SNS COLLEGE OF TECHNOLOGY

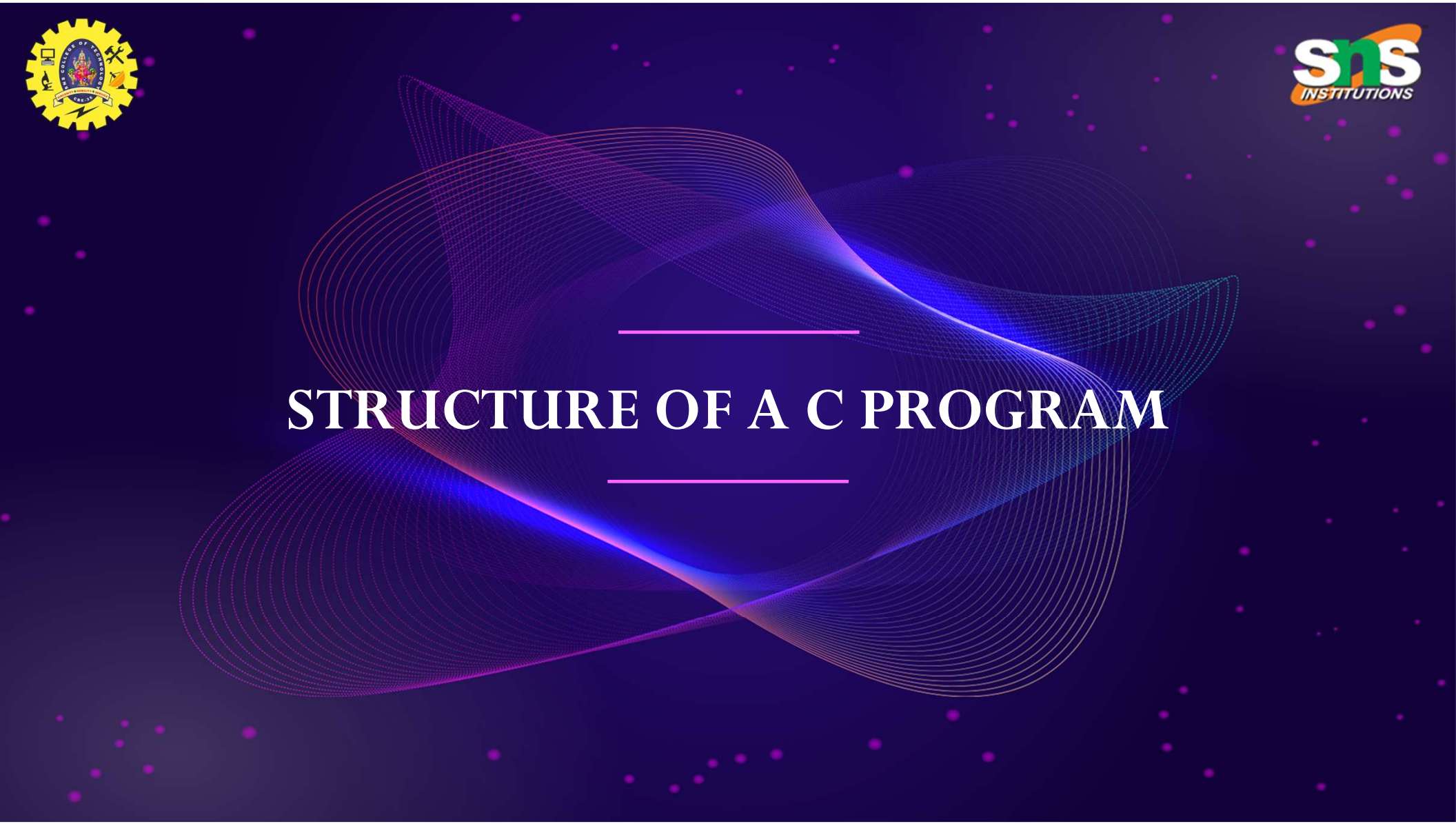
Coimbatore – 35
An Autonomous Institution

DEPARTMENT OF INFORMATION TECHNOLOGY

23CST101 PROBLEM SOLVING and C PROGRAMMING

I YEAR - I SEM

UNIT II – C PROGRAMMING BASICS


STRUCTURE OF A C PROGRAM

Structure of a C program

Basically the structure of a C program divides into 6 sections and they are,

01 Documentation

02 Preprocessor

03 Definition

04 Global declaration

05 Main function

06 User defined functions

Documentation Section

- ✓ It includes the statement specified at the beginning of a program, such as a program's **name, date, description,** and **title**, which is represented using commands
- ✓ Single line commands will be represented by //
- ✓ Multi – line commands will be represented as /* ……. */

```
//program1.c
```

```
/*
…
Overview of code
…
*/
```

Preprocessor Section

```
#include<stdio.h>
#include<conio.h>
```

Preprocessor Section (contd..)

- ✓ The preprocessor section contains all the header files used in a program. It informs the system to link the header files to the system libraries
- ✓ A header file in C/C++ contains:

  i) Function definitions ii) Data type definitions iii) Macros

- ✓ Header files offer these features by importing them into your program with the help of a preprocessor directive called **#include.**
- ✓ These preprocessor directives are responsible for instructing the C/C++ compiler that these files need to be processed before compilation.

Contd….

