Programming Fundamentals (CT-175) Lab 01

DevC++ Integrated Development Environment – Introduction, Features, and Compilation Process

Objectives

The objective of this lab is to familiarize students with the features of Integrated Development Environment (IDE) for C language and basic structure of C program. By the end of this lab, students will be able to create, compile, and execute basic input-output programs written in C language.

Tools Required

DevC++ IDE

Course Coordinator –
Course Instructor –
Lab Instructor –
Prepared By Department of Computer Science and Information Technology
NED University of Engineering and Technology

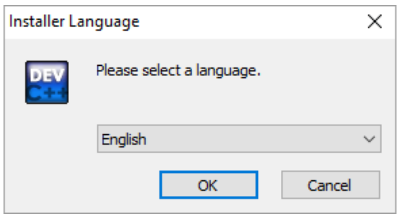
IDE – Integrated Development Environment

IDE stands for Integrated Development Environment. It is a software application that provides facilities for developing software. It consists of tools such as source code editor, automation tools, and debugger.

For C language programming we will be using DevC++. DevC++ is a full-featured IDE for the C/C++ programming language. DevC++ is free software and is distributed under the GNU General Public License. Thus we can distribute or modify the IDE freely. It offers to the programmer a simple and unified tool to edit, compile, link, and debug programs.

Installation and Configuration of DevC++ IDE

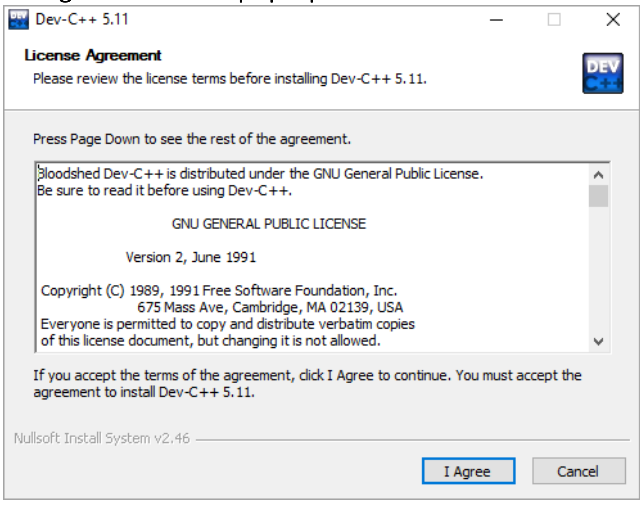
1. Download the required installer for dev-C++ IDE from https://www.bloodshed.net/
2. In order to start the installation process, double click on the downloaded file.
3. Select the language of our choice as shown in the below screenshot.


Installer Language

Please select a language.

English

4. Agree to the license agreement that pop-ups next.


Dev-C++ 5.11

License Agreement

Please review the license terms before installing Dev-C++ 5.11.

Press Page Down to see the rest of the agreement.

Bloodshed Dev-C++ is distributed under the GNU General Public License.
Be sure to read it before using Dev-C++.

GNU GENERAL PUBLIC LICENSE

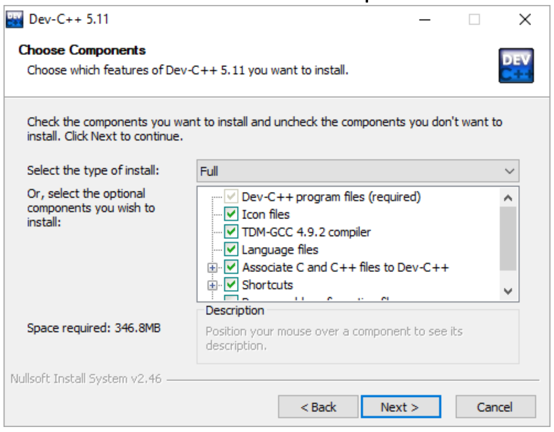
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
675 Mass Ave, Cambridge, MA 02139, USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

If you accept the terms of the agreement, click I Agree to continue. You must accept the agreement to install Dev-C++ 5.11.

