

C Programming

Kasun De Zoysa

C Programming

- C is a powerful general-purpose programming language.
- It is fast, portable and available in all platforms.
- C language used for wide range of applications from Operating systems like Windows, Unix and iOS to software that is used for creating 3D movies.
- C programming is highly efficient. That's the main reason why it's very popular despite being more than 40 years old.
- If you are new to programming, C is a good choice to start your programming journey.

History of C programming

The PDP-11 version of Unix system was written in assembly language. Assembly languages (**A**) are low-level programming languages that are specific to a particular computer architecture. They are hard to write and understand.

The developers (Dennis Ritchie and Stephen C. Johnson) of Unix Operating system decided to rewrite the system in **B** language. However, B couldn't suffice some of the features of PDP-11, which led to the development of **C**.

History of C programming

In 1972, the development of **C** started on the PDP-11 Unix system. A large part of Unix was then rewritten in **C**.

By 1973, **C** was powerful enough to be used in Unix Kernel.

Dennis Ritchie and Stephen C. Johnson made further changes to the language for several years to make it portable in Unix Operating system.

Books on C Programming

In 1978, the first book of C programming, **The C Programming Language**, was published. The first edition of the book provided programmers informal specification of the language. Written by Brian Kernighan and Dennis Ritchie, this book is popular among C programmers as "K&R".

However, personally I believe, **C Programming: A Modern Approach** (2nd Edition) is the best book for beginners.

Materials available at

"Learn C Programming - The Definitive Guide"

<https://www.programiz.com/c-programming> is used in this course.

What will you gain if you learn C?

- You will understand how a computer works.
- You will write better programs.
- You will find it much easier to learn other programming languages.
- Opportunity to work on open source projects that impact millions of people.

If you are busy and don't want to invest time on something that doesn't have direct effect on your day-to-day work, C programming is not for you.

The fun begins:

Your first C program

```
#include <stdio.h>
int main()
{
    printf("Ayubowan C!\n");
    return 0;
}
```

```
root@d3103c6a9341:/home# gcc -o ayubowan. ayubowan.c
root@d3103c6a9341:/home# ./ayubowan.
Ayubowan C!
root@d3103c6a9341:/home#
```

Your first C program

In this program, we have used `printf()` **function** which displays the text inside the quotation mark. Since **`printf()`** is defined in **`stdio.h`**, you need to include `stdio.h`.

In C programming, the code execution begins from the start of `main()` function (doesn't matter if `main()` isn't located at the beginning).

The code inside the curly braces `{ }` is the body of `main()` function. The `main()` function is **mandatory** in every C program.

Your first C program

The `printf()` is a library function that sends formatted output to the screen (displays the string inside the quotation mark).

Notice the semicolon at the end of the statement.

The return statement **`return 0;`** inside the `main()` function ends the program. This statement isn't mandatory. However, it's considered good programming practice to use it.

Key notes to take away

- All C program starts from the **main()** function and it's **mandatory**.
- You can use the required header file that's necessary in the program.
- C is **case-sensitive**; the use of uppercase letter and lowercase letter have different meanings.
- The C program ends when the program encounters the return statement inside the main() function.
- The statement in a C program ends with a **semicolon**.

Docker

Docker file

```
FROM gcc
```

```
MAINTAINER Kasun De Zoysa
```

```
RUN apt-get update && apt-get install
```

```
-y \
```

```
vim \
```

```
gdb \
```

```
git
```

Build a Docker image

```
docker build -t gcc .
```

Run Docker image call gcc and name it as gcc

```
docker run -v ~/prg:/home --name gcc -it gcc
```