- ✓ Every C program should necessarily contain the header file **<stdio.h>** which stands for standard input and output used to take input with the help of **scanf() function** and display the output using **printf() function.**
- ✓ The source file contains #include which is responsible for directing the C/C++ compiler that this file needs to be processed before compilation and includes all the necessary data type and function definitions

Define Section

- The define section comprises of different constants declared using the define keyword

```
#define a = 4
```

Global declaration

- The global section comprises of all the global declarations in the program.
- Anything which is declared as global can be used throughout the entire program
- It should be declared before the main function

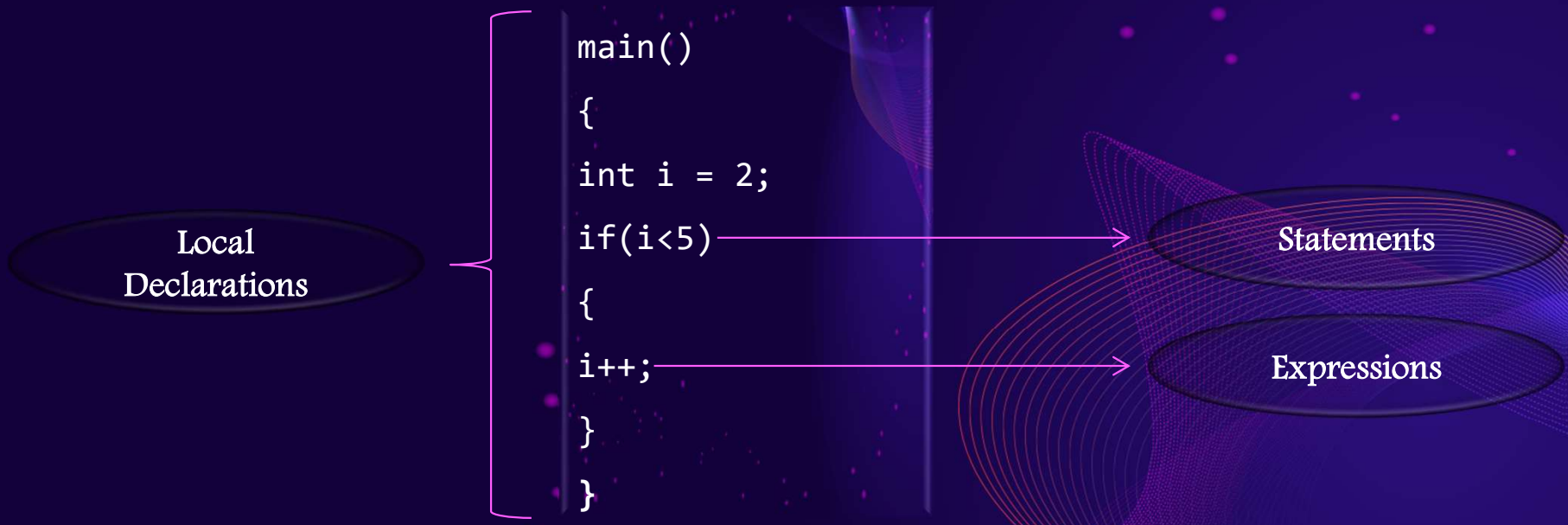
Main function

- main() is the first function to be executed by the computer.
- It is necessary for a code to include the main(). It is like any other function available in the C library.
- Parenthesis () are used for passing parameters (if any) to a function.

Declaration of main( )

main( ) function can be declared in three ways

- main( )
- void main( ) - specifies that program will not return any value
- int main( ) - specifies that program can return integer type data



User defined functions

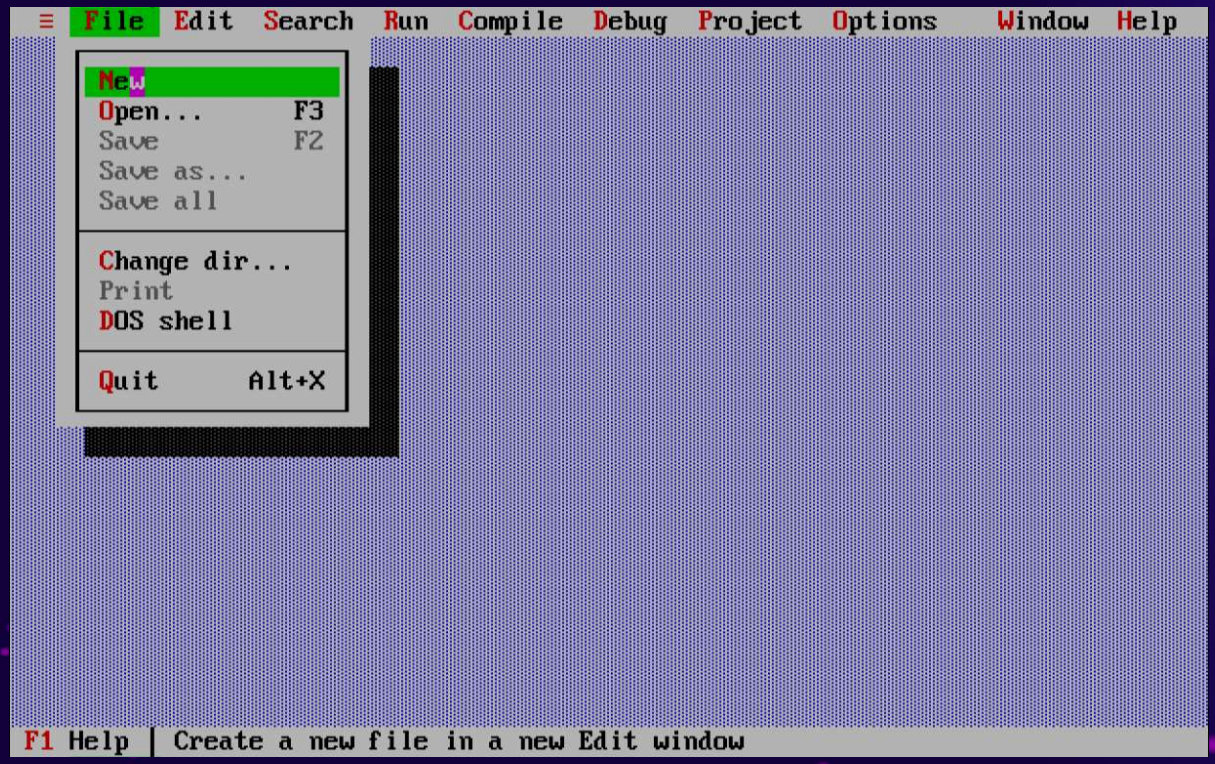
- The user defined functions specified the functions specified as per the requirements of the user.
- For example, color( ), sum( ), division( ), etc.