Nullsoft Install System v2.46

5. Select the components that we need to install as a part of the dev-C++ installation.


Dev-C++ 5.11

Choose Components

Choose which features of Dev-C++ 5.11 you want to install.

Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select the type of install: Full

Or, select the optional components you wish to install:

- ✓ Dev-C++ program files (required)
- ✓ Icon files
- ✓ TDM-GCC 4.9.2 compiler
- ✓ Language files
- ✓ Associate C and C++ files to Dev-C++
- ✓ Shortcuts

Space required: 346.8MB

Description

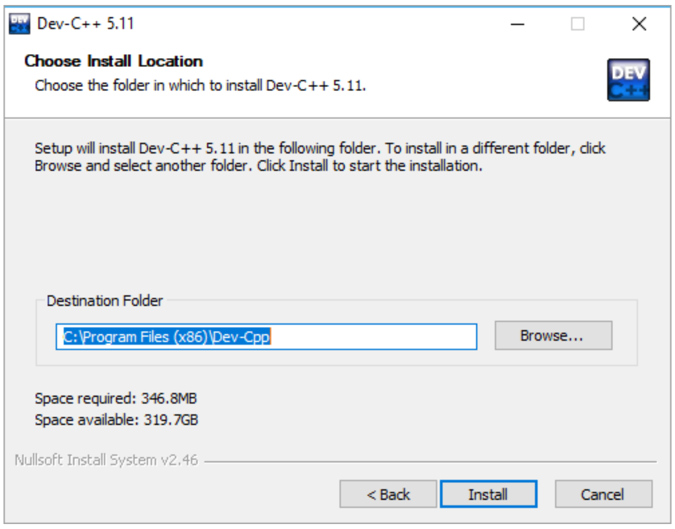
Position your mouse over a component to see its description.

Nullsoft Install System v2.46

As shown in the above screenshot, we are provided with a list of components available for installation and a checkbox against each component. We can check/uncheck each box to indicate which components to install. Click next once the components are selected.

6. Now the installer prompts the user for the destination folder where the dev-C++ files/libraries etc. are to be copied. Click on install if you do not want to change the default path to

destination folder. Otherwise, first define the path to destination folder and then click on install.


Dev-C++ 5.11

Choose Install Location

Choose the folder in which to install Dev-C++ 5.11.

Setup will install Dev-C++ 5.11 in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

Destination Folder

C:\Program Files (x86)\Dev-Cpp

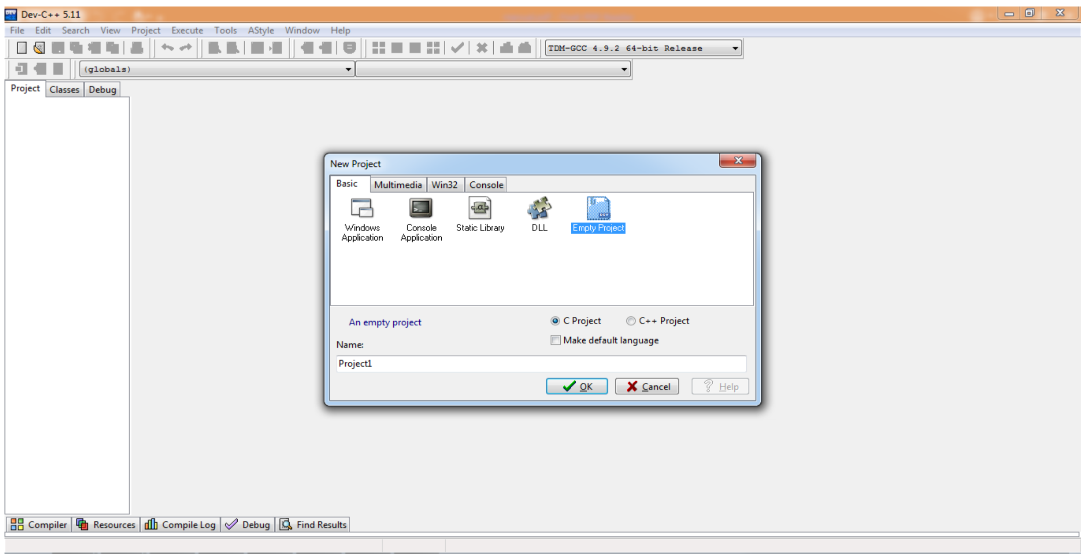
Space required: 346.8MB
Space available: 319.7GB

Nullsoft Install System v2.46

You are done with the installation of DevC++ IDE. Now let us use it to create the very first project. Double click on the DevC++ icon.