Docker

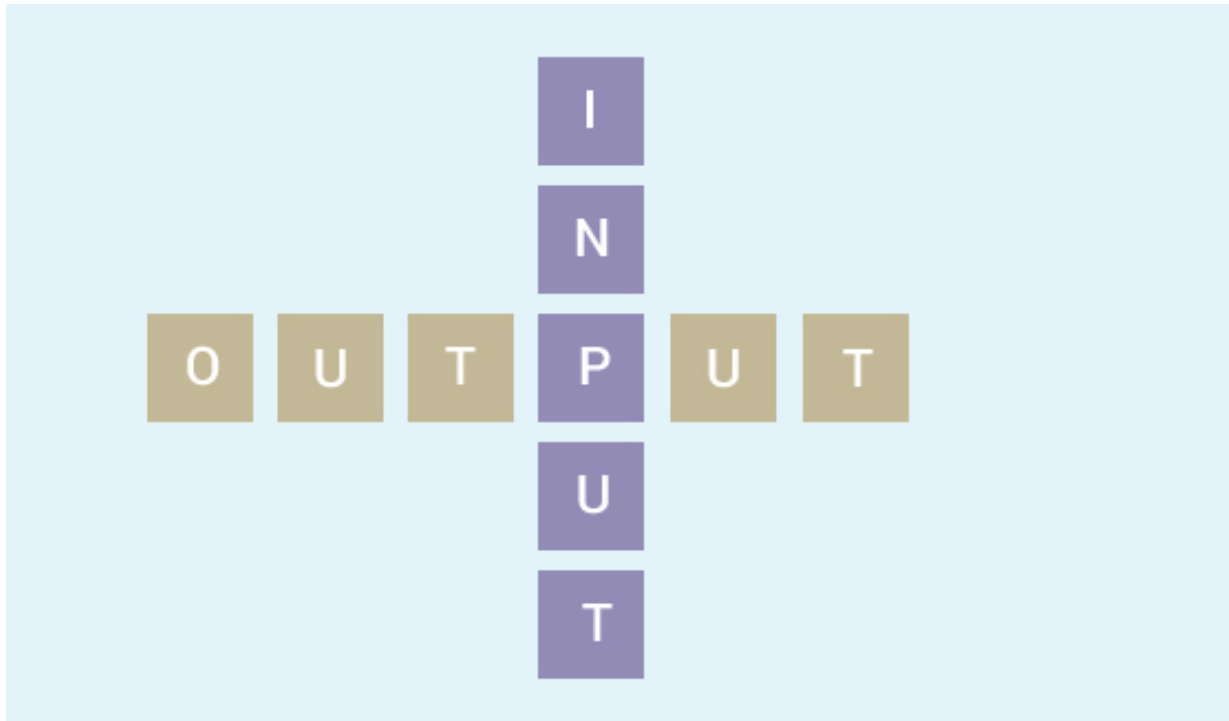
```
× docker
Kasuns-MacBook-Air:prg kasundezoysa$ clear
Kasuns-MacBook-Air:prg kasundezoysa$ docker run -v ~/prg:/home --name gcc -it gcc
root@d3103c6a9341:/# cd /home/
root@d3103c6a9341:/home# ls
Dockerfile
root@d3103c6a9341:/home# vim ayubowan.c
```

```
× docker
#include <stdio.h>

int main()
{
    printf("Ayubowan C!\n");
    return 0;
}

~
~
~
~
~
~
:WQ
```

```
root@d3103c6a9341:/home# vim ayubowan.c
root@d3103c6a9341:/home# ls
Dockerfile  ayubowan.c
root@d3103c6a9341:/home# gcc -o ayubowan. ayubowan.c
root@d3103c6a9341:/home# ./ayubowan.
Ayubowan C!
root@d3103c6a9341:/home#
```



C programming has several in-built library functions to perform input and output tasks.

Two commonly used functions for I/O (Input/Output) are `printf()` and `scanf()`.

The `scanf()` function reads formatted input from standard input (keyboard) whereas the `printf()` function sends formatted output to the standard output (screen).

C Output

```
#include <stdio.h>          //This is needed to run printf() function.
int main()
{
    printf("C Programming"); //displays the content inside quotation
    return 0;
}
```

```
#include <stdio.h>
int main()
{
    int testInteger = 5;
    printf("Number = %d", testInteger);
    return 0;
}
```

Inside the quotation of printf() function, there is a format string "%d" (for integer). If the format string matches the argument (testInteger in this case), it is displayed on the screen.

How this program works?

All valid C program must contain the `main()` function. The code execution begins from the start of `main()` function.

The `printf()` is a library function to send formatted output to the screen.

The `printf()` function is declared in "`stdio.h`" header file. Here, `stdio.h` is a header file (standard input output header file) and `#include` is a preprocessor directive to paste the code from the header file when necessary.

When the compiler encounters `printf()` function and doesn't find `stdio.h` header file, compiler shows error.

The `return 0;` statement is the "Exit status" of the program. In simple terms, program ends.

Comments

As programs get bigger and more complicated, they get more difficult to read. Formal languages are dense, and it is often difficult to look at a piece of code and figure out what it is doing, or why.

For this reason, it is a good idea to add notes to your programs to explain in natural language what the program is doing.

These notes are called comments , and they start with The // symbol or /* */ for C.

C Integer Input/Output

```
#include <stdio.h>
int main()
{
    int testInteger;
    printf("Enter an integer: ");
    scanf("%d",&testInteger);
    printf("Number = %d",testInteger);
    return 0;
}
```

The `scanf()` function reads formatted input from the keyboard. When user enters an integer, it is stored in variable `testInteger`.

Note the '`&`' sign before `testInteger`; `&testInteger` gets the address of `testInteger` and the value is stored in that address.

C Floats Input/Output

```
#include <stdio.h>
int main()
{
    float f;
    printf("Enter a number: ");
    // %f format string is used in case of floats
    scanf("%f",&f);
    printf("Value = %f", f);
    return 0;
}
```

The format string "%f" is used to read and display formatted in case of floats.

C Character I/O

```
#include <stdio.h>
int main()
{
    char chr;
    printf("Enter a character: ");
    scanf("%c",&chr);
    printf("You entered %c.",chr);
    return 0;
}
```

Format string %c is used in case of character types.

