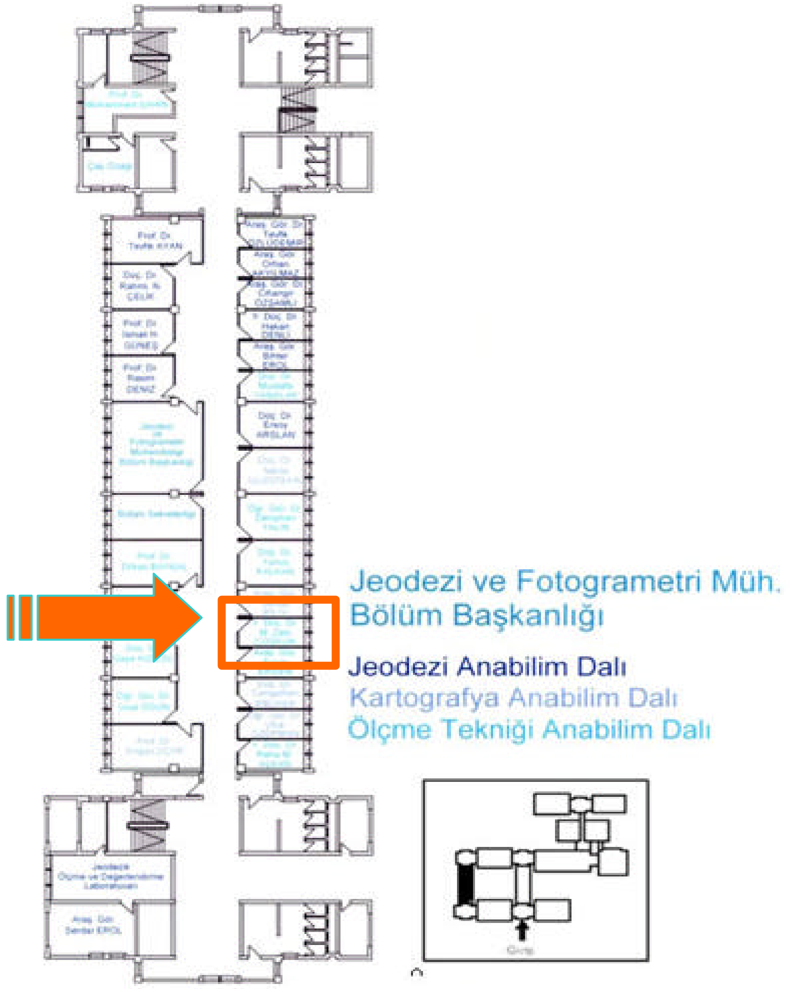
Instructor

- **Mehmet Zeki COSKUN**
- Assistant Professor
  at the
  Geodesy & Photogrammetry, Civil Eng.
- (212) 285-6573
- **coskunmeh@itu.edu.tr**
- http://atlas.cc.itu.edu.tr/~coskun

Address


Jeodezi ve Fotogrametri Müh.
Bölüm Başkanlığı
Jeodezi Anabilim Dalı
Kartografya Anabilim Dalı
Ölçme Tekniği Anabilim Dalı

Consultation of Students:

Monday 10:00 – 12:00

Course WEB site

- Lecture Notes
  - http://atlas.cc.itu.edu.tr/~coskun

- Compiler
  - http://atlas.cc.itu.edu.tr/~coskun

- Samples
  - http://www.be.itu.edu.tr/kaynak/c

Working Environment

- Windows 95, 98, 2000, NT, XP
- Linux
  - Browser, etc.
  - Compiler
  - Editor

Lecture Notes

- What you will be reading on this screen will always be available at the Web site

- Taking notes: it’s up to you to decide!

The following slides are compiled from free version of the books:

Text Books:

Sams Teach Yourself C in 21 Days (Sams teach Yourself) By Peter

Aitken Sams Teach Yourself C in 24 hours (Published by Sams) By Tony Zhang.. .

The Electronic Versions of the books:

http://server11.hypermart.net/davidbook901/data/c/c1f8c0d9.htm
http://www.informit.com/itlibrary
http://www.free-book.co.uk/computers-internet/programming/c/
http://aelinik.free.fr/c/

Schedule

First two hours for Lecture

Last two hours Lab.