Basic Syntax

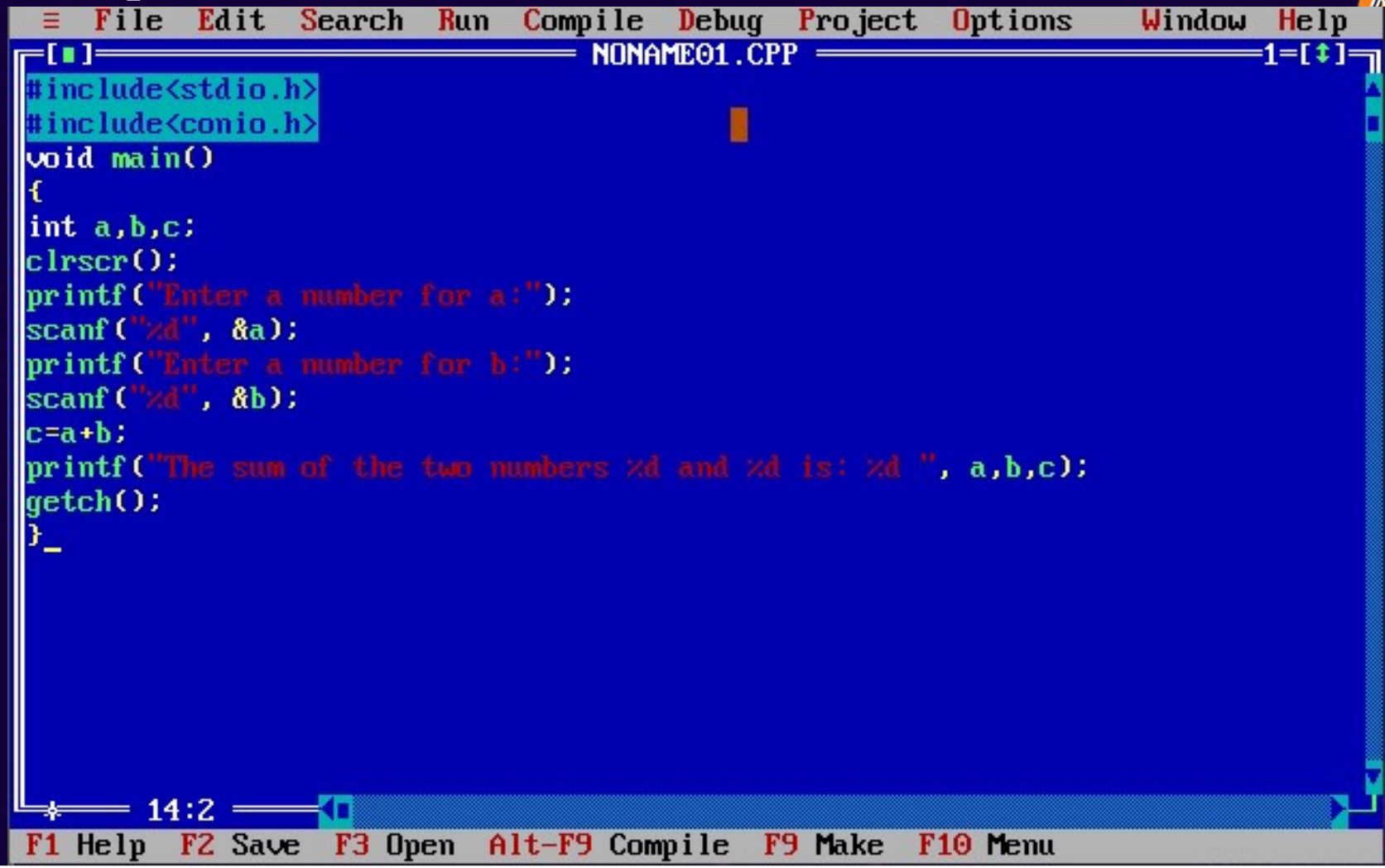
```
#include<stdio.h> //(Header Files)
main( )           //(Main function)
{
// Statements to be executed;
}
```