C ASCII Code

```
#include <stdio.h>
int main()
{
    char chr;
    printf("Enter a character: ");
    scanf("%c",&chr);

    // When %c text format is used, character is displayed
    printf("You entered %c.\n",chr);

    // When %d text format is used, integer is displayed i
    printf("ASCII value of %c is %d.", chr, chr);
    return 0;
}
```



Sign in to **GitHub**
to continue to **GitHub Classroom**

Username or email address

Password

[Forgot password?](#)

Sign in

New to GitHub? [Create an account.](#)



Authorize GitHub Classroom



GitHub Classroom by **github**

wants to access your **sathindu** account



Personal user data

Email addresses (read-only)



Repositories



Authorize github

Authorizing will redirect to
<https://classroom.github.com>

UCSC Programming Courses

@ucslabs



Accepted the **IS1101** assignment

You are ready to go!

You may receive an invitation to join [@ucslabs](#) via email invitation on your behalf. No further action is necessary.

Your assignment has been created here: <https://github.com/ucslabs/is1101-sathindu>

UCSC Programming Courses

@ucslabs



Accept the **IS1101** assignment

Accepting this assignment will give you access to the **is1101-sathindu** repository in the [@ucslabs](#) organization on GitHub.

Accept this assignment

UCSC Programming Courses

@ucslabs



Accepted the **is1101** assignment

You are ready to go!

You may receive an invitation to join [@ucslabs](#) via email invitation on your behalf. No further action is necessary.

Your assignment has been created here: <https://github.com/ucslabs/is1101-nandulee>

12 commits

1 branch

0 releases

1 contributor

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Clone or download ▾



kasundezoyosa Net activity ...

lec1	Finish
lec2	add Dockerfile
Dockerfile	Add Dockerfile
README.md	Update README.md

Clone with HTTPS ?

[Use SSH](#)

Use Git or checkout with SVN using the web URL.

<https://github.com/ucslabs/is1101-nandulee>



Copied!

Open in Desktop

Download ZIP

10 hours ago

× docker

```
root@d3103c6a9341:/home# git clone https://nandulee@github.com/ucscclabs/is1101-nandulee
.git
Cloning into 'is1101-nandulee'...
Password for 'https://nandulee@github.com':
remote: Counting objects: 42, done.
remote: Compressing objects: 100% (24/24), done.
remote: Total 42 (delta 9), reused 42 (delta 9), pack-reused 0
Unpacking objects: 100% (42/42), done.
root@d3103c6a9341:/home# ls
Dockerfile  ayubowan.  ayubowan.c  is1101-nandulee
root@d3103c6a9341:/home# cd is1101-nandulee/
root@d3103c6a9341:/home/is1101-nandulee# ls
Dockerfile  README.md  array.c  docker.txt  lec1  lec2
root@d3103c6a9341:/home/is1101-nandulee#
```

×

docker

```
root@d3103c6a9341:/home/is1101-nandulee# cd lec1/
root@d3103c6a9341:/home/is1101-nandulee/lec1# ls
ayumowan  ayumowan.c
root@d3103c6a9341:/home/is1101-nandulee/lec1# vim ayumowan.c
```

docker

```
root@d3103c6a9341:/home/is1101-nandulee/lec1# gcc -o ayumowan ayumowan.c
root@d3103c6a9341:/home/is1101-nandulee/lec1# ./ayumowan
Ayubowan KASUN
root@d3103c6a9341:/home/is1101-nandulee/lec1#
```

×

docker

```
#include <stdio.h>

int main(){
    printf("Ayubowan KASUN\n");
    return 0;
}

~
~
~
~
~
~
~
~
~
```

:WC

× docker

```
root@d3103c6a9341:/home/is1101-nandulee/lec1# git commit
```

```
On branch master
```

```
Your branch is up-to-date with 'origin/master'.
```

```
Changes not staged for commit:
```

```
    modified:   ayumowan
```

```
    modified:   ayumowan.c
```

```
no changes added to commit
```

```
root@d3103c6a9341:/home/is1101-nandulee/lec1# git commit -a
```

× docker

```
My first commit
```

```
# Please enter the commit message for your changes. Lines starting  
# with '#' will be ignored, and an empty message aborts the commit.
```

```
# On branch master
```

```
# Your branch is up-to-date with 'origin/master'.
```

```
#
```

```
# Changes to be committed:
```

```
#    modified:   ayumowan
```

```
#    modified:   ayumowan.c
```

```
#
```

```
~
```

```
~
```

```
~
```

```
-- INSERT --
```

1,16

All

```

X docker
root@d3103c6a9341:/home/is1101-nandulee/lec1# git commit -a
[master c077093] My first commit
 2 files changed, 1 insertion(+), 1 deletion(-)
root@d3103c6a9341:/home/is1101-nandulee/lec1# git push
Password for 'https://nandulee@github.com':
Counting objects: 5, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 571 bytes | 0 bytes/s, done.
Total 5 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/ucscclabs/is1101-nandulee.git
    00866f2..c077093  master -> master
root@d3103c6a9341:/home/is1101-nandulee/lec1#
```

2 contributors  

7 lines (5 sloc) | 74 Bytes

```

1  #include <stdio.h>
2
3  int main(){
4      printf("Ayubowan KASUN\n");
5      return 0;
6  }
```

Problems

1. Write a C program to read and print your name
2. Write a C program to add two Integers