CREATE A NEW PROJECT

To create a new project or source file in dev-C++ go to menu File -> New->Project. Here, we can specify the project name. Make sure to select the “Empty Project” and also to check the “C Project” button.


Dev-C++ 5.11

TDM-GCC 4.9.2 64-bit Release

(globals)

New Project

Basic | Multimedia | Win32 | Console

Windows Application, Console Application, Static Library, DLL, Empty Project

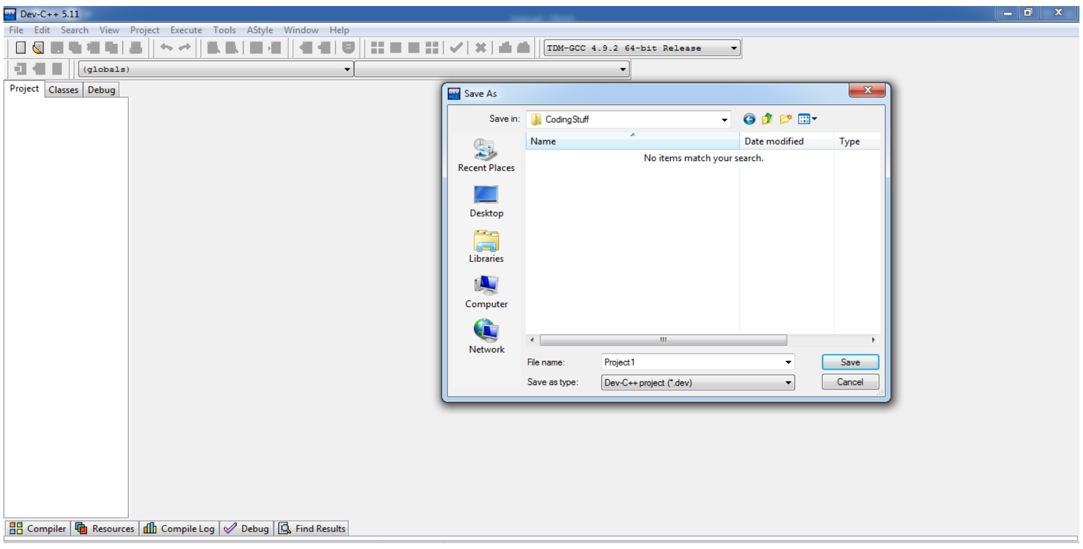
An empty project

[✓ C Project] C++ Project

Make default language

Name: Project1

Once the entire information is provided, we can click ok and the IDE will ask for the path where the project is to be saved. When this is done, a workspace will open with the project explorer on the left-hand side that shows the project we just created.


Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
Save As
Recent Places
Desktop
Libraries
Computer
Network
Compiler Resources Compile Log Debug Find Results

