the C++ programming language. Select the folder where the project will be saved and name of the project.
2. Navigate to the Projects workspace under the Management window and click the + sign on the Sources folder. This will open the main.cpp

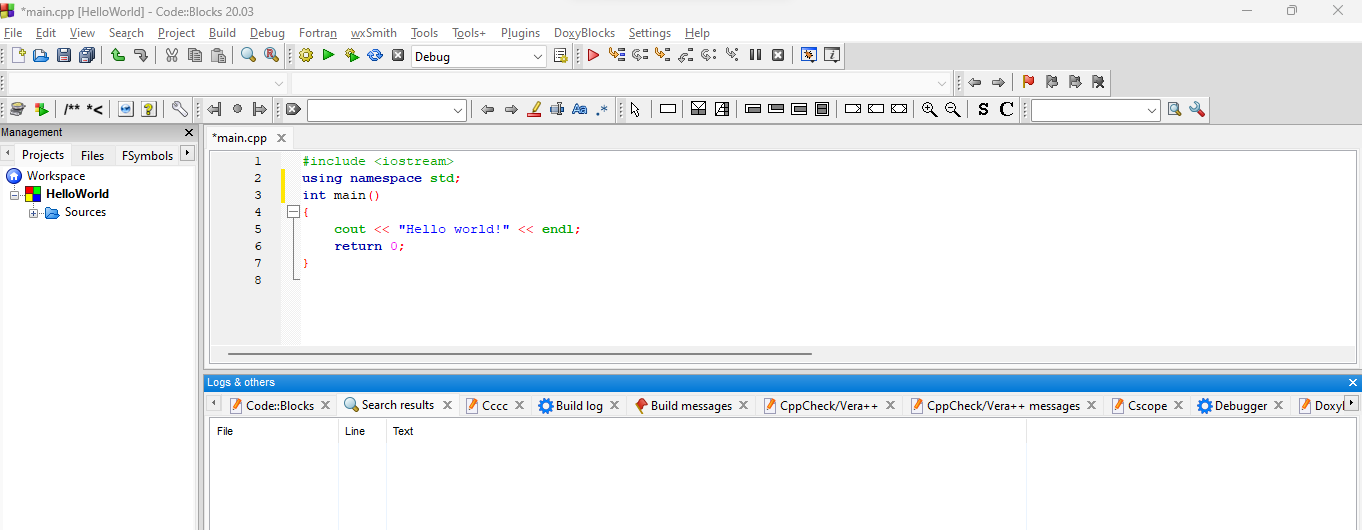


Figure 4. 9: Hello World sample project

To run the project, select build and click build and run. The sample output will be as shown in Figure 4.9.

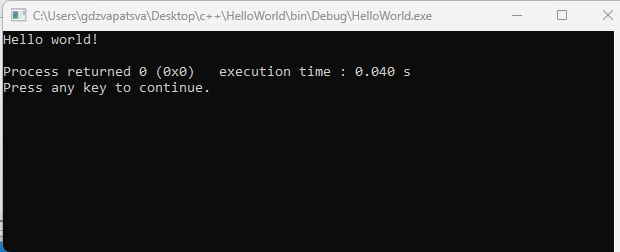


Figure 4. 10: Sample output of a C++ program

# FORMATIVE ASSESSMENT 4.2 GROUP TASK

**4.2.1** As part of software installation, you are expected to be able to install any IDE and the C++ compiler. We have listed a lot of IDE’s such as Code::Blocks, Geany

Task.

In groups, you are required to uninstall CodeBlocks and Mingw.

Reinstall Code::Blocks and Mingw

Test if gcc/g++ are available.

Run a simple hello world program.

(20)

**Total :20 marks**

## 4.4 C/C++ on the Raspberry Pi

**Introduction**

We have already introduced compiler in C++. We are now going to discuss the GCC compiler.

### 4.4.1 Explain what the acronym GCC stands for

The GNU Compiler Collection, or GCC, is a collection of compilers and development tools that are available for a variety of operating systems. GCC was created by the Free Software Foundation (FSF) and made available as totally free (as in libre) software.

### 4.4.2 GCC collection contains

GCC includes support primarily for C and C++ and includes Objective-C, Ada, Go, Fortran, and D. The Free Software Foundation (FSF) wrote GCC and released it as completely free (as in libre) software. Developers can take advantage of a wide range of capabilities provided by GCC, such as cross-platform compilation, deep code optimization, and support for different languages.

### 4.4.3 List compilers included in the GCC

gcc comes with a compiler named g++ that specifically compiles C++ programs, regardless of the file extension. Sometimes you need to use the C++ compiler even though the file extension is not a C++ extension; in this case you need to use g++.

### 4.4.4 Explain the difference between GCC and MinGW

[GCC](https://gcc.gnu.org/) is a free and open-source compiler for C and C++ (and other languages like Objective-C, Fortran, D). MinGW-w64 is an open-source C library that targets both 32-bit and 64-bit Windows platforms. 'MinGW' refers to the "Minimalist GNU for Windows" project. Even though GCC and MinGW-w64 can be used on other platforms to generate Windows executables, the WinLibs project only focusses on building versions that run natively on Windows. MinGW includes the GCC port, GNU Binutils for Windows, which is a set of freely distributable static import libraries and header files that enable the usage of the Windows API, as well as a Windows native build of the GNU Project's GNU Debugger and other utilities.

### 4.4.5 Install and configure GCC C/C++ compiler on Raspberry Pi

Raspberry Pi 4 devices already have the stock GCC on it and that should include the G++. If yours doesn’t there is a command, you can invoke.

Type the following command:

sudo apt install build-essential

Alternatively, follow the steps below:

Step 1 Update raspberry Pi by typing the following command

sudo apt update && sudo apt upgrade -y

Step 2: Make sure git is installed otherwise follow steps to install git. You can check if git is installed by typing the following command:

git –version

Step 3: In this step you need to open a terminal and download a binary Gcc.

git clone <https://bitbucket.org/sol_prog/raspberry-pi-gcc-binary.git>

Step 4: Now you need to extract the archive, then move the extracted compilers to /opt and remove the repository. Use the following commands

cd raspberry-pi-gcc-binary

tar -xjvf gcc-10.1.0-armhf-raspbian.tar.bz2

sudo mv gcc-10.1.0 /opt

cd ..

rm -rf raspberry-pi-gcc-binary

**Difference between GCC and G++**

Table 1: Differences between G++ and GCC

|  |  |  |
| --- | --- | --- |
| **Parameter** | **G++** | **GCC** |
| **Uses** | G++ is used for compiling C++ | Used for compiling C |
| **Compilation** | can compile either.cpp or.c files, but they will only be treated and released as C++ files. | can compile either.cpp or.c files, but they will be treated and released as C++ or C |
| **Command for Compilation** | g++ fileName.cpp -o binary. | gcc fileName.c -o binary. |
| **File Linking in Library** | When we use the G++ command to link the object files, the files automatically link in the standard C++ libraries. | It does not happen in the case of GCC. |

### 4.4.6 Install and configure Geany on Raspberry Pi

Geany is a very lightweight code editor or IDE. Due to its extensive capability and effective use of computer resources like CPU and, most importantly, RAM, Geany is ideally suited for programming on the Raspberry Pi. On Raspberry Pi 4, Geany comes pre-installed.

