

INTRODUCTION TO C

Sumaiyah Zahid

WHY C?

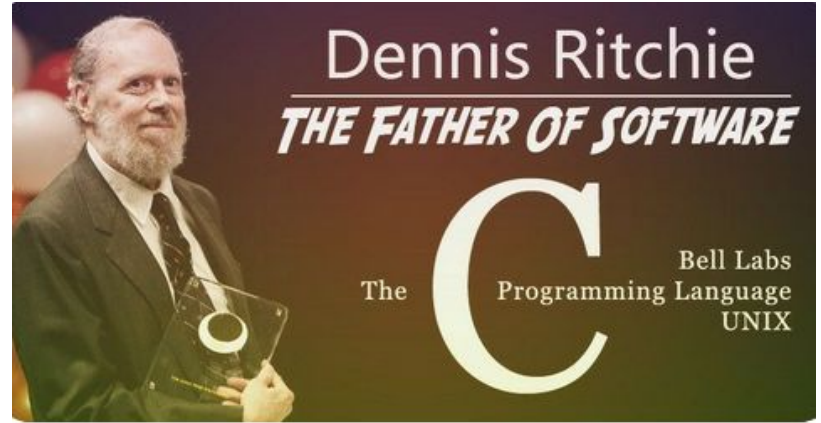
Small size

Easy to learn

Structured language

Middle Level Language

Pointer implementation – extensive use of pointers for memory, array, structures and functions.



Display “Hello World”

```
printf("hello, world");
```

Start

End

```
int main(void)
{
}
```


Start
Display "Hello World"
End

```
#include <stdio.h>

int main(void)
{
    printf("hello world ");
}
```

DESKTOP APPLICATION

Home / Browse / Development / Integrated Development Environments (IDE) / Dev-C++



Dev-C++

A free, portable, fast and simple C/C++ IDE
Brought to you by: [orwelldevcpp](#)

★★★★★ 134 Reviews

Downloads: 177,420 This Week

Last Update: 2016-11-29

[Download](#) [Get Updates](#) [Share This](#)

Windows | BSD


Summary	Files	Reviews	Support	External Link ▾	Tracker	Code	Forums
---------	-------	---------	---------	-----------------	---------	------	--------

A new and improved fork of Bloodshed Dev-C++

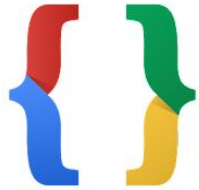
Features

- TDM-GCC 4.9.2 32/64bit
- Syntax highlighting
- Code completion
- Code insight
- Editable shortcuts
- GPROF profiling
- GDB debugging
- AStyle code formatting
- Devpak IDE extensions
- External tools

Project Samples



ANDROID APPLICATIONS



CppDroid - C/C++ IDE

Anton Smirnov Education

Everyone

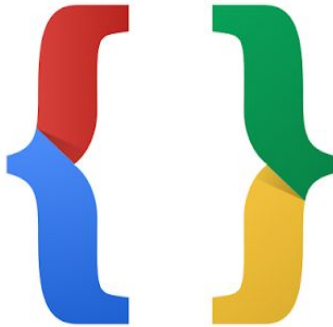
★★★★★ 34,848

Contains Ads - Offers in-app purchases

This app is available for your device

Add to Wishlist

Install



Cxxdroid - C++ compiler IDE for mobile development

IIEC Education

Everyone

★★★★★ 6,760

Contains Ads - Offers in-app purchases

This app is available for your device

Add to Wishlist

Install

Powerful offline compiler and neat design in one app



Supports most recent standards



Prebuilt libraries available in the repository



Interactive mode for terminal and advanced



IPHONE APPLICATIONS



Mobile C [C/C++ Compiler]

Lee Jeong Seop Education

★★★★★ 8,022

Everyone

Contains Ads

This app is available for your device

Add to Wishlist

Install



Learn C Coding
in Mobile Devices



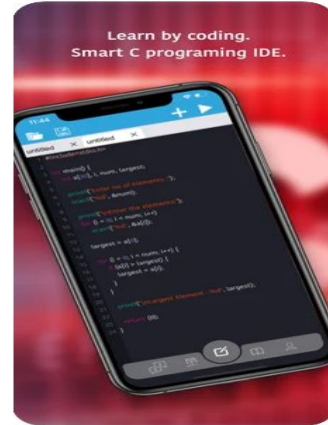
C Compiler 4+

C Compiler - Tutorial -Samples
sutheesh sukumaran
Designed for iPad

★★★★★ 4.1 • 67 Ratings

Free · Offers In-App Purchases

Screenshots iPad iPhone



Source Code
Human readable program

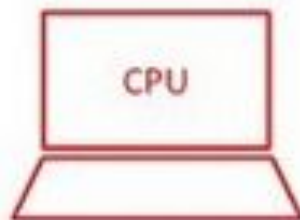


Compiler



Object Code
Executable machine code

```
101101101101
010111011011
011011100100
111111010101
011011011011
101101011011
```