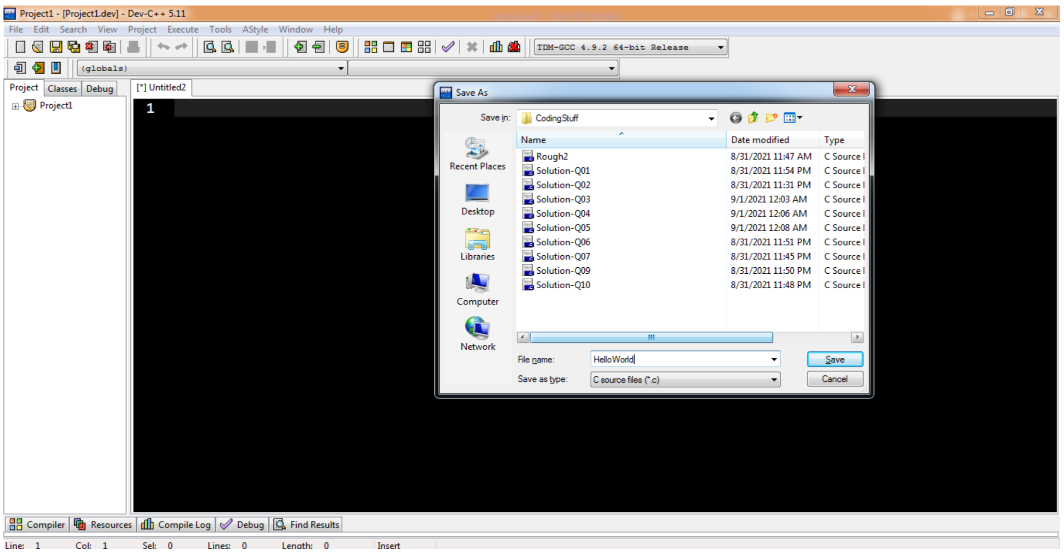
Save in: CodingStuff

Name	Date modified	Type
No items match your search.		

File name: Project1
Save as type: Dev-C++ project (*.dev)
Save
Cancel

ADD A SOURCE FILE TO A PROJECT

Add a new file by clicking **Project ->New File** or Right-click on **Project Name** in the project explorer and click **New File.**


Project1 - [Project1.dev] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
Project1
[*] Untitled2
Save As
Recent Places
Desktop
Libraries
Computer
Network
Compiler Resources Compile Log Debug Find Results

Save in: CodingStuff

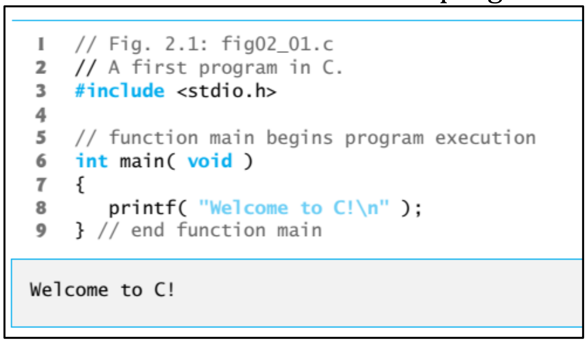
Name	Date modified	Type
Rough2	8/31/2021 11:47 AM	C Source I
Solution-Q01	8/31/2021 11:54 PM	C Source I
Solution-Q02	8/31/2021 11:31 PM	C Source I
Solution-Q03	9/1/2021 12:03 AM	C Source I
Solution-Q04	9/1/2021 12:06 AM	C Source I
Solution-Q05	9/1/2021 12:08 AM	C Source I
Solution-Q06	8/31/2021 11:51 PM	C Source I
Solution-Q07	8/31/2021 11:45 PM	C Source I
Solution-Q09	8/31/2021 11:50 PM	C Source I
Solution-Q10	8/31/2021 11:48 PM	C Source I

File name: HelloWorld
Save as type: C source files (*.c)
Save
Cancel

Line: 1 Col: 1 Sel: 0 Lines: 0 Length: 0 Insert

Structure of a Program in C

Following is the traditional structure of a "Hello World" program in C language.



```c
// Fig. 2.1: fig02_01.c
// A first program in C.
#include <stdio.h>

// function main begins program execution
int main( void )
{
   printf( "Welcome to C!\n" );
} // end function main
```

```
Welcome to C!
```

Main( ) Function

All C programs are divided into units called functions. No matter how many functions there are in a C program, **main ( )** is the one to which the control is passed from the operating system when the program is run; it's the first function executed.