**Steps to install Geany on raspberry Pi**

Step 1: Open a terminal and run the following command:

sudo apt-get install geany

**Geany Features**

* Lightweight and fast
* Split screen editing
* Broad cross platform support
* Extensibility through plugins

# FORMATIVE ASSESSMENT 4.3 INDIVIDUAL TASK

4.3.1 Explain what GCC is. (2)

4.3.2 Differentiate MingW and GCC (2)

4.3.3 Define the term IDE. (2)

4.3.4 Write the steps to install Geany on a Raspberry Pi. (4)

4.3.5 Complete the following table to differentiate G++ from GCC. (8)

|  |  |  |
| --- | --- | --- |
| **Parameter** | **G++** | **GCC** |
| **Uses** |  |  |
| **Compilation** |  |  |
| **Compilation Command** |  |  |
| **File Linking in Library** |  |  |

**Total : 18 marks**

## 4.5 Debugging C++ applications

**Introduction**

When programmers code programs, they expect the code to work according to their plans. However, source code often breaks or does not work as anticipated. When this happens, programmers need to find where the problem is so that the code gives them the output they expect. Debugging tools are used to execute the program in a controlled setting, check the code step by step, and identify and correct the problem.

### 4.5.1 Define the term debugging

**VOCABULARY**

Debugging is the process of identifying and resolving errors or flaws in software source code.

There are two main categories of errors that need debugging:

1. **Compile-time**: These occur due to misuse of language constructs, such as syntax errors. Normally easy to find by using compiler tools and warnings to fix reported problems.
2. **Run-time**: These are much harder to figure out, as they cause the program to generate incorrect output (or “crash”) during execution. Example is a runtime error.

Programmers will check for the following during the debugging process, which can be carried out manually or automatically using software debugging tools:

* Syntax errors- A syntax error is a bug that occurs when a computer program has an incorrectly typed statement. Here is an example of a syntax error:

Graphical user interface, application

Description automatically generated

Figure 4. 11:Syntax error

Looking at Figure 4.11, notice the squiggly red line in line 1. The C++ word which is causing the error is <*iosteam*>. The correct spelling is supposed to be <iostream>.

* Runtime errors-occurs during runtime and could be caused by inputting wrong data type. For instance, the program which is supposed to accept a integer (number) and user types in a string.
* Logic errors-occurs when program runs but gives incorrect results

Testing and debugging work together to make sure software programs function as intended. Programmers test after finishing a segment or a portion of code to find faults and problems. Once flaws are identified, programmers can start the debugging process and try to clean up any errors in the software.

### 4.5.2 Reasons for debugging

1. Any software which is developed needs to be bug-free before releasing.
2. Competitive advantage-Bug free programs will build a good reputation for the software firm
3. Saves Time**-** Performing debugging at the initial stage saves the [time of software developers](https://www.educba.com/career-as-a-software-developers/).
4. Making sure that the software programs are serving the intended purpose

### 4.5.3 Common debugging techniques

1. Print debugging - The developer checks the flow of print statements by watching live or recorded print statements. This process is also called tracing
2. Interactive debugging- Using debugger tools, interactive debugging enables the processing of an application's code execution one step at a time and pauses it to inspect or modify application state.
3. Remote debugging allows developers to debug one or more programs on a remote machine while the IDE is only running on your local machine.

**Debugging Strategies**

Programmers employ a variety of strategies to avoid errors and shorten the debugging process. The following are a few debugging strategies that programmers frequently use.

1. Backtracking - The programmer works backwards through the source code, starting at the statement where an error symptom is found.
2. Cause Elimination Method- To identify the source of a point of failure, the software engineer compiles a list of likely causes for errors and conducts tests.
3. Shotgun Debugging - a developer's educated guesses-based debugging approach that relies on trial-and-error and crossing one's fingers.
4. Program Slicing- A series of program statements are run in the program under certain circumstances by quality assurance testers.

### 4.5.4 Define the term breakpoint as it relates to debugging

A breakpoint is a deliberate halt in an application's code by the developer for debugging purposes. This enables the programmer to inspect the internal state of the application at that time. We are going to discuss two types of breakpoints:

**Conditional breakpoints**- Conditional breakpoints allow developers to break inside a code block when a defined expression evaluates to true. The breakpoint is triggered when something in memory meets a given condition.

**Action breakpoints**- these allows us to take an action like printing to the console when a breakpoint is at a given spot.

Note: Everything that you can do in breakpoints can still be done by the code.

**How to set a conditional breakpoint in Code::Blocks**

Breakpoints are set in the normal editor window by clicking on the “gutter” on the left-hand side of the source file and a red dot will appear.

The following breakpoint commands are available under the Debug menu:

* Debug->Toggle Breakpoint: set or clear a breakpoint at the cursor
* Debug->Remove all breakpoints: remove all breakpoints from the project. A programmer can also edit the breakpoint.

A disabled breakpoint is useful to temporarily ignore a breakpoint during a debug session without removing it.

Let us do a simple practice on debugging. We are going to use the following code which adds two numbers and display the result.

**EXAMPLE 4.5**

Step 1: Type the following code and save it.

Text

Description automatically generated

Figure 4. 12:Adding breakpoint

Step 2:Right click on line 6 and click add a breakpoint. You will see a red dot like the one in Figure 4.12

Step 3: Select Debug and click Start/Continue. Immediately you will see the console appearing without any text.

NOTE: You can set up the screen display to tile vertically so that the console and the IDE are side by side as shown in Figure 4.13.

Step 4:Select Debug and click Step into. Immediately, you will see a small triangle inside the red dot.

Step 5: Select Next Line and you will see the console displaying “Enter first number”. Enter a number in the console and press enter. Return to the IDE and select Next Line. Repeat until the steps until the yellow triangle gets to line 15 and exit.

The Message window will display line numbers as the process at each point of execution and the resulting output. See Figure 4.13.