CPU

WRITING A C PROGRAM

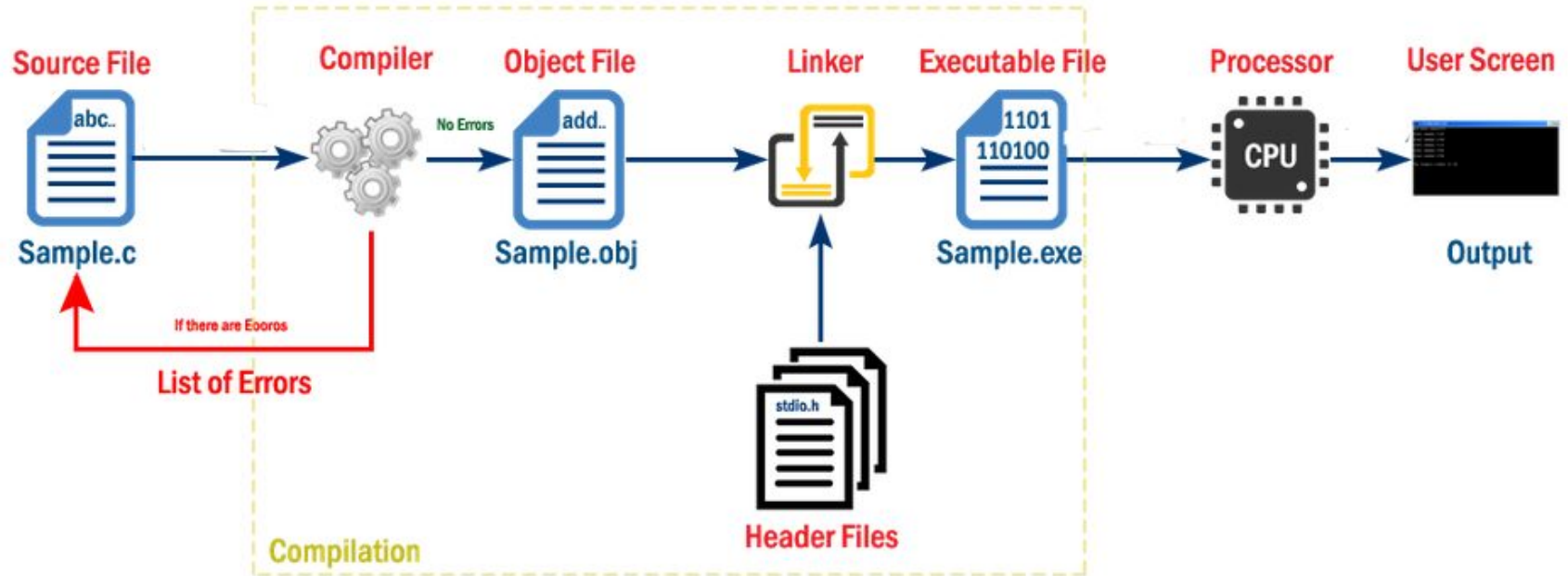
A programmer uses a text editor to create or modify files containing C code.

Code is also known as source code.

A file containing source code is called a source file.

After a C source file has been created, the programmer must invoke the C compiler before the program can be executed (run).

ARCHITECTURAL DIAGRAM



ANATOMY OF C PROGRAM

[illegible]

PROGRAM HEADER COMMENT

A comment is descriptive text used to help a reader of the program understand its content.

All comments must begin with the characters `/*` and end with the characters `*/`

These are called comment delimiters

The program header comment always comes first.

PREPROCESSOR DIRECTIVES

Lines that begin with a # in column 1 are called preprocessor directives (commands).

Example: the `#include <stdio.h>` directive causes the preprocessor to include a copy of the standard input/output header file `stdio.h` at this point in the code.