Delimiters

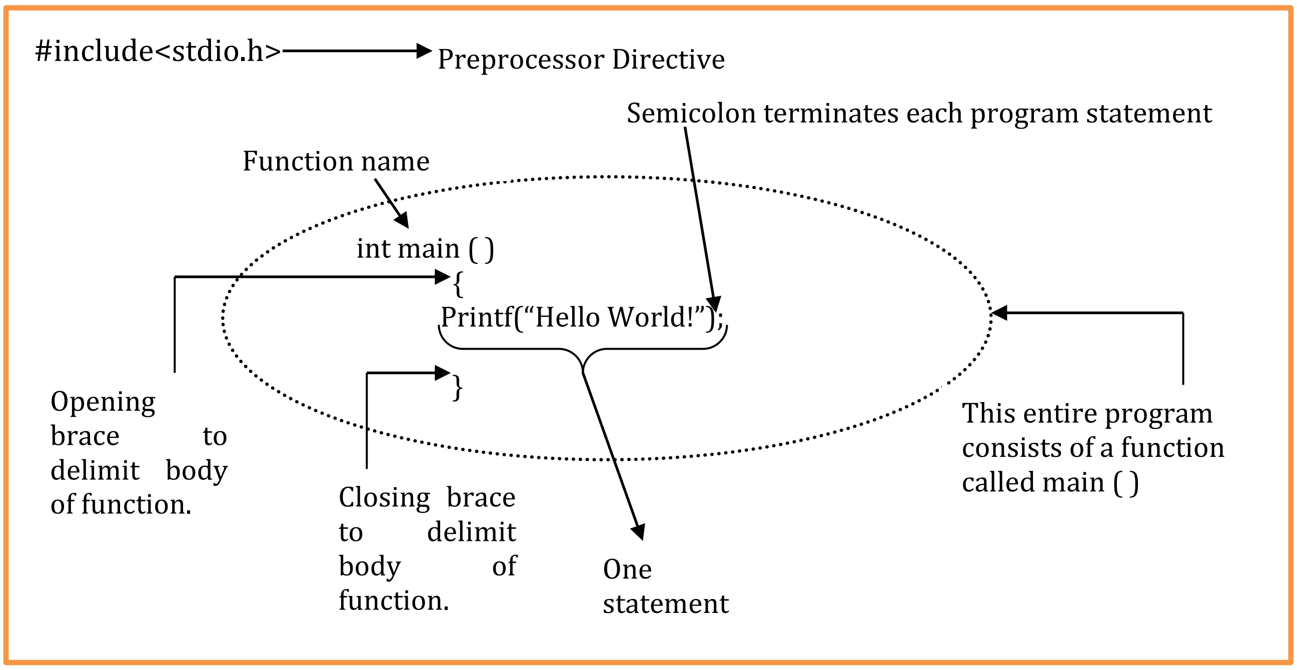
They signal the beginning and end of the body of the function. The opening brace ({) indicates a block of code that forms a distinct unit is about to begin. The closing brace (}) terminates the block of code. They are also used in loops and decision making statements.

Statement Terminator

A statement in a C program is terminated with a semicolon. Printf( ) is an example of a output statement. A semicolon does not separate statements; it terminates the line, not the carriage return you type afterwards. C does not pay attention to carriage return and white space characters.

Preprocessor library

A pound sign # indicates that the remainder of that line is an instruction (directive) to the preprocessor. #include <stdio.h> tells the preprocessor to add the code found in that file to the beginning of this program. This code handles all character stream inputs and outputs. Without this you cannot read from the keyboard or display anything to the monitor.


#include<stdio.h>
Preprocessor Directive
Semicolon terminates each program statement
Function name
int main ( )
{
Printf("Hello World!");
}
Opening brace to delimit body of function.
Closing brace to delimit body of function.
One statement
This entire program consists of a function called main ( )