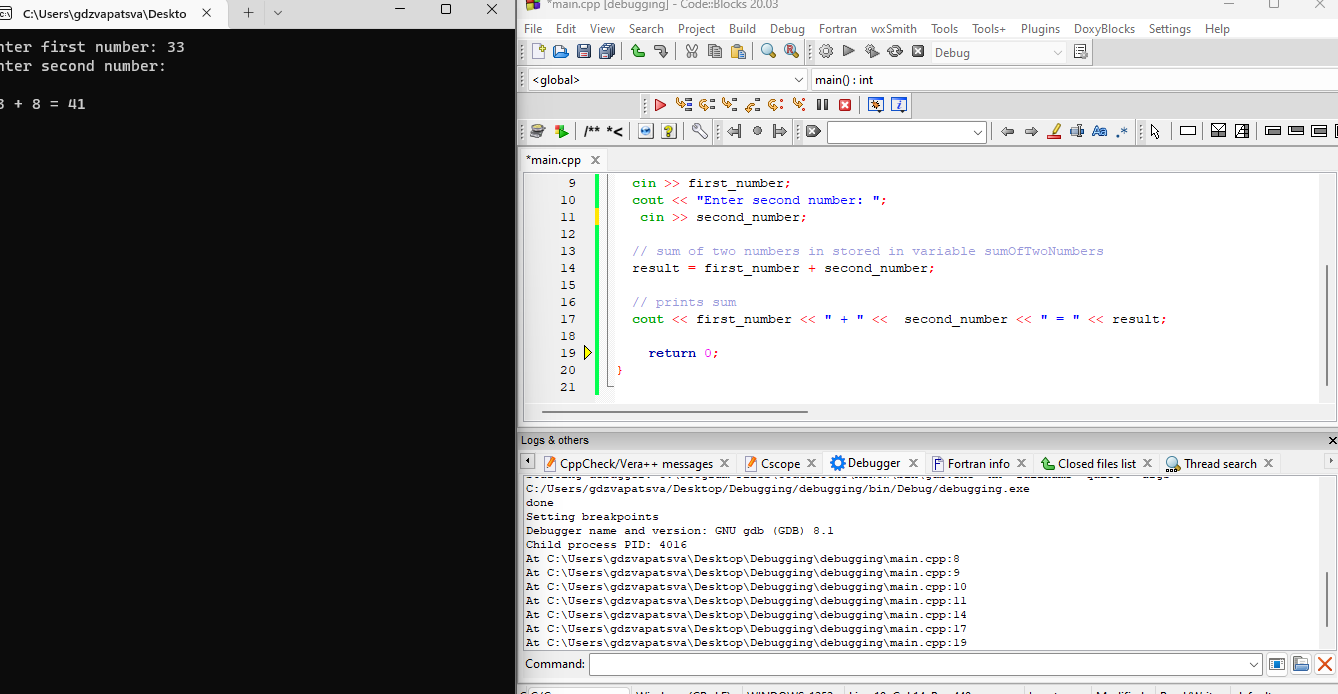


Figure 4. 13:Debugging Output

### 4.5.5 Define the term stepping as it relates to debugging

Stepping is a debugging approach in which code is executed a single instruction at a time. A debugger, among other things, allows the programmer to step through code in execution, set breakpoints, view memory, change variables, and track variables. The gnu debugger is a popular serial debugger (gdb). We will discuss the GDB in section 4.5.6.

### 4.5.6 Explain what the GDB tool is used for

GDB stands for GNU Debugger and is a powerful debugging tool for languages such as C++, C and others. You can step through the code, [set breakpoints](https://www.sciencedirect.com/topics/engineering/set-breakpoints), examine and change variables, and so on. GDB is command line only, which makes it difficult to use. GDB's graphical front ends translate GUI commands into GDB text commands.

When a C++ program crashes, developers can view the crash logs by using gdb. GDB works with executable files produced during the compilation process.

To check if gdb is installed, in our case we will locate it in the MingW/bin folder since it comes packed within the IDE. So, we can navigate to that folder and type the following command:

gdb –help

The output will be as follows:

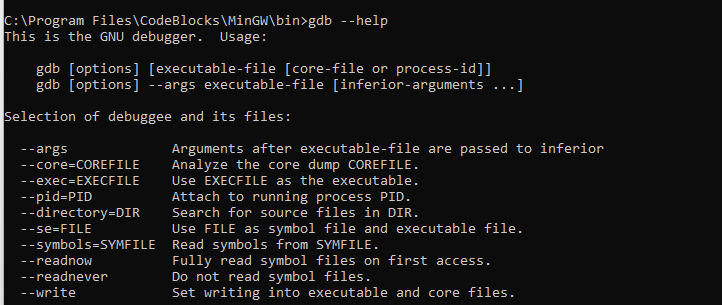


Figure 4. 14:gdb output

### 4.5.7 Install and configure MinGW on desktop PC

As you witnessed during Code::Blocks installation, you have a choice of selecting CodeBlocks with MingW installation file or without. In our case we selected one with MingW.

In case you uninstall MingW or you need to install it separate from the IDE, here are the steps:

**Step 1**: Go to the [MinGW](https://sourceforge.net/projects/mingw/) website (https://sourceforge.net/projects/mingw/) and click on Download.

**Step 2**: Double click and open the exe MinGW file and click install.

**Step 3:**When you click Continue, the MinGW installation manager will appear.

**Step 4**: Right-click on each option in the installation manager and then select Mark for installation. Click Apply changes and close. Now MingW is installed.

### 4.5.8 Locate the GDB application in the MINGW installation

To locate the gdb application from MingW, navigate to MingW\bin and you will find the application in there. Alternatively, you can type the following command in the terminal:

dir "\gdb.exe\*" /s

The output of the following code will be as shown in Figure 4.11.

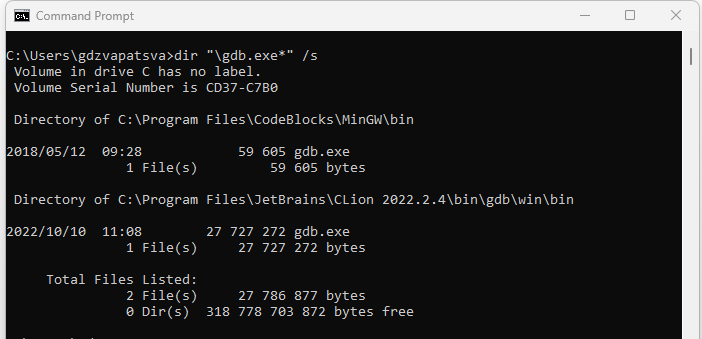


Figure 4. 15:Locating gdb application

### 4.5.9 Add the MINGW bin directory to the operating system path environment variable

Remember, in our case, we installed CodeBlocks with MingW. So, our MingW folder is located inside the CodeBlocks folder. We need to set up the path so that the compiler is available in our terminal irrespective of path. Now, if we run the following commands: gcc –version or g++ --version, we get an error message as shown in Figure 4.12

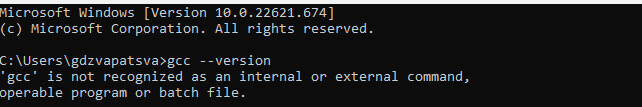


Figure 4. 16:Unrecognised command

To solve the error message in Figure 4.12, we have to set the path. So first let us navigate to CodeBlocks folder and look for MingW. Inside MingW, locate the path for bin and type cmd in the address bar. The terminal will open with the current location of bin folder. Copy the path of the terminal.

