

# CEng 230 Introduction to C Programming

Seyyit Alper SERT

Department of Computer Engineering  
2017-2018 Fall

Web Pages

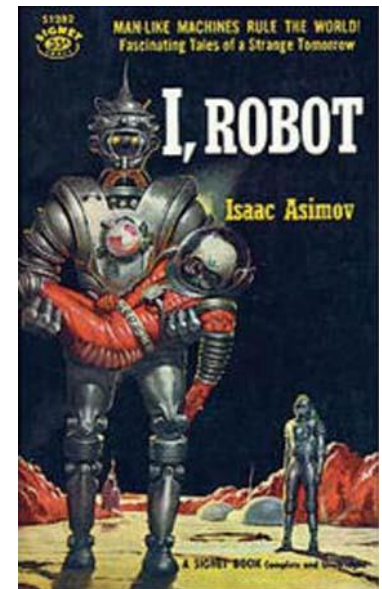
Official Course Page: [ceng230.ceng.metu.edu.tr](http://ceng230.ceng.metu.edu.tr)

Learning Management System (LMS): [odtuclass.metu.edu.tr](http://odtuclass.metu.edu.tr)

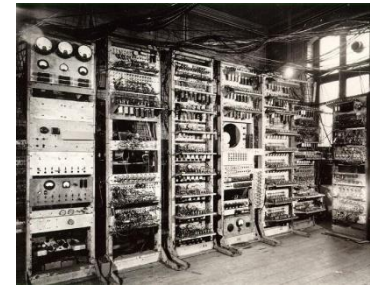
Contact: [alper.sert@ceng.metu.edu.tr](mailto:alper.sert@ceng.metu.edu.tr)

**"All the high-school students will be taught the fundamentals of computer technology will become proficient in binary arithmetic and will be trained to perfection in the use of the computer languages "**

Science fiction writer *Isaac Asimov's* Predictions  
For 2014 From in 1960s



- **COMPUTER**



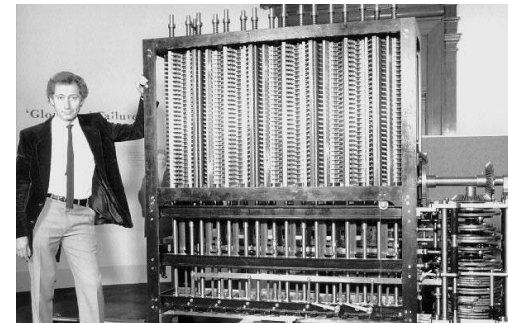
- Device capable of performing computations and making logical decisions
- Computers process data under the control of sets of instructions called *computer programs*

- **Hardware**