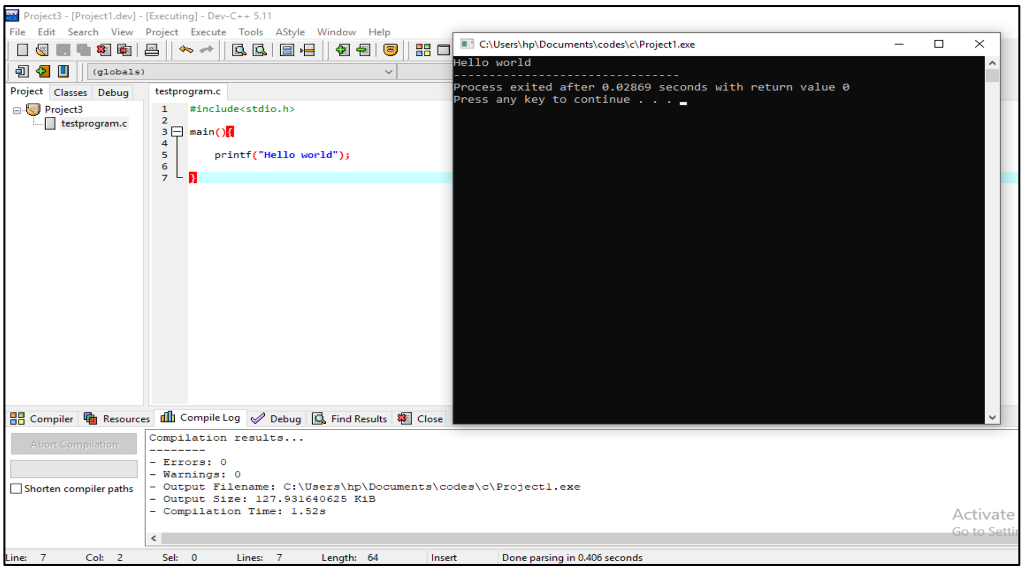
COMPILE/BUILD

When we have all the code ready for the project, we will now compile and build the project.

- ✓ To compile the project, click **Execute -> Compile** (or click F9).
- ✓ We can see the compilation status in the "**Compile Log**" tab in the workspace.
- ✓ If there are any errors whether syntax or linker errors, then they will appear in the compiler tab.
- ✓ Once the project is compiled successfully, we need to run it.

EXECUTE PROJECT

- ✓ Click on **Execute ->Run**.(or click F10)
- ✓ The console window that gives us the output will be shown in the below screenshot.


Project3 - [Project1.dev] - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
Project3
testprogram.c
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
C:\Users\hp\Documents\codes\c\Project1.exe

```
#include<stdio.h>

main(){

    printf("Hello world");

}
```

```
Hello world
--------------------------------
Process exited after 0.02869 seconds with return value 0
Press any key to continue . . .
```

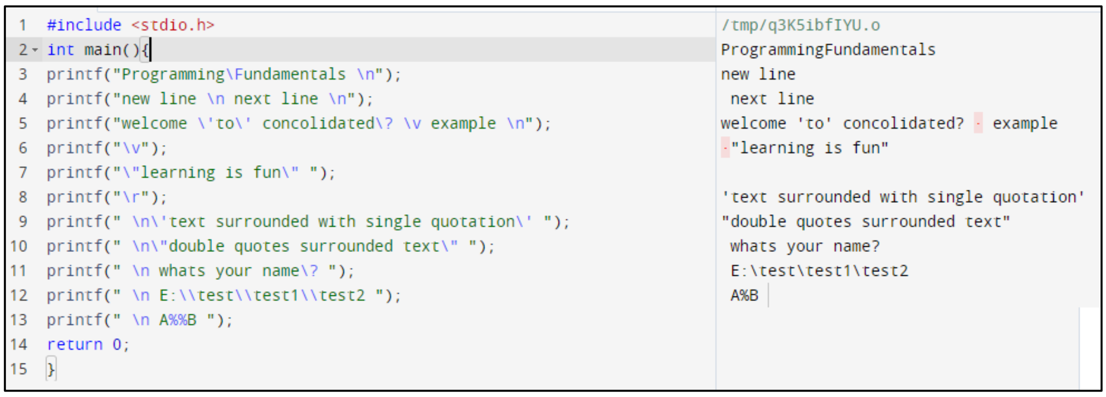
```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\hp\Documents\codes\c\Project1.exe
- Output Size: 127.931640625 KiB
- Compilation Time: 1.52s
```

```
Line: 7   Col: 2   Sel: 0   Lines: 7   Length: 64   Insert   Done parsing in 0.406 seconds
```

Escape Sequences

The backslash (\) is called an escape character. It indicates that printf is supposed to do something out of the ordinary. When encountering a backslash in a string, the compiler looks ahead at the next character and combines it with the backslash to form an escape sequence. Some common escape sequences are listed in the following Table.