Alternatively, when in the bin folder, right click any of the files and select properties option. Copy the path under location section. Now let us set the path.

Step 1: Open environment Variables by typing env from the search bar

Step 2: Click Environment Variables Tab

Step 3: Under System variables, highlight path and click on edit

Step 4: Click New and add the copied path for bin and click OK and close any window for system variables open. Also close any command prompt window open

Step 5: Now text if the compilers are now accessible by typing any of the following command:

g++ --version

gcc –version

The terminal will give output such as one in Figure 4.13.

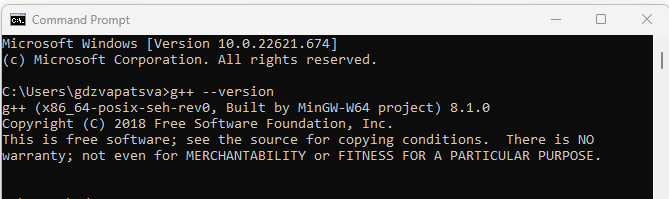


Figure 4. 17: Output of compiler on system path

Let us test our compiler

**EXAMPLE 4.6**

Open the terminal from the desktop.

Type the following command:

Notepad c\_test.cpp

You will be prompted to create a file called c\_test click yes.

Type the following c++ code

#include <iostream>

using namespace std;

int main ()

{

cout<<"1-2-3 testing";

return 0;

}

Type the following on the terminal:

g++ c\_test.cpp -o c\_test.exe

Press enter and type c\_test.exe. The output of the compiled c++ file will appear on the terminal. See sample Figure 4.14

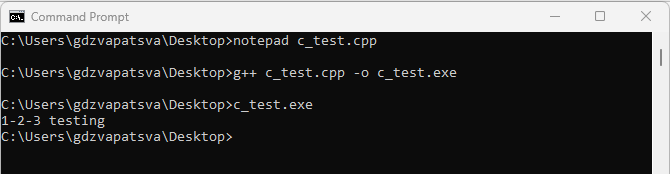


Figure 4. 18:Sample code execution and output

### 4.5.10 Compile source code wit the -g flag in IDE

To compile source code (.cpp) files using g flag is pretty straightforward as shown below.

First you will need to navigate to the folder where the project has been saved through the command prompt. For example, I navigate to the path shown since I have source code in that folder.

C:\Users\User1\Desktop\ADDS>g++ -g addition.cpp -o hello.exe

Our executable file has been generated. But what was the purpose of the g flag.

The 'g' flag instructs the compiler to provide more source code information in the executable than it otherwise would. This makes using a debugger like gdb much simpler because it will be able to use variable names that are present in the source code as references. gcc -g generates debug information to be used by GDB debugger.

# FORMATIVE ASSESSMENT 4.4 INDIVIDUAL TASK

4.4.1 Define the following terms:

* 1. Syntax error
  2. Runtime error
  3. Logic errors (6)

4.4.2 Give **FOUR** reasons why debugging is important. (3)

4.4.3 Which debugging technique allows the programmer to debug one or more applications on a remote machine. (1)

4.4.4 Define the term stepping as used applied in C++ debugging process. (2)

4.4.5 Write a program which accepts first\_name and surname and then prints out the fullname. Your code should look as the one in Figure 4.19.

Text

Description automatically generated

Figure 4. 19:Sample code

Question. Add a breakpoint in line six and run the debugging process and display the output of the console as shown in Figure 4.20. Print the screenshot as displayed.

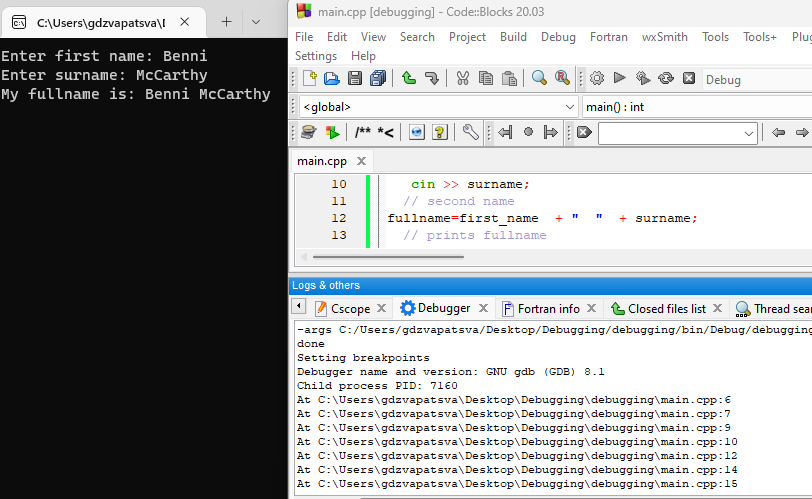


Figure 4. 20:Output

(10)

**Total: 22 marks**

## 4.6 Debugging C++ with GDBgui

**Introduction**

Developers commit a significant amount of time to debugging and maintaining existing codebases. Understanding various debugging methods is critical. Some developers still prefer more manual methods of debugging. There are also many snapshot tools that provide a detailed report of the issues and errors that occur after a specific section of code is executed. gdbgui is a browser-based frontend to gdb, the gnu debugger. You can add breakpoints, view stack traces, and more in C++ and other languages. gdbgui is another debugging tool built on top of gdb.

### 4.6.1 Explain the concept of a debugger frontend

gdbgui provides developers with a browser-based frontend through which they can add breakpoints, view stack traces, and change the context and parameter values while the debugger is running.

### 4.6.2 Install and configure gdbgui on Desktop PC

To install gdbgui, use pip command as follows:

pip install gdbgui==0.13.5.0

gdbgui can be problematic on Windows with incompatibility issues. The most common error you will encounter says:

RuntimeError: Windows is not supported at this time. Versions lower than 0.14.x. are Windows compatible.

The solution is to install gdbgui version less than 0.14.x

### 4.6.3 Start gdbgui application

To start gdbgui, type the following command in the terminal:

>gdbgui

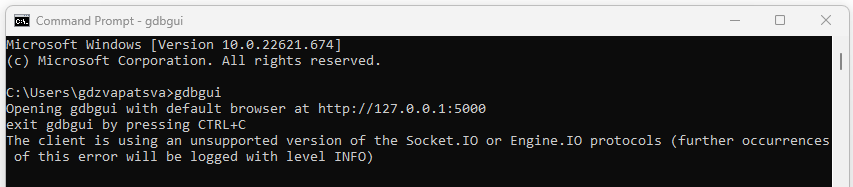


Figure 4. 21: Starting gdbgui

Once you type the command and press enter, the browser will open and

### 4.6.4 Load a compiled C/C++ binary (.exe)

Graphical user interface, text, application

Description automatically generated

Now you can debug your executable files using a browser version.