This header file was included because it contains information about the `printf ()` function that is used in this program.

STDIO.H

When we write our programs, there are libraries of functions to help us so that we do not have to write the same code over and over.

Some of the functions are very complex and long. Not having to write them ourselves make it easier and faster to write programs.

Using the functions will also make it easier to learn to program!

INT MAIN (VOID)

Every program must have a function called main. This is where program execution begins.

main() is placed in the source code file as the first function for readability.

The reserved word “int” indicates that main() returns an integer.

The parentheses following the reserved word “main” indicate that it is a function.

The reserved word “void” means nothing is there.

THE FUNCTION BODY

A left brace (curly bracket) { begins the body of every function.

A corresponding right brace } ends the function body.

```
PRINTF ("HELLO WORLD \N");
```

This line is a C statement.

It is a call to the function `printf ()` with a single argument (parameter), namely the string `"Hello, World!\n"`.

Even though a string may contain many characters, the string itself should be thought of as a single quantity.

Notice that this line ends with a semicolon. All statements in C end with a semicolon.

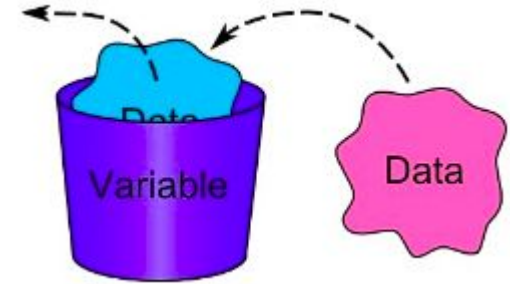
RETURN 0;

In C programs the main function is of type int and therefore it should return an integer value. The return value of the main function is considered the "Exit Status" of the application.

On most operating systems returning 0 is a success status like saying "The program worked fine".

VARIABLES

A variable is a space in the computer's memory set aside for a certain kind of data and given a name for easy reference.



Variable Declaration

e.g: `int x,y,z;`

`float a,b,c;`

`char z;`

VARIABLES

Declaration:

```
int x;
```

Initialization:

```
x=0;
```

Declaration & Initialization:

```
int x=0;
```

int age = 20; ← value

datatype variable_name

VARIABLES TYPES

Type	Memory (byte)	Range
char	1	-128 to 127
int	2	-32,768 to 32,767
long int	4	-2,147,483,648 to 2,147,483,647
float	4	10^{-38} to 10^{38} 7 digits precisions
double	8	10^{-308} to 10^{308} 7 digits precisions

Print “Variable is” , variable

```
printf(“Your variable is %d”, myvariable);
```

PRINTF()

The printf function has a special formatting character (%) -- a character following this defines a certain format for a variable:

%c - characters

%d - integers

%f - floats

e.g.

```
printf("%c %d %f",ch,i,x);
```

NOTE: Format statement enclosed in "...", variables follow after.
Make sure order of format and variable data types match up.

FORMAT SPECIFIER

`%c` Single Character

`%s` String

`%d` Signed decimal integer

`%f` Floating point (decimal notation)

`%e` Floating point (exponential notation)

`%u` Unsigned decimal integer

`%x` Unsigned hexadecimal integer

`%o` Unsigned octal integer

`l` prefix used with `%d`, `%u`, `%x`, `%o` to specify long integer (e.g. `%ld`) `%lf` long double

EXAMPLE

```
# include <stdio.h>
int main ()
{
    int age = 28;
    char gender = 'F';
    printf(" Hello there \n");
    printf("I am Sumaiyah and my age is %d", age);
    printf("My gender = %c" , gender);
}
```

TASKS

Write a code in C to calculate the area of a rectangle where length = 4 and width = 5.

Don't take inputs. Declare and initialize the variables.

TASKS

Write a code in C to calculate the percentage from subject marks.

Declare and initialize all subject marks by yourself.
Don't take inputs.

TASKS

Write a program that converts a temperature given in Celsius to Fahrenheit. The formula to convert Celsius to Fahrenheit is:

$$F = \left(\frac{9}{5} \times C \right) + 32$$

TASKS

If a four-digit number is 4567, write a program to obtain the sum of the first and last digit of this number.

ESCAPE SEQUENCE

The following list shows the common escape sequences:

- `\n` New line
- `\t` Tab
- `\b` Backspace
- `\r` Carriage return
- `\f` Form feed
- `\'` Single quote
- `\"` Double quote
- `\\` Backslash