Escape Sequence	Meaning
\n	New Line
\t	Horizontal Tab
\b	BackSpace
\r	Carriage Return
\a	Audible bell
\'	Printing single quotation
\"	printing double quotation
\?	Question Mark Sequence
\\	Back Slash
\f	Form Feed
\v	Vertical Tab
\0	Null Value
\nnn	Print octal value
\xhh	Print Hexadecimal value

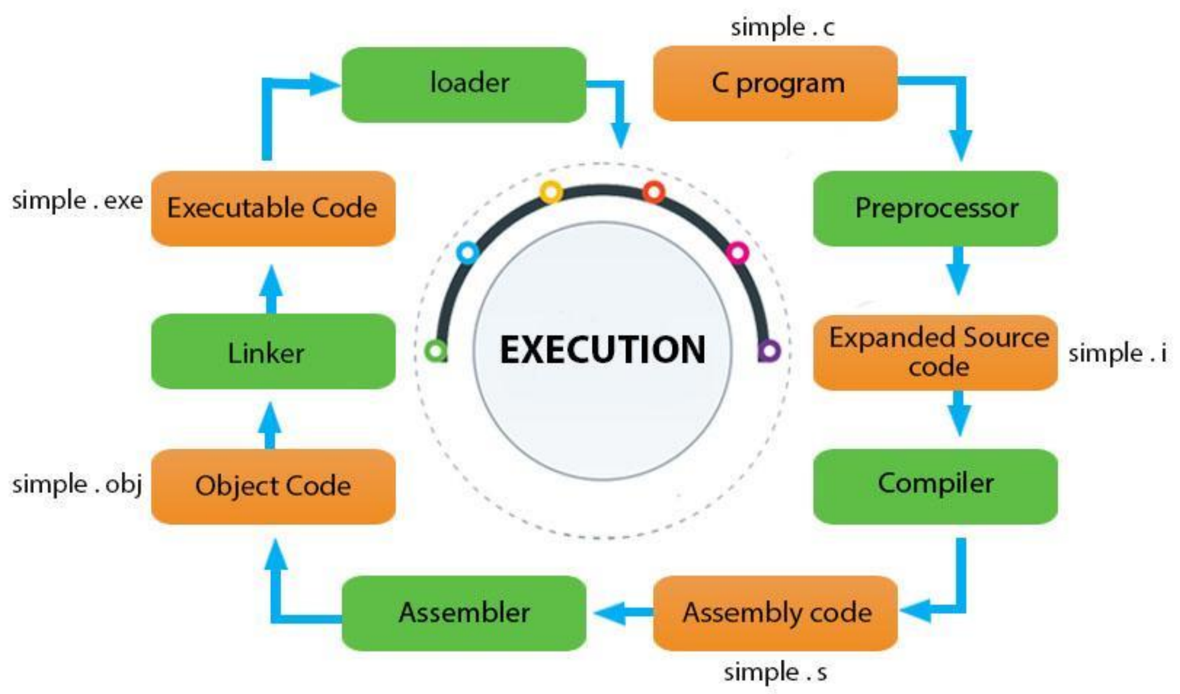


```c
#include <stdio.h>
int main(){
printf("Programming\Fundamentals \n");
printf("new line \n next line \n");
printf("welcome \'to\' concolidated\? \v example \n");
printf("\v");
printf("\"learning is fun\" ");
printf("\r");
printf(" \n\'text surrounded with single quotation\' ");
printf(" \n\"double quotes surrounded text\" ");
printf(" \n whats your name\? ");
printf(" \n E:\\test\\test1\\test2 ");
printf(" \n A%%B ");
return 0;
}
```

```
/tmp/q3K5ibfIYU.o
ProgrammingFundamentals
new line
 next line
welcome 'to' concolidated?   example
 "learning is fun"

'text surrounded with single quotation'
"double quotes surrounded text"
 whats your name?
 E:\test\test1\test2
 A%B
```

FLOW OF EXECUTION OF C PROGRAM

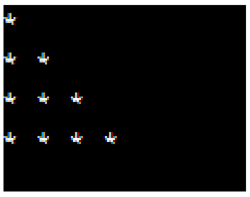
C is a high-level programming language developed in 1972 by Dennis Ritchie at AT&T Bell Laboratories. Programs in C typically go through six phases to be executed i.e., edit, preprocess, compile, link, load and execute.