To load the binary you just need to copy the path of the executable file and click on load binary. The full source code will appear in the browser.

### 4.6.5 Run the loaded binary

To run the file, next to the address bar, you will find and an arrow which allows you to run the program. You will need to set breakpoint just the same way we did in Code::Blocks

### 4.6.6 Step through the running binary line by line

While working with gdbgui, it allows programmer to type commands in the browser terminal sections.

The most frequently used GDB commands are the following:

* r: start the program
* s (step): execute next program line (after stopping); step into any function calls in the line
* n (next): execute next program line (after stopping); step over any function calls in the line
* d: delete breakpoints

### 4.6.7 Investigate (watch) the values of local variables

Use a watch condition to keep track of changes in a variable's or an expression's current value that influences a storage location's address. With one key distinction, setting watch conditions differs from setting conditional breakpoints in that they cause the program to terminate as soon as a variable's value changes from its initial value. The storage address's content is computed at the time the watch condition is set and used by the debugger to watch a variable. A breakpoint is created, the program terminates, and the content at the storage location is altered from the value it had when the watch condition was set or when the last watch condition occurred.

# FORMATIVE ASSESSMENT 4.5 GROUP TASK

4.5.1 Install gdbgui and debug the following program.

Text

Description automatically generated

(20)

**eLink**

<https://www.youtube.com/embed/Cd0BOOdGjrw>

4.7 The Git Version Control System

**Introduction**

There is often confusion between the terms *source control*, *revision control* and *version control*. The figure below provides a quick overview of source-code management. In level 2, we explained the differences by use of the following diagram shown in Figure 4:22.

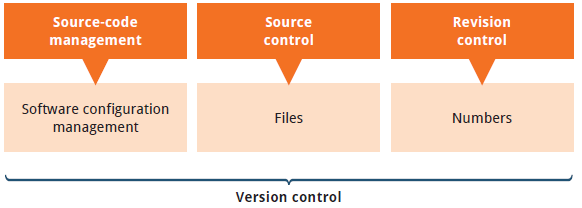
****

Figure 4. 22: Version control

Version control is a system that records changes to a file or set of files over time so that you can recall specific versions later. Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done in the code.  For the examples in this book, you will use software source code as the files being version controlled, though in reality you can do this with nearly any type of file on a computer.

**Types of Version Control Systems:**

* Local Version Control Systems
* Centralized Version Control Systems
* Distributed Version Control Systems

**Local Version Control Systems- It is one of the most straightforward types and includes a database that maintains revision control over all file modifications.**

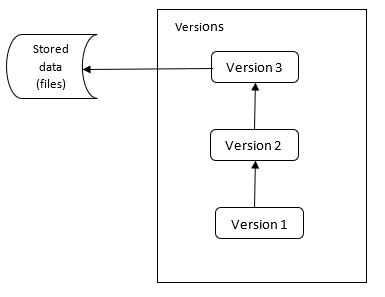


Figure 4. 23: Local Version Control Systems

**In LVCS, every patch set contains only the changes made to the file since its last version. To see what the file looked like at any given moment; it is necessary to add up all the relevant patches to the file in order until that given moment.**

**Centralized Version Control Systems- These only have a single repository globally, and each user must commit for their modifications to be reflected in the repository. With CVCS, this way, everyone usually knows what everyone else on the project is doing. Administrators have control over who can do what which might be a downfall of CVCS. Figure 4.24 shows how CVCS looks like.**

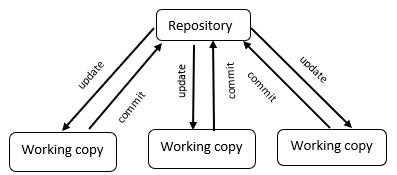


Figure 4. 24: Central Version Control System

**Distributed Version Control Systems (DVCS)-** Multiple repositories are present in distributed version control systems, and each user has a separate working copy and repository. You cannot share your modifications with others by simply committing them. This is because commit just makes those changes accessible in your local repository; to make them visible on the main repository, you must push them. Git is an example of DVCS. Linus Torvalds created Git in 2005 for the development of the Linux kernel. Git allows multiple developers to work together and supports non-linear development through its thousands of parallel branches.

Diagram

Description automatically generated

Figure 4. 25: Distributed Version Control System

### 4.7.1 Defining Git is and what it is used for

According to Git website, Git is a free and open-source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. Git is easier and simpler to learn. Other similar software for version control includes Subversion, CVS, Perforce, and ClearCase. At the time of writing this book, Git was the commonly used VCS.

Git allows several developers to collaborate on non-linear development by recording changes in the source code. Developers used to send their code to the central server without having backups. Any changes made to the source code were unknown to the other developers. There was no communication between any of the developers. However, with Git, every developer has an entire copy of the code on their local systems. Any changes made to the source code can be tracked by others. There is regular communication between the developers. For Git to flows to be completed- there must be a hosting platform for the remote repositories. Git uses GitHub.

GitHub is a web-based Git repository hosting service, which offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. Github is exclusively cloud based hosting service which enable programmers /users to manage Git repositories.

### 4.7.2 Three main goals of the Git VCS

Git has three main goals that it attempts to achieve, and these are speed, data integrity and distributed.

**Speed**

Git handles all projects with speed irrespective of sizes. The speed might only be controlled by the network. Git servers are very efficient.

**Data integrity**

Each item in Git is checksummed for identification purposes before being saved. This means that no file or directory can have its contents changed without Git being aware of it. Git is fundamental to its ethos and has this functionality built in from the ground up.

**Distributed**

One of the nicest features of Git is that it's distributed. This means that instead of doing a "checkout" of the current tip of the source code, you do a "clone" of the entire repository.

### 4.7.3 Relationship between local Git directory and directory located on server

Let us explain in simpler terms. When programmer(s) are working on a project, they save their file on the local machine. Once they initialise Git in that folder- the folder is known as the local repository. Any changes will need to be committed so that they are in the staging area. I hope this explanation very succinct. Here is a what you need to understand with Git. Git has three main states that your files can reside in: modified, staged, and committed:

* Modified- means that you have changed the file but have not committed it to your database yet.
* Staged- means that you have marked a modified file in its current version to go into your next commit snapshot.
* Committed -means that the data is safely stored in your local database.