Editor and file creation

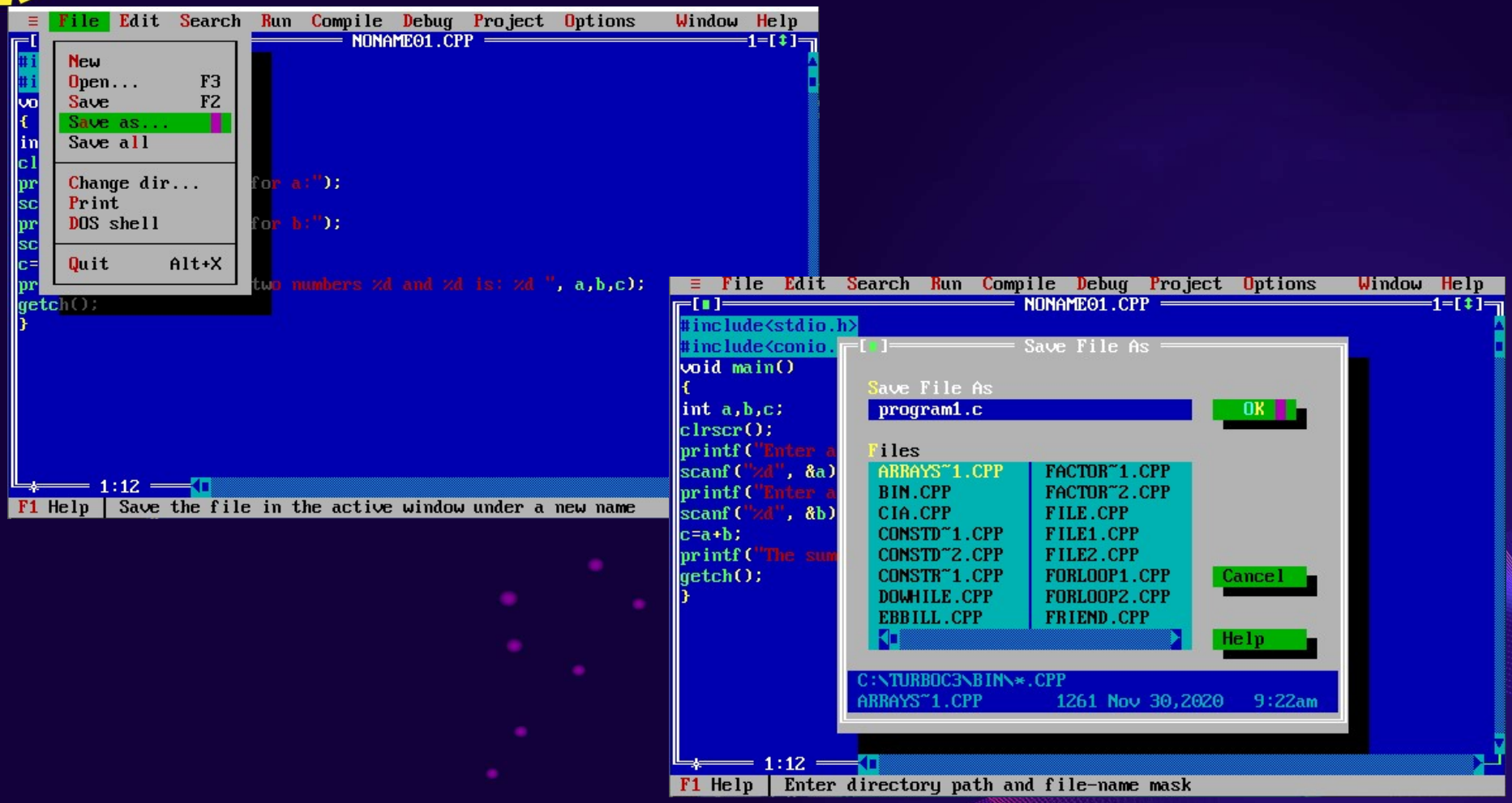
Editor used: Turbo C++



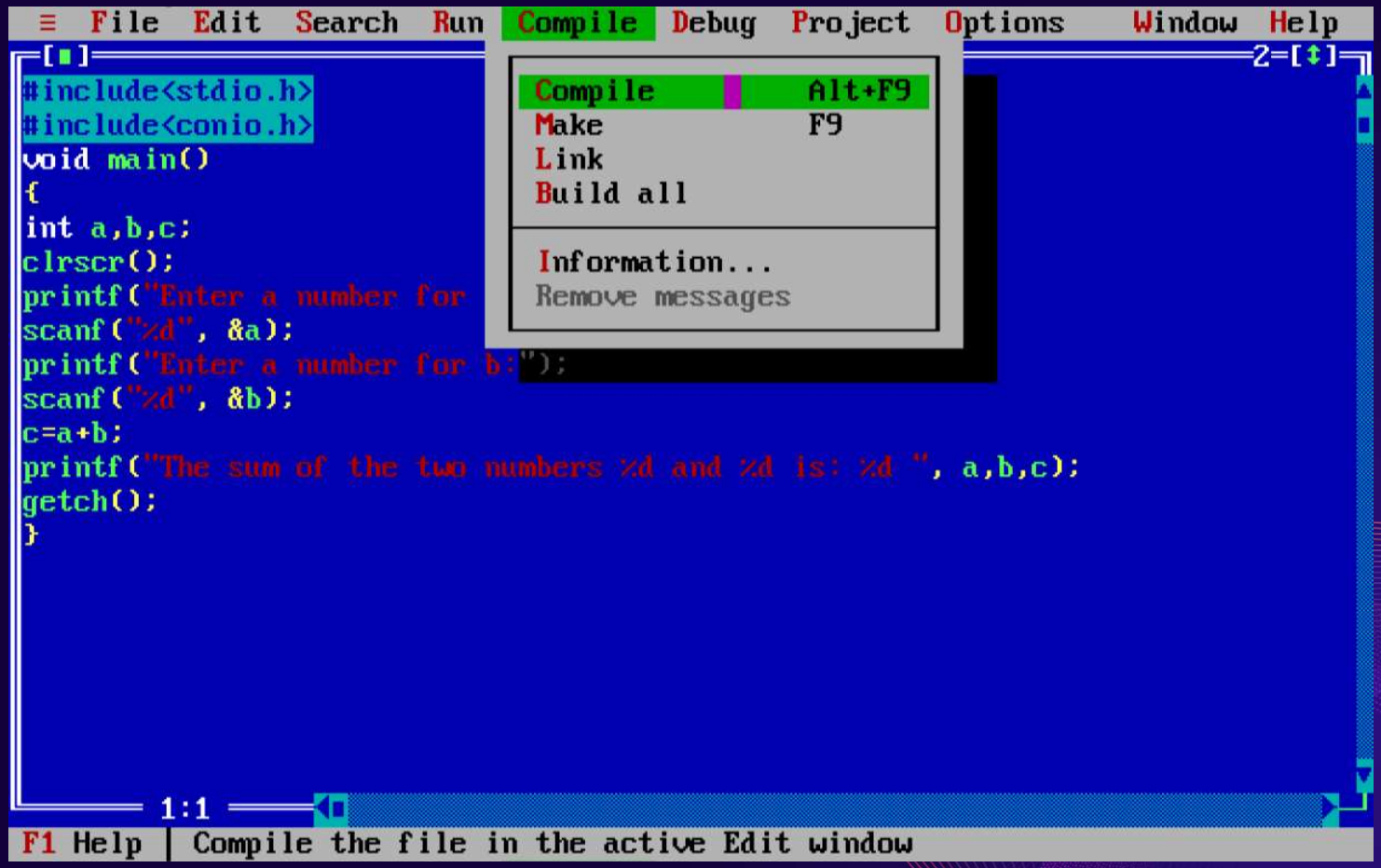
Example


File Edit Search Run Compile Debug Project Options Window Help
NONAME01.CPP
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
clrscr();
printf("Enter a number for a:");
scanf("%d", &a);
printf("Enter a number for b:");
scanf("%d", &b);
c=a+b;
printf("The sum of the two numbers %d and %d is: %d ", a,b,c);
getch();
}
14:2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