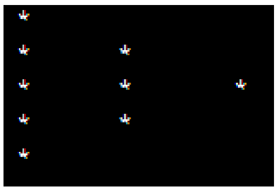
1. Write a c program and save it with the extension ".c".
2. The C program is sent to **preprocessor** first. The preprocessor is responsible to convert preprocessor directives (# sign denotes directives) into their respective values. The preprocessor generates an expanded source code.
3. Expanded source code is sent to **compiler** which compiles the code and converts it into assembly code. **Note -** Computer programs are written using high-level programming languages. The source code is converted into machine-understandable machine code. A compiler is used for this conversion. Compiler is a translator that converts the source code from high-level programming language to a lower level machine language in order to create an executable program.
4. The assembly code is sent to **assembler** which assembles the code and converts it into object code. Now a simple.obj file is generated.
5. The object code is sent to **linker** that links it to the library such as header files. Then it is converted into executable code. **Note -** Library functions are not a part of any C program but of the C software. Thus, the compiler doesn't know the operation of any function, whether it is printf or scanf. The definitions of these functions are stored in their respective library which the compiler should be able to link. This is what the Linker does. So, when we write #include, it includes stdio.h library which gives access to Standard Input and Output. The linker links the object files to the library functions and the program becomes a .exe file. A simple.exe file is generated which is in an executable format.
6. The executable code is sent to **loader** which loads it into memory and then it is executed. After execution, output is sent to console. **Note -** Whenever we give the command to execute a particular program, the loader comes into work. The loader will load the .exe file in RAM and inform the CPU with the starting point of the address where this program is loaded.

Exercise

1. Write a C Program to play beep five times.
2. What does the following code print?
   printf("*\n**\n***\n****\n*****\n");
3. Write a C program to print the following shapes using escape sequences.

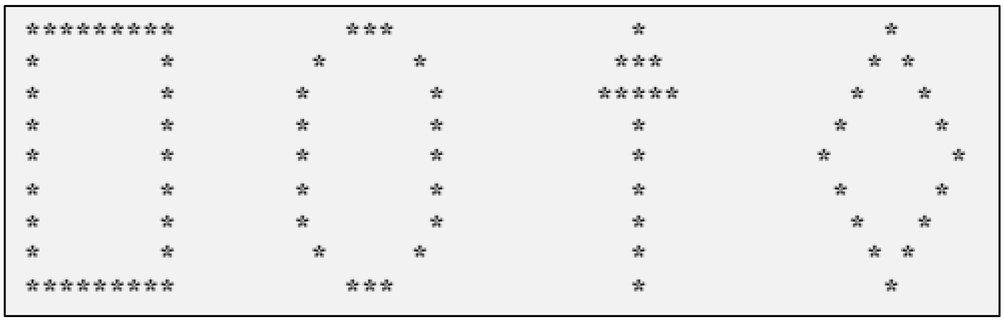


```
*

*  *

*  *  *

*  *  *  *
```



```
*
*      *
*      *      *
*      *
*
```

4. Write a program that prints the following shapes with asterisks.



```
*********          ***              *             *
*       *        *     *           ***           * *
*       *       *       *         *****         *   *
*       *       *       *           *          *     *
*       *       *       *           *         *       *
*       *       *       *           *          *     *
*       *       *       *           *           *   *
*       *        *     *            *            * *
*********          ***              *             *
```

Lab 01 Evaluation		
Student Name:		Student ID: Date:
Task No.	Marks	Remarks by teacher in accordance with the rubrics
1		
2		
3		
4		
5		