Figure 4. 26: Git workflow

(Source:https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F)

### 4.7.4 Software license used for Git

Open-source software is frequently shared via the public repositories on GitHub. You must grant people the right to freely use, modify, and distribute the software for your repository to be considered fully open source. There is no pressure on you to select a license. However, without a license, copyright rules that are in effect by default apply, which means that you maintain ownership of your source code and that no one else is permitted to copy, distribute, or create derivative works of your work. Most authors save the text of their license in a file with the name LICENSE.txt. There is a variety of licenses to choose from.

### 4.7.5 Advantages and disadvantages of software license used by Git

Git’s open-source licensing has several advantages and disadvantages.

**Advantages of Git licensing**

1. **Flexibility and agility**- Technology agility is made possible by open-source licensing, which often provides several solutions to a given issue.
2. **Speed**- The ability to use community versions to get started, discover whether they can address your business challenge, and begin generating value immediately is a significant advantage of open-source licensing.
3. **Cost-effectiveness**- Open-source licensing is generally much more cost-effective.
4. **Attract better talent**- Most working developers are familiar with open source, and many of them think it will be the future of their field. Attracts large community.
5. **Collaboration culture**

**Disadvantages of Git licensing**

1. Some individuals may find it less user-friendly because we are indirectly copying the source code.
2. Support issues can also go wrong from time to time.
3. The open-source license itself is free, however there are costs related to its plugin modules and a host of other things.

### 4.7.6 Explain why Git was created

Git was created to resolve weaknesses of other version control systems for instance the centralised version control system. Git's branching features are among its greatest benefits. Git branches are inexpensive and simple to combine, in contrast to centralized version control solutions. This makes it easier to employ the feature branch workflow, which is well-liked by Git users.

### 4.7.7 Relationship between Git and hosting providers like GitHub

Most people tend to confuse Git and GitHub. The key difference between Git and GitHub is that Git is is a software and GitHub is a service. To make sure that the same copies are available with the remote repository, which is found on the GitHub server, the programmer will need to push the changes. You install Git locally on your machine while GitHub is hosted on the web.

### 4.7.8 Major open-source projects using Git

Since its inception, open software has fundamentally altered the digital ecosystem. On GitHub, developers are working on multiple projects at the same time. Here is a collection of open-source projects that make use of Git.

* Flutter
* freeCodeCamp
* First Contributors
* TensorFlow

### 4.7.9 Install and configure Git on desktop

The first important thing to do before installing git will be to check if it’s not installed already on your machine. Git uses commands. The command to run when checking for installed versions is git version or git - -version. These two commands produce the same result. Open command prompt and run the command.

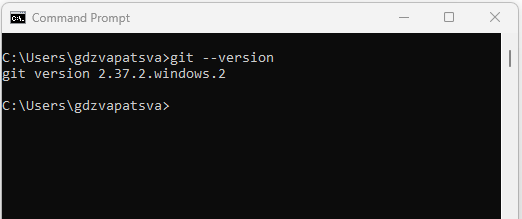


Figure 4. 27:Git checking

If Git is already installed, you will be able to see the version installed as in Figure 4.27. As you can see, my machine had Git installed and I am going to uninstall it and reinstall again for the sake of this lesson. If Git is not installed, you will get the following error message on your command prompt:

*“'git' is not recognized as an internal or external command, operable program or batch file.”*

Now you can go ahead and follow installation steps.

There are also a few ways to install Git on Windows. The most official build is available for download on the Git website.

Steps to Install Git on Windows Desktop

1. Navigate to the following link <https://git-scm.com/download/win> and download Git
2. Open the executable and run the file. You will see license agreement screen as shown in Figure- 4.28. Click Next.

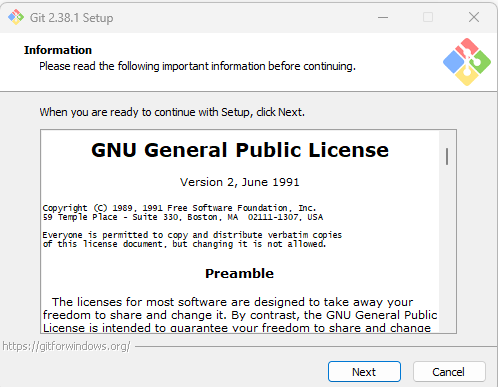


Figure 4. 28: Git License Agreement

1. Continue to click Next. As for setting the path: leave the default one.

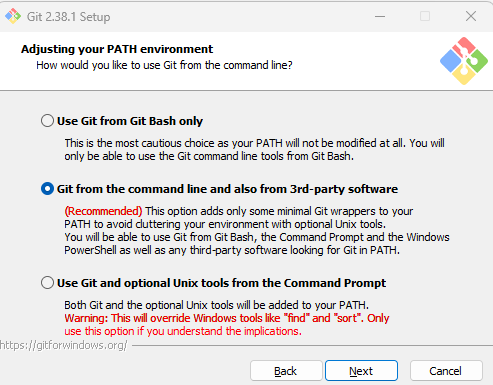


Figure 4. 29:Selecting Git Path environment

1. Continue to click Next (Leave the default options selected). Once complete run the git – version/git version. You will see the new version installed.

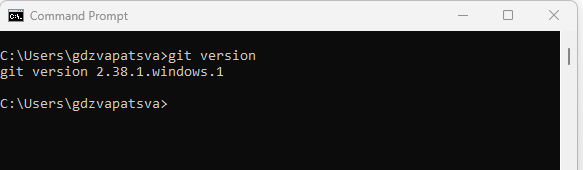


Figure 4. 30:Git Version

So, you can see that, earlier, I had git version 2.37.2 and now I have installed version 2.38.1.

Please take note, the version will change but at the time of writing this book, the version I installed was 2.38.1

**Git configuration**

1. Open command line and type the following command to configure your Git username, where <your name> will be your GitHub username.

git config –global user.name “your name”

1. Add the user email by typing:

git config –global user.email “your\_email-address”

1. To check if the configuration is successful, type the following command:

git config --list

You will see the username and email that you used to configure your Git.

1. The last important thing is to go to GitHub and sign up for an account. Remember the email address you used must be same so that your Git on your machine is linked to your GitHub account.

When you are done setting up your git, execute the following command to see that your username and email has been successfully configured:

git config --list

In my case my output shows as follows.

Text

Description automatically generated

Figure 4. 31: Git config output

The last 2 lines at the bottom of Figure 4.31 shows my git name and my git email which I used. In your case, your output must reflect the username and password which you used.

**Pushing work into GitHub**

We need to understand some commands used with git. Here are some of the common commands

**Creating a Git repository**

git init - initialize an existing directory as a Git repository

**git add**

Adds files in the to the staging area for Git. Before a file is available to commit to a repository, the file needs to be added to the Git index (staging area). You can add specific file buy using:

git add <filenames>

Or you can add all the files in the directory by using:

git add .