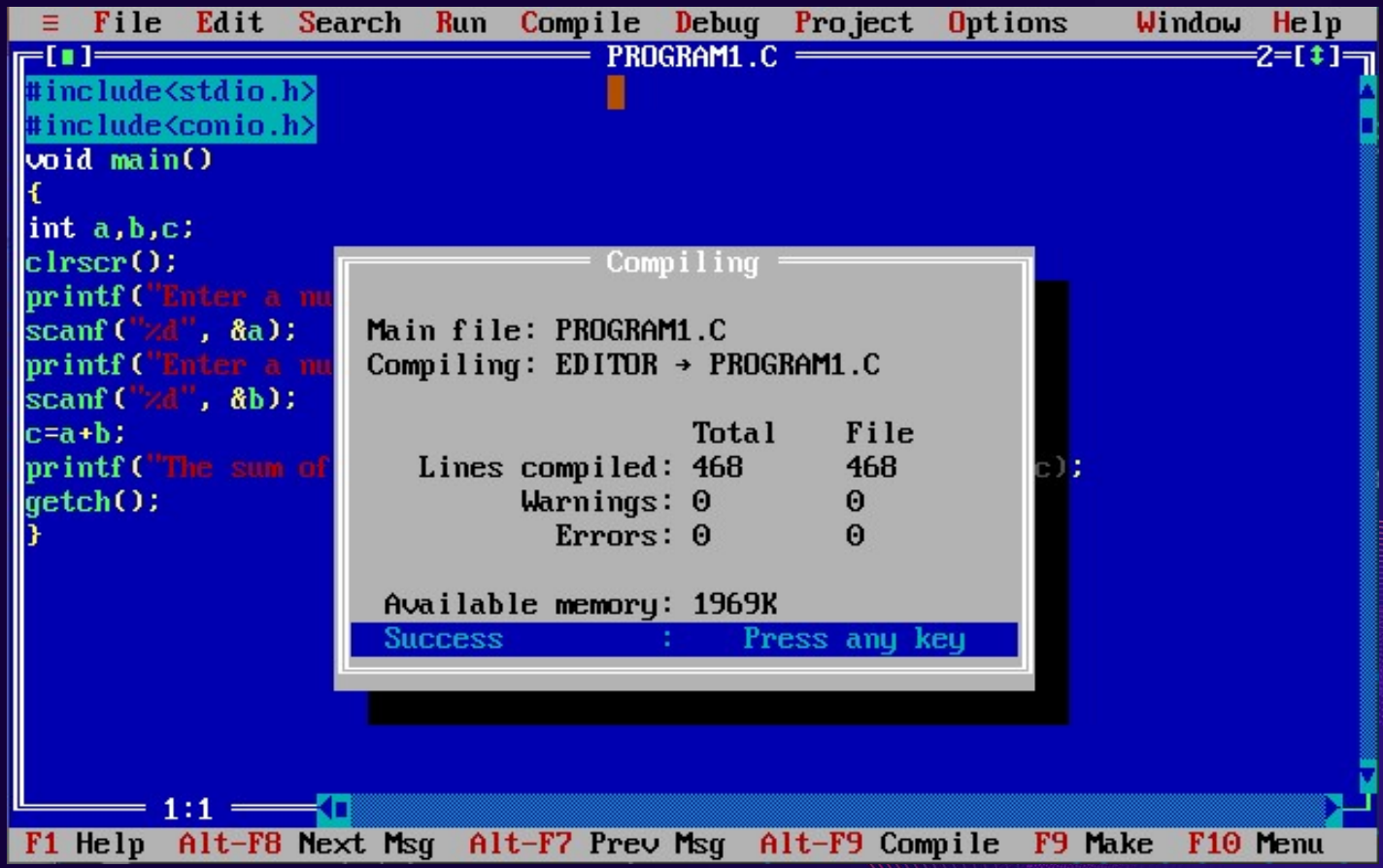
Example


File Edit Search Run Compile Debug Project Options Window Help
NONAME01.CPP
New
Open... F3
Save F2
Save as...
Save all
Change dir...
Print
DOS shell
Quit Alt+X
1:12
F1 Help | Save the file in the active window under a new name
Save File As
Save File As
program1.c
OK
Files
ARRAYS~1.CPP
BIN.CPP
CIA.CPP
CONSTD~1.CPP
CONSTD~2.CPP
CONSTR~1.CPP
DOWHILE.CPP
EBBILL.CPP
FACTOR~1.CPP
FACTOR~2.CPP
FILE.CPP
FILE1.CPP
FILE2.CPP
FORLOOP1.CPP
FORLOOP2.CPP
FRIEND.CPP
Cancel
Help
C:\TURBOC3\BIN*.CPP