Requirements

[4 or more HOMEWORKS = 20%]

  - Pick it up from to Web site; turn it in by

- **Quizes [2 QUIZES = 15%]**
  - at any time
- **Midterm [1Midterm = 25%]**
  - will inform
- **Final [40%]**

BIL104E: Introduction to Scientific and Engineering Computing
Lecture 1

- ❖Inroduction
- ❖Getting Started
- ❖Writing your First C Program

Computing systems

- Central Processing Unit (CPU) [has registers]
- Memory (these days, RAM, Random Access Memory]
- Input/Output units: video card+monitor, keyboard, etc.
- Storage units [disks]

Computer memory organization

- Units of measure
  - Basic unit: bit
  - 8 bits = 1 byte
  - 1024 bytes = 1 Kbyte or 1 K
  - 1024 K = 1 Megabyte or 1 M

- CPU registers, RAM, ROM [**R**ead-**O**nly **M**emory]

- *n* bytes = 1 word (*n*=4, 8, *etc*.)
  ⇨ 1 word = 32 bits, or 64 bits, *etc*.

Computer memory organization

- Associated with each word or byte is an **address**
- As far as the computer is concerned, instructions are data

Languages

- Three types of programming languages

    1. Machine languages

```
1100000100
1000101011
1110101010
```

    2. Assembly languages
        - elementary computer operations

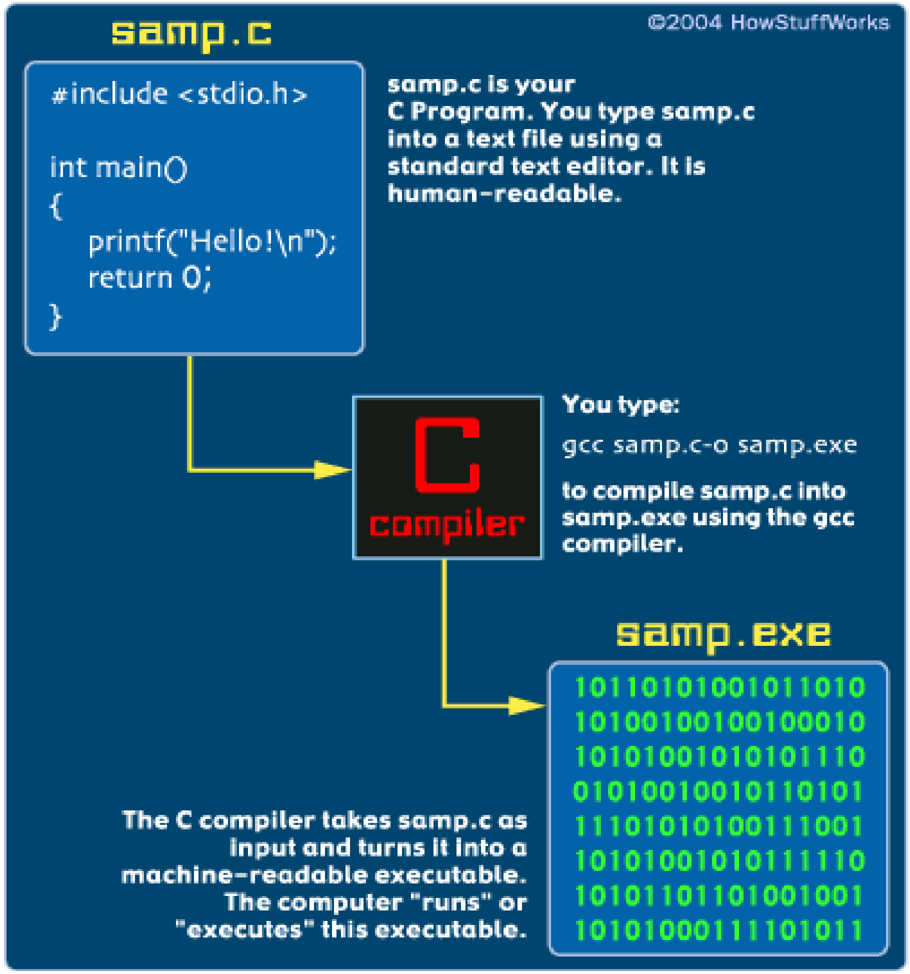
```
LOAD   BASEPAY
ADD    OVERPAY
STORE  GROSSPAY
```

Languages

3. High-level languages
   - Codes similar to everyday English
   - Use mathematical notations (translated via compilers)

```
grossPay = basePay + overTimePay
```

How a code runs



```c
#include <stdio.h>

int main()
{
    printf("Hello!\n");
    return 0;
}
```

samp.c is your C Program. You type samp.c into a text file using a standard text editor. It is human-readable.