**git commit**

Record the changes made to the files to a local repository. For easy reference, each commit has a unique ID and the identifier -m is used to attach a message e.g.

git commit -m “second push”

**git push**

Is used to push the committed changes which are now on the stagging area to the remote repository (GitHub) Once you type this command, an exact copy of the repository will be available in GitHub

git push -u origin main

**git remote**

To connect a local repository with a remote repository. A remote repository can have a name set to avoid having to remember the URL of the repository.

# Add remote repository

git remote <command> <remote\_name> <remote\_URL>

example:

git remote add origin https://github.com/gpdzvapatsva/testing\_repo.git

# List named remote repositories

git remote -v

**git branch**

To determine what branch the local repository is on, add a new branch, or delete a branch.

**EXAMPLE 4.7**

* Create a folder on the desktop and call it git\_test.
* Let us create a C++ project called testing and save it in the git\_test folder
* By default, the main.cpp file shows default code for hello world.
* Close the project and open the git\_test folder. Type cmd in the address bar to open the git\_test in command prompt. There are various ways of navigating. Do the one which works well for you.
* Now initialise your folder to a git repository by typing: git init
* Next add files to git repository by typing : git add .
* Commit changes by typing git commit -m “initial push”

Figure 4.32 shows all the above commands in succession

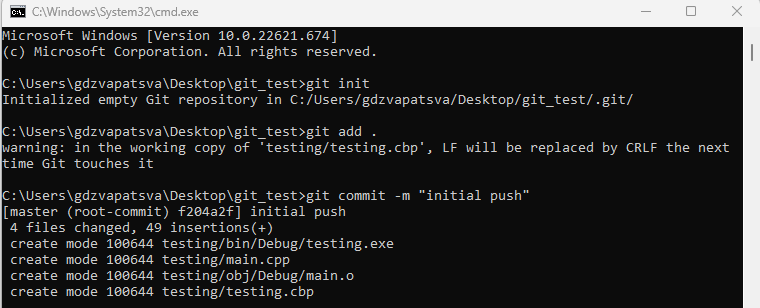


Figure 4. 32: Initialising a repository

* Now open GitHub using the credentials you used to create an account.
* On the left panel you will see a green button inscribed New. Click it and give a name to your remote repository. Let use c++\_hello\_world and hit create repository at the bottom.
* Now you will be presented with several option of pushing your work. For the purposes of this practice- select the option which says: “…or push an existing repository from the command line”. Copy all the three lines together and paste them on the command prompt as shown in Figure 4:33 and hit enter.

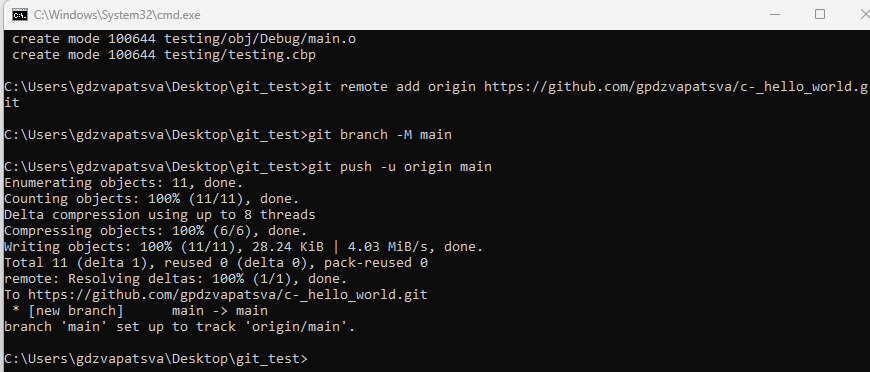


Figure 4. 33: Pushing work into a GitHub

* You will notice that a URL has been created for your repository. You can now share the repository with whoever wants the project.

Congratulations. You have pushed your first project.

### 4.7.10 Install and configure Git on Raspberry Pi

Raspberry Pi 4 model comes with Git preinstalled. Simply run the same commands as on Windows machine as shown in Figure 4.34.

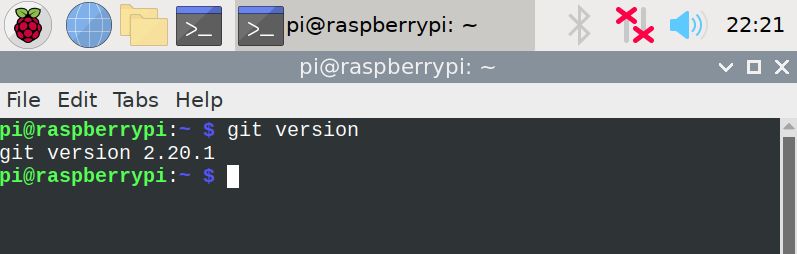


Figure 4. 34:Git of Raspberry Pi 4

For most people, the easiest and the recommended way to install Git is to install it using the [apt](https://linuxize.com/post/how-to-use-apt-command/) package management tool.

Installing Git with Apt

The Git package is included in the Raspbian default repositories. Here are the steps to install git on earlier versions.

**Step 1:** Open the terminal of Raspberry Pi OS, then write the following command:

$ sudo apt update

$ sudo apt install git

**Step 2:**Then write ‘Y’ to proceed further. And wait for some time to install

**Step 3**: Check if Git has been successfully installed by running the following command:

git –version

**Step 4**: Now configure Git using the following command:

git config –global user.name “your name”

git config –global user.email “your email”

* + 1. **List and explain common terminology associated with distributed VCS’s including:**

**Branch**

Earlier in section 4.6.4, we introduced branching. Just a reminder, branching means you diverge from the main line of development and continue to do work without messing with that main line. Developers can continue to work without affecting the main branch and if satisfied then merge to the main branch.

How to create a branch

We are going to use the git\_test repository we created for the hello\_world program. Syntax for creating a branch:

Create a new branch

>git branch <branch\_name>

Let us create a branch called first\_branch

After creating the branch we can check how many branches are on our repository by typing:

>git branch -a

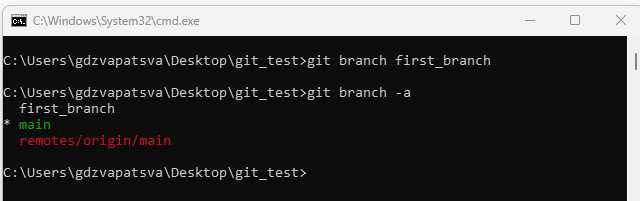


Figure 4. 35:Creating and checking branches

As you can see in Figure 4.35, we created a branch called first\_branch. There was the main branch already and after checking the number of branches we see two branches as output.

**Checkout**

git checkout

To start working in a different branch, use git checkout to switch branches.