ARRAYS~1.CPP 1261 Nov 30,2020 9:22am
F1 Help | Enter directory path and file-name mask

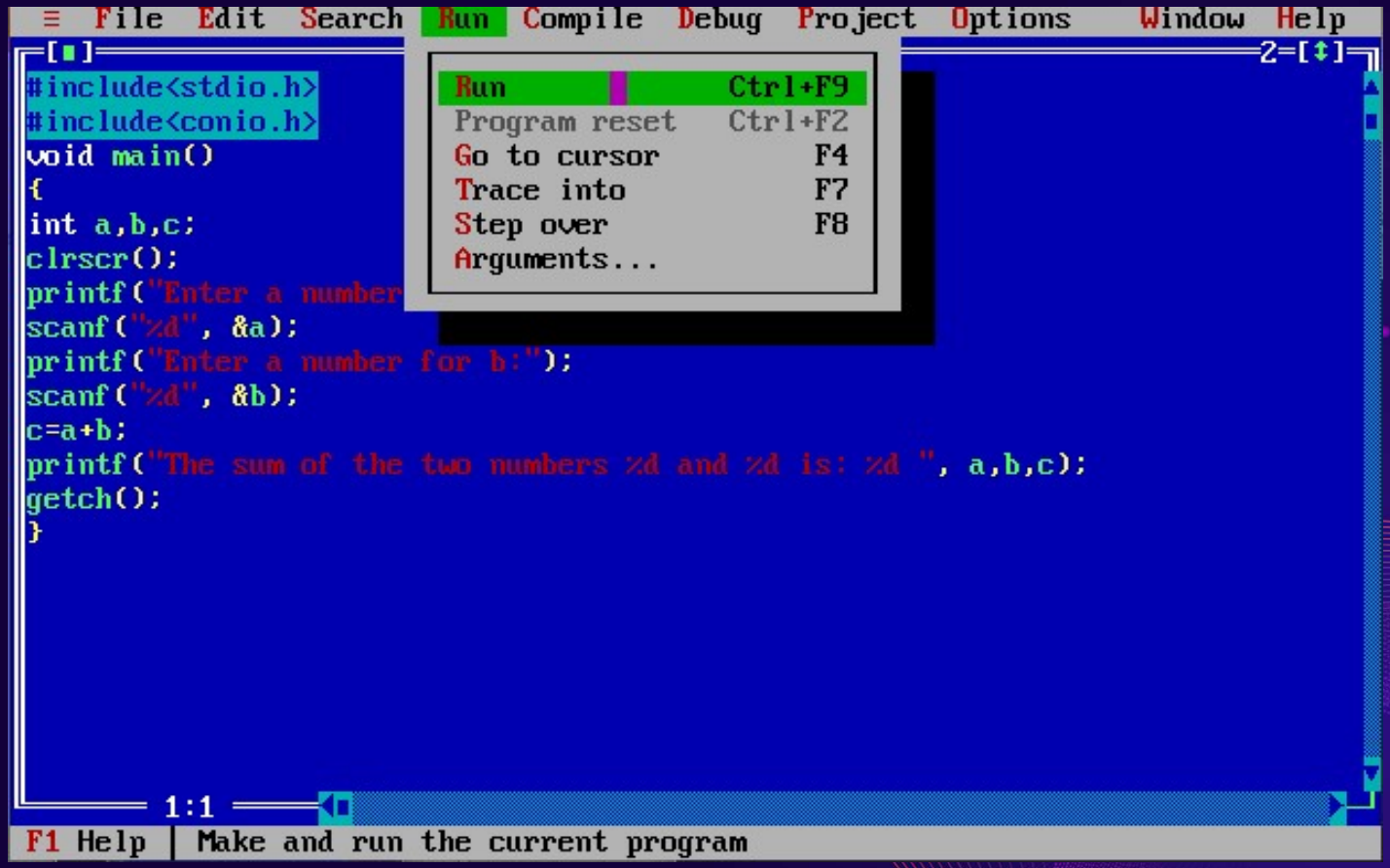
Example


File Edit Search Run Compile Debug Project Options Window Help
2
Compile Alt+F9
Make F9
Link
Build all
Information...
Remove messages
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
clrscr();
printf("Enter a number for
scanf("%d", &a);
printf("Enter a number for b:");
scanf("%d", &b);
c=a+b;
printf("The sum of the two numbers %d and %d is: %d ", a,b,c);
getch();
}
1:1
F1 Help | Compile the file in the active Edit window