You type:

```
gcc samp.c-o samp.exe
```

to compile samp.c into samp.exe using the gcc compiler.

```
1011010100101 1010
```

```
10110101001011010
10100100100100010
10101001010101110
01010010010110101
11101010100111001
10101001010111110
10101101101001001
10101000111101011
```

The C compiler takes samp.c as input and turns it into a machine-readable executable. The computer "runs" or "executes" this executable.

A Brief History of the C Language

- C was created by Dennis Ritchie at the Bell Telephone Laboratories in 1972.
- The language wasn't created for the fun of it, but for a specific purpose: to design the UNIX operating system (which is used on many computers). From the beginning,
- C was intended to be useful to allow busy programmers to get things done.

A Brief History of the C Language

Now, what about the name?

- The C language is so named because its predecessor was called B.
- The B language was developed by Ken Thompson of Bell Labs.
- You should be able to guess why it was called B.

- Programmers everywhere began using it to write all sorts of programs.
- Soon, however, different organizations began utilizing their own versions of C, and subtle differences between implementations started to cause programmers hackaches.
- In response to this problem, the American National Standards Institute (ANSI) formed a committee in 1983 to establish a standard definition of C, which became known as ANSI Standard C. With few exceptions, every modern C compiler has the ability to adhehere to this standard.

Why Use C?

- powerful and flexible.
- a popular language
- a portable language
- a language of few words, containing only a handful of terms
- modular

- What about C++?

- You might have heard about C++ and the programming technique called *object-oriented programming*. Perhaps you're wondering what the differences are between C and C++ and whether

C++ is a superset of C, which means that C++ contains everything C does, plus new additions for object-oriented programming.

Preparing to Program

- Determine the objective(s) of the program.
- Determine the methods you want to use in writing the program.
- Create the program to solve the problem.
- Run the program to see the results.

The Program Development Cycle

- **Step 1** Use an editor for write your source code. By tradition, C source code files have the extension “*.C” (for example, MYPROG.C, DATABASE.C, and so on).

- **Step 2** Compile the program using a compiler. If the compiler doesn't find and errors in the program, it produces an object file. The compiler produces object files with an .OBJ extension and the save name as the source code file (for example, MYPROG.C compiles to MYPPOG.OBJ). If the compiler finds errors, it reports them. You must return to step 1 to make corrections in your source code.

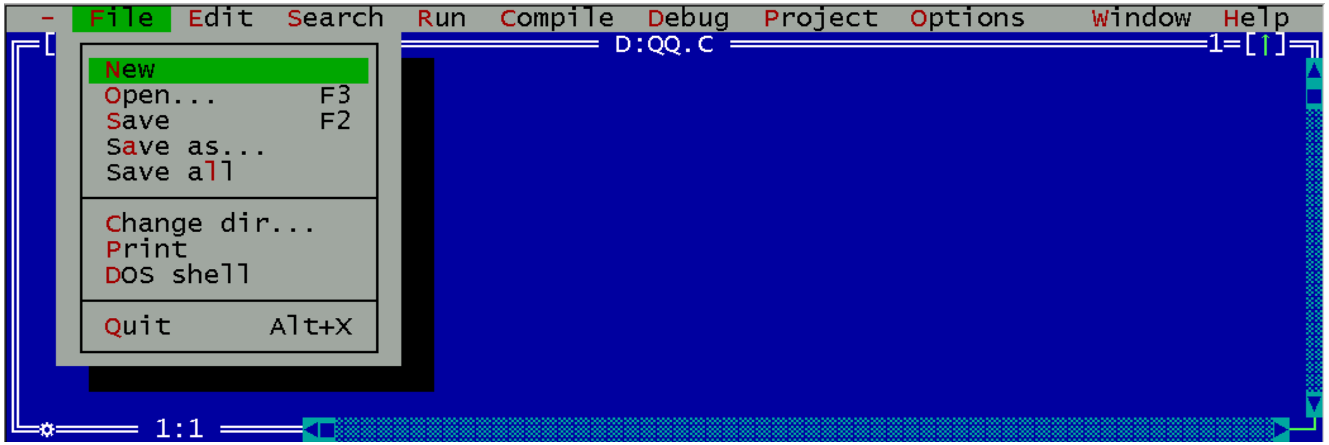
- **Step 3** Link the program using a linker. If no errors occur, the linked produces an executable program located in a disk file with an .EXE extension and the same name as the object file (for example, MYPROG.OBJ is linked to create MYPROG.EXE).

- **Step 4** Execute the program. You should test to determine whether it functions properly. If not, start again with step 1 and make modifications and additions to your source code.

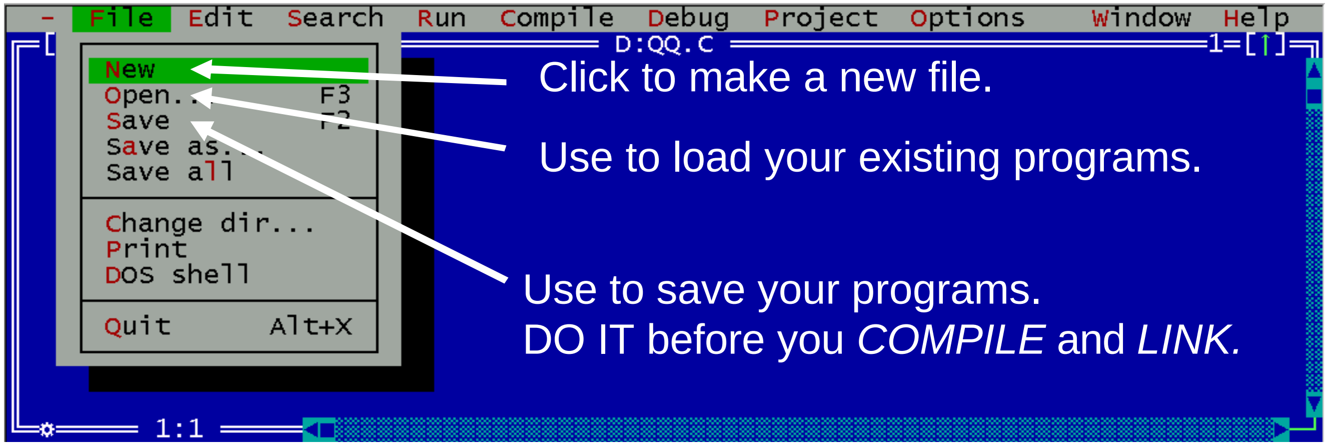
The C Standard Library

- C programs consist of pieces/modules called functions
  - A programmer can create his/her own functions
    - Advantage: the programmer knows exactly how it works
    - Disadvantage: time consuming
  - Programmers will often use the C library functions
    - Use these as building blocks
  - Avoid re-inventing the wheel
    - If a premade function exists, generally best to use it rather than write your own
    - Library functions carefully written, efficient, and portable

Interface of C Editor



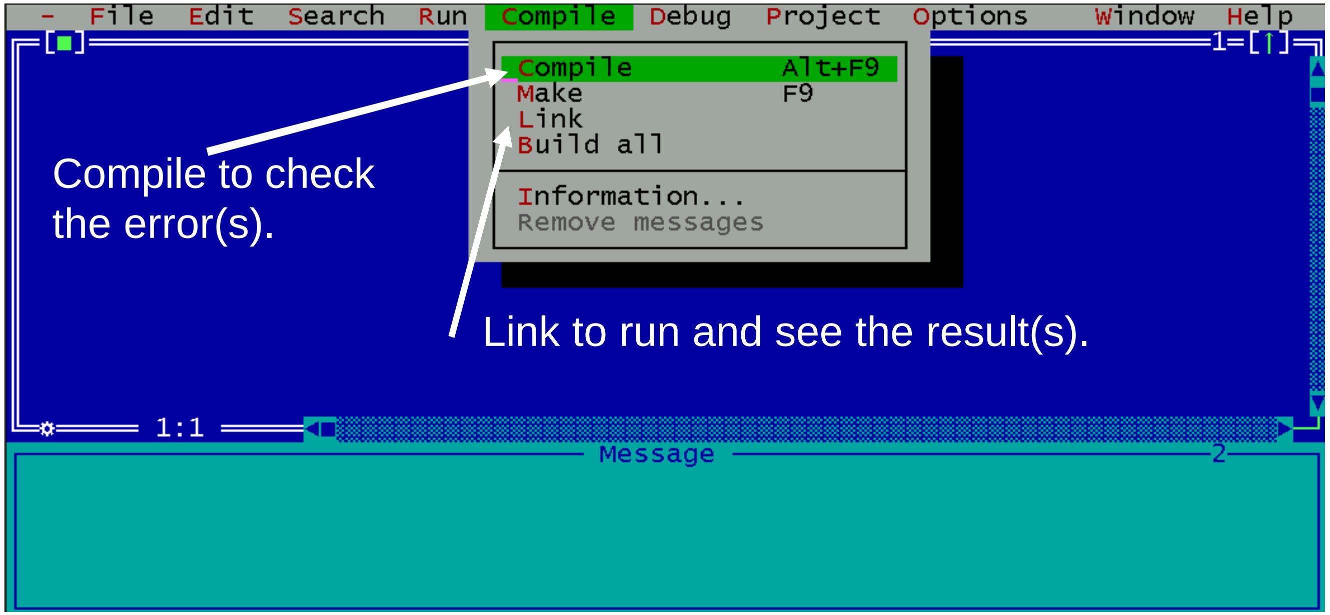
Interface of C Editor



Click to make a new file.

Use to load your existing programs.

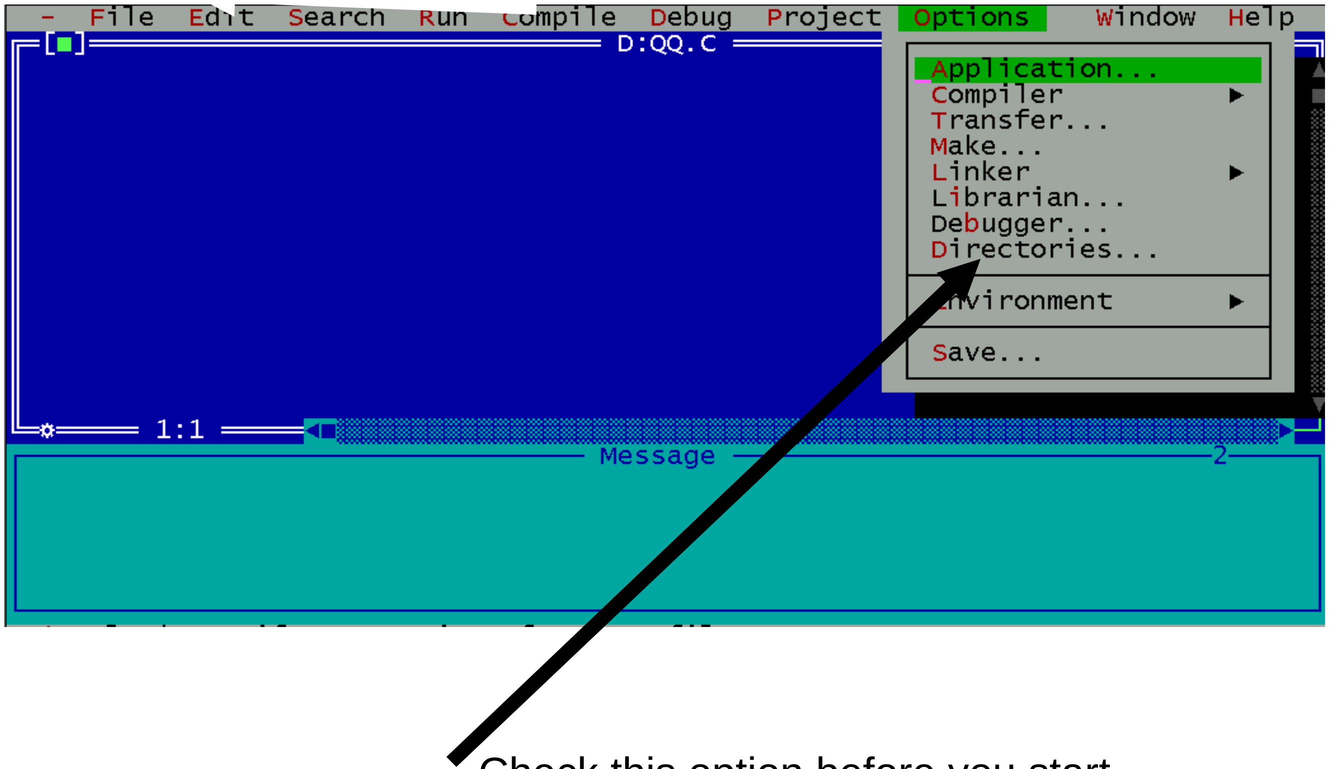
Use to save your programs.
DO IT before you *COMPILE* and *LINK*.



Compile to check
the error(s).

Link to run and see the result(s).

Interface of C Editor



Check this option before you start.

Examples of C programs: Our first program

```
  /* A first program in C */
#include <stdio.h>

int main()
{
   printf( "Welcome to C!\n" );

   return 0;
}
```

```
Output for the program is:
Welcome to C!
```

```
1
2    /* A first program in C */
3 #include <stdio.h>
4
5 int main()
6 {
7    printf( "Welcome to C!\n" );
8
9    return 0;
10}
```

Line 1: a blank line. Compiler does nothing... Line 1 is ignored by computer.

```
1
2    /* A first program in C */
3  #include <stdio.h>
4
5  int main()
6  {
7     printf( "Welcome to C!\n" );
8
9     return 0;
10}
```

Line 2: Text surrounded by ***/**** and ****/*** is ignored by computer. Used to describe program

```
1
2    /* A first program in C */
3  #include <stdio.h>
4
5  int main()
6  {
7     printf( "Welcome to C!\n" );
8
9     return 0;
10 }
```

Line 3: A header file. Preprocessor directive. Tells computer to load contents of a certain file.

```
1
2    /* A first program in C */
3 #include <stdio.h>
4
5 int main()
6 {
7    printf( "Welcome to C!\n" );
8
9    return 0;
10}
```

Line 3: C programs contain one or more functions, exactly one of which must be **`main.`** Parenthesis used to indicate a function. **`int`** means that **`main`** "returns" an integer value

```c
1
2    /* A first program in C */
3 #include <stdio.h>
4
5 int main()
6 {
7    printf( "Welcome to C!\n" );
8
9    return 0;
10}
```

Line 6: Braces (**{** and **}**) indicate a block. The bodies of all functions must be contained in braces. **{** means block (main program body) starts.

```
1
2    /* A first program in C */
3 #include <stdio.h>
4
5 int main()
6 {
7    printf( "Welcome to C!\n" );
8
9    return 0;
10}
```

Line 7: Instructs computer to perform an action. Specifically, prints the string of characters within quotes (“ ”). All statements must end with a semicolon (**;**). Escape character (****): Indicates that printf should do something out of the ordinary... **\n** is the newline character.

```
1
2    /* A first program in C */
3  #include <stdio.h>
4
5  int main()
6  {
7     printf( "Welcome to C!\n" );
8
9     return 0;
10 }
```

Line 9: A way to exit a function. **`return 0`**, in this case, means that the program terminated normally.

```
1
2    /* A first program in C */
3 #include <stdio.h>
4
5 int main()
6 {
7    printf( "Welcome to C!\n" );
8
9    return 0;
10}
```

Line 9: right brace **{** indicates end **main** of has reached.

Examples of C programs: Our second program

(terminating the program with exit() function)

```c

  /* A first program in C */
#include <stdio.h>

main()
{
   printf( "Welcome to C!\n" );

   exit(0);
}
```

```
Output for the program is:
Welcome to C!
```

Examples of C programs: Our second program

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

main ( ){   /* note that we did not use int */
   float x=125;
   double y;
   y = pow(x,3);   /* returns 125 cubed */
   printf(“Hello World\n”);
   exit(0);
}
```

- **`()`** shows a main or a function
- program execution begins at **`main()`**
- keywords are written in lower-case
- C is case sensitive, use lower-case and try not to capitalise variable names
- statements are terminated with a semi-colon **`;`**
- text strings are enclosed in double quotes (**“ ”**)
- **`\n`** means position the cursor on the beginning of the next line
- **`printf()`** can be used to display text to the screen
- the curly braces **`{}`** define the beginning and end of a program block

General Notes About C

- Program clarity
  - Programs that are convoluted are difficult to read, understand, and modify
- C is a portable language
  - Programs can run on many different computers
  - However, portability is an elusive goal
- We will do a careful walkthrough of C
  - Some details and subtleties are not covered
  - If you need additional technical details
    - Read the C standard document
    - Read the book by Kernigan and Ritchie