> git checkout <branch\_name>

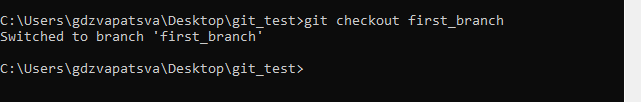


Figure 4. 36:Git Checkout

You can also move to a branch and create at the same time by typing the following command:

git checkout -b <branch\_name>

See Figure 4.37

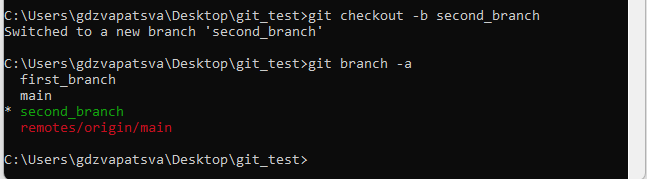


Figure 4. 37:Creating and navigating to branch

To delete a branch, you need to type:

git branch -d <branch\_name>

**NOTE:** To delete a branch, you must navigate to a branch in a higher level. For instance, to delete second\_branch, you must switch to the main branch first.

**Clone**

git clone

To create a local working copy of an existing remote repository, use git clone to copy and download the repository to a computer. Cloning is the equivalent of git init when working with a remote repository. Git will create a directory locally with all files and repository history.

Syntax

git clone <remote\_URL>

EXAMPLE

Lets clone a Python repository I created. The repository name is called <https://github.com/gpdzvapatsva/areaofcircle>

Make sure you navigate to the directory where you want the code cloned into. In our case, let us navigate to the desktop.

C:\Users\gdzvapatsva\Desktop>git clone <https://github.com/gpdzvapatsva/areaofcircle>

Output

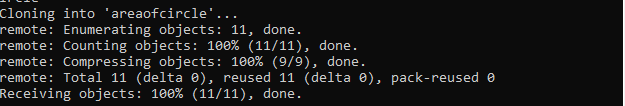


Figure 4. 38:Git clone

Compare your output with that on Figure 4.38 and check if you now have that Python project on to your desktop.

**Stage**

Staging is a step before the commit process in git. That is, a commit in git is performed in two steps: staging and actual commit. staging helps you keep extra local files hanging around.

git-stage - Add file contents to the staging area

**Commit**

git commit

As mentioned earlier, git commit is used to record the changes made to the files to a local repository. For easy reference, each commit has a unique ID. It’s best practice to include a message with each commit explaining the changes made in a commit. Adding a commit message helps to find a particular change or understanding the changes.

Syntax

git commit -m <comment>

git commit is used after adding changes and before pushing the changes to the remote repository. You have seen how we committed the git\_test folder. If we make any changes to the file, we will need to run git add and then git commit.

**Conflict**

When two different branches make updates to the same line in a file, or when a file is destroyed in one branch but changed in the other, a conflict results. For instance, conflicts generally arise when two people have changed the same lines in a file, or if one developer deleted a file while another developer was modifying it.The likelihood of conflicts occurring increases when working in a team environment Conflicts over merges can be resolved using a variety of tools.

**Head**

In the current checkout branch, the HEAD identifies the most recent commit. Like a pointer to any reference, it. The "current branch" is known as the HEAD. The HEAD is moved to the new branch when you switch branches with "checkout." The **git show head** is used to check the status of the Head. This command will show the location of the Head. Figure 4.39 illustrates the output of git show head.

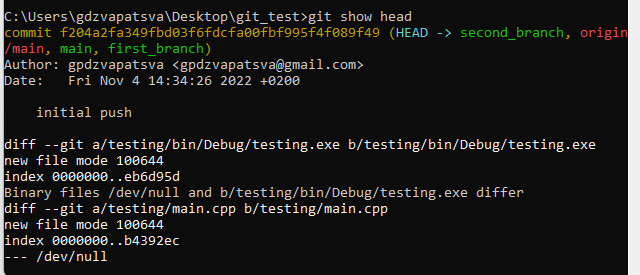
****

Figure 4. 39: Sample output of git show head

**Repository**

Each file in your project, together with its revision history, is stored in a repository. Within the repository, you can coordinate and oversee project activities. To see also if the folder is a git repository, open the folder, you will find another subfolder with extension .git inside.

You typically obtain a Git repository in one of two ways:

* You can take a local directory that is currently not under version control, and turn it into a Git repository, or
* You can clone an existing Git repository from elsewhere.

Git has four main states in which files can be saved: working directory, staging area, local repository and central repository.

* Working directory: where projects are created (code written) and changes are made
* Staging area: where code can be reviews before making a commitment
* Local repository: where you may commit changes to the project before pushing them to the central GitHub repository
* Central repository: where the primary project is stored (a copy is kept locally by each team member).

**Initialise**

A new Git repository is created with the git init command. It can be used to start a fresh, empty repository or convert an existing, unversioned project to a Git repository. This is typically the first command you'll perform in a new project because most additional Git commands are not accessible outside of an initialized repository.

# FORMATIVE ASSESSMENT 4.6 INDIVIDUAL TASK

4.6.1 Define the term version control. (2)

4.6.2 What is the difference between Git and GitHub? (4)

4.6.3 List FIVE benefits come with using GIT? (5)

4.6.4 Write the command used to initialise a Git repository. (2)

4.6.5 What does the command git -v do when typed on the git bash terminal. (2)

4.6.6 Practical

* Create a folder on the desktop and name it “git\_revision”.
* Add a text file called file.txt
* Add the following line of text through the terminal into a file called file.txt

“Working with git and github is saves time”

* Add the directory to git (including the file.txt)
* Commit the changes and add a comment which reads “initial push”
* Create a remote repository and give it the same name as the local repository and push the local folder to remote repository.
* What is the name of the remote repository.

(20)

**Total :35 marks**

# SUMMATIVE ACTIVITY 4.7 INDIVIDUAL TASK

4.7.1 Differentiate centralised version control systems from distributed version control systems. (4)

4.7.2 Define the term source code as applied in computer programming. (2)

4.7.3 List FOUR examples of IDE’s used for C++ programming. (4)

4.7.4 Explain how to display out from a C++ program. (2)

4.7.5 What are the THREE advantages of using GIT? (3)

4.7.6 Execute a Git command to create a copy of the following repository.

https://github.com/gpdzvapatsva/git\_revision.git

Write the command you used. (2)

4.7.7 What does the command git config --list do? (2)

4.7.8 What are the major differences between GCC and G++. (4)

4.7.9 Execute a git command to change the current user registered on your machine to “cplusguru”.

What command did you use to do this. (2)

4.7.10 List THREE common debugging techniques. (3)

**Total :28 Marks**

**References**

<https://education.github.com/git-cheat-sheet-education.pdf>

<https://git-scm.com/>

<https://byjus.com/gate/difference-between-gcc-and-g-plus-plus/>