Example


File Edit Search Run Compile Debug Project Options Window Help
PROGRAM1.C
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
clrscr();
printf("Enter a nu
scanf("%d", &a);
printf("Enter a nu
scanf("%d", &b);
c=a+b;
printf("The sum of
getch();
}
Compiling
Main file: PROGRAM1.C
Compiling: EDITOR → PROGRAM1.C
Total File
Lines compiled: 468 468
Warnings: 0 0
Errors: 0 0
Available memory: 1969K
Success : Press any key
1:1
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu

Example


File Edit Search Run Compile Debug Project Options Window Help
2=[↕]
#include<stdio.h>
#include<conio.h>
void main()
{
int a,b,c;
clrscr();
printf("Enter a number
scanf("%d", &a);
printf("Enter a number for b:");
scanf("%d", &b);
c=a+b;
printf("The sum of the two numbers %d and %d is: %d ", a,b,c);
getch();
}
Run Ctrl+F9
Program reset Ctrl+F2
Go to cursor F4
Trace into F7
Step over F8
Arguments...
1:1
F1 Help | Make and run the current program

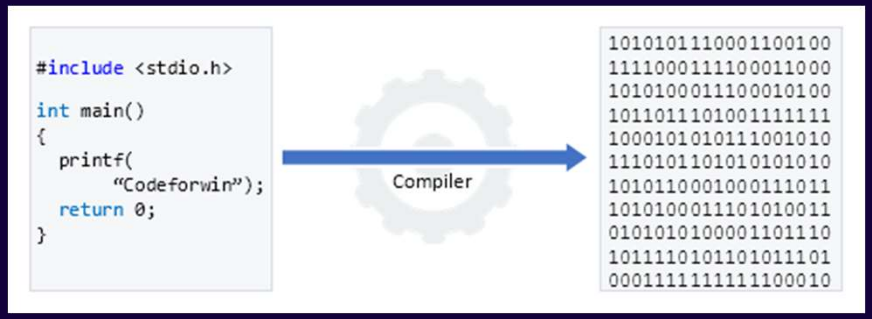
Example

```
Enter a number for a:12
Enter a number for b:3
The sum of the two numbers 12 and 3 is: 15
```


COMPILATION & LINKING

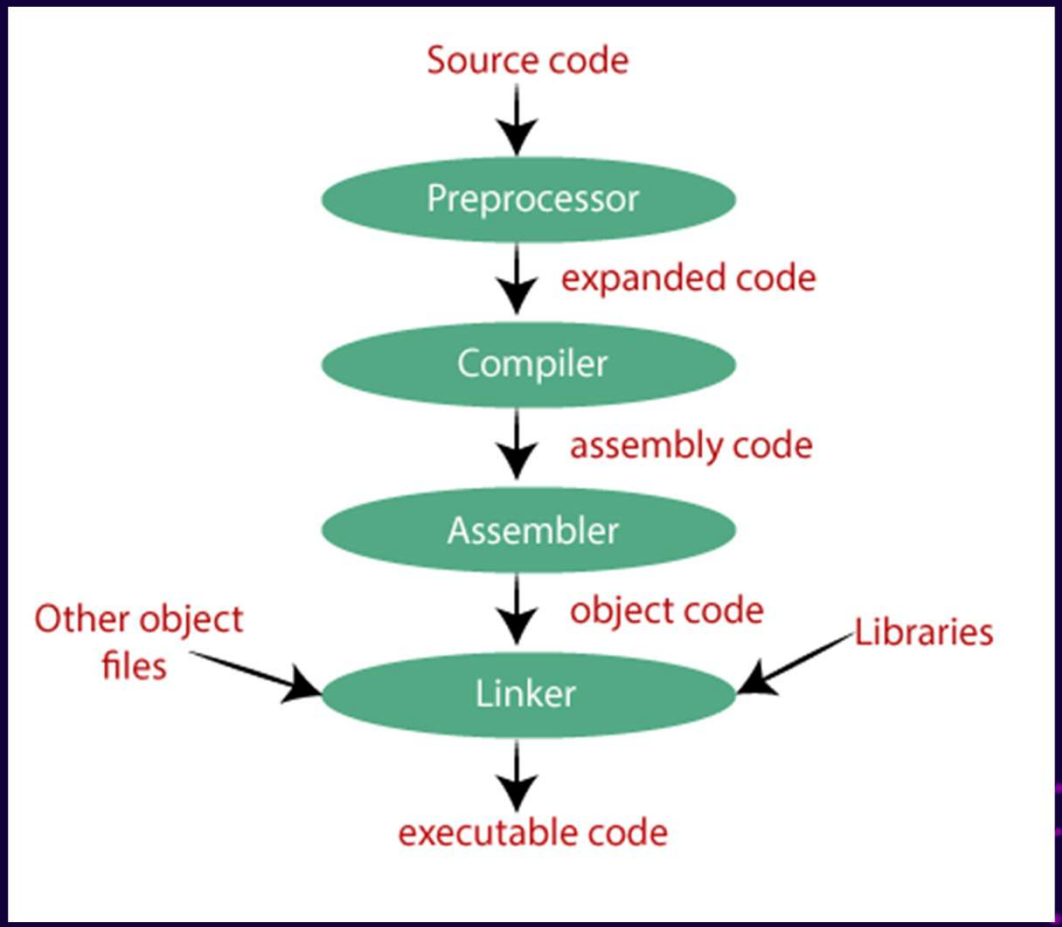
What is Compilation?

The compilation is a process of converting the source code into object code. It is done with the help of the compiler. The compiler checks the source code for the syntactical or structural errors, and if the source code is error-free, then it generates the object code



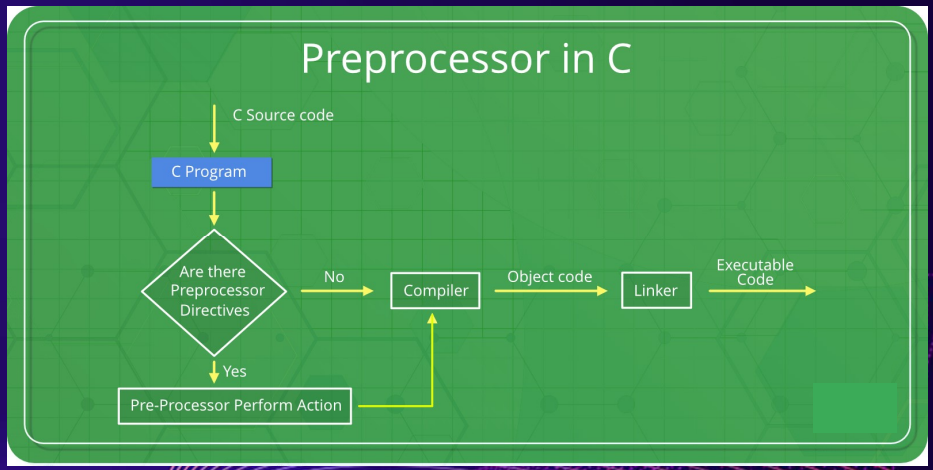
The c compilation process converts the source code taken as input into the object code or machine code. The compilation process can be divided into four steps, i.e., **Pre-processing, Compiling, Assembling,** and **Linking.**

Compilation process



Preprocessor

- ✓ The source code is the code which is written in a text editor and the source code file is given an extension ".c".
- ✓ The preprocessor takes the source code as an input, and it removes all the comments from the source code. The preprocessor takes the preprocessor directive and interprets it.
- ✓ For example, if **<stdio.h>,** the directive is available in the program, then the preprocessor interprets the directive and replace this directive with the content of the **'stdio.h'** file.



Compiler

- ✓ The code which is expanded by the preprocessor is passed to the compiler. The compiler converts this code into assembly code. Or we can say that the C compiler converts the pre-processed code into assembly code.
- ✓ The compiler won't give an error unless the source code is not well-formed.

Assembler

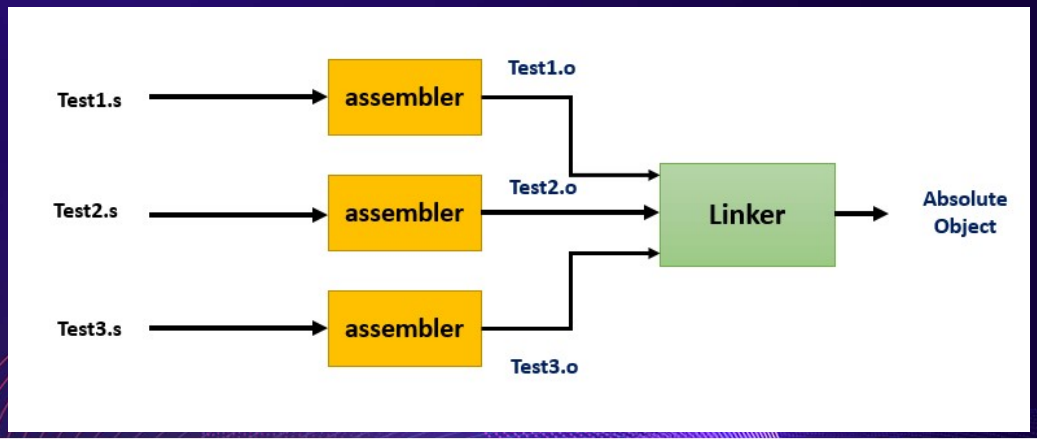
- ✓ The assembly code is converted into object code by using an assembler. The name of the object file generated by the assembler is the same as the source file.
- ✓ The extension of the object file in DOS is '.obj,' and in UNIX, the extension is 'o'.
- ✓ If the name of the source file is 'hello.c', then the name of the object file would be 'hello.obj'.
- ✓ These object files can be used as static libraries as well.

Linker

- ✓ Mainly, all the programs written in C use library functions. These library functions are pre-compiled, and the object code of these library files is stored with '.lib' (or '.a') extension.
- ✓ The main working of the linker is to combine the object code of library files with the object code of our program.
- ✓ Sometimes the situation arises when our program refers to the functions defined in other files; then linker plays a very important role in this.
- ✓ It links the object code of these files to our program.
- ✓ Therefore, we conclude that the job of the linker is to link the object code of our program with the object code of the library files and other files.

Linker (contd…)

- ✓ The output of the linker is the executable file.
- ✓ The name of the executable file is the same as the source file but differs only in their extensions.
- ✓ In DOS, the extension of the executable file is '.exe', and in UNIX, the executable file can be named as 'a.out'.
- ✓ For example, if we are using printf() function in a program, then the linker adds its associated code in an output file and sends to Loader.



Loader

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.

Example Program

hello.c

```
#include<stdio.h>
#include<conio.h>
int main()
{
printf("C programming");
return 0;
}
```

Flow of the program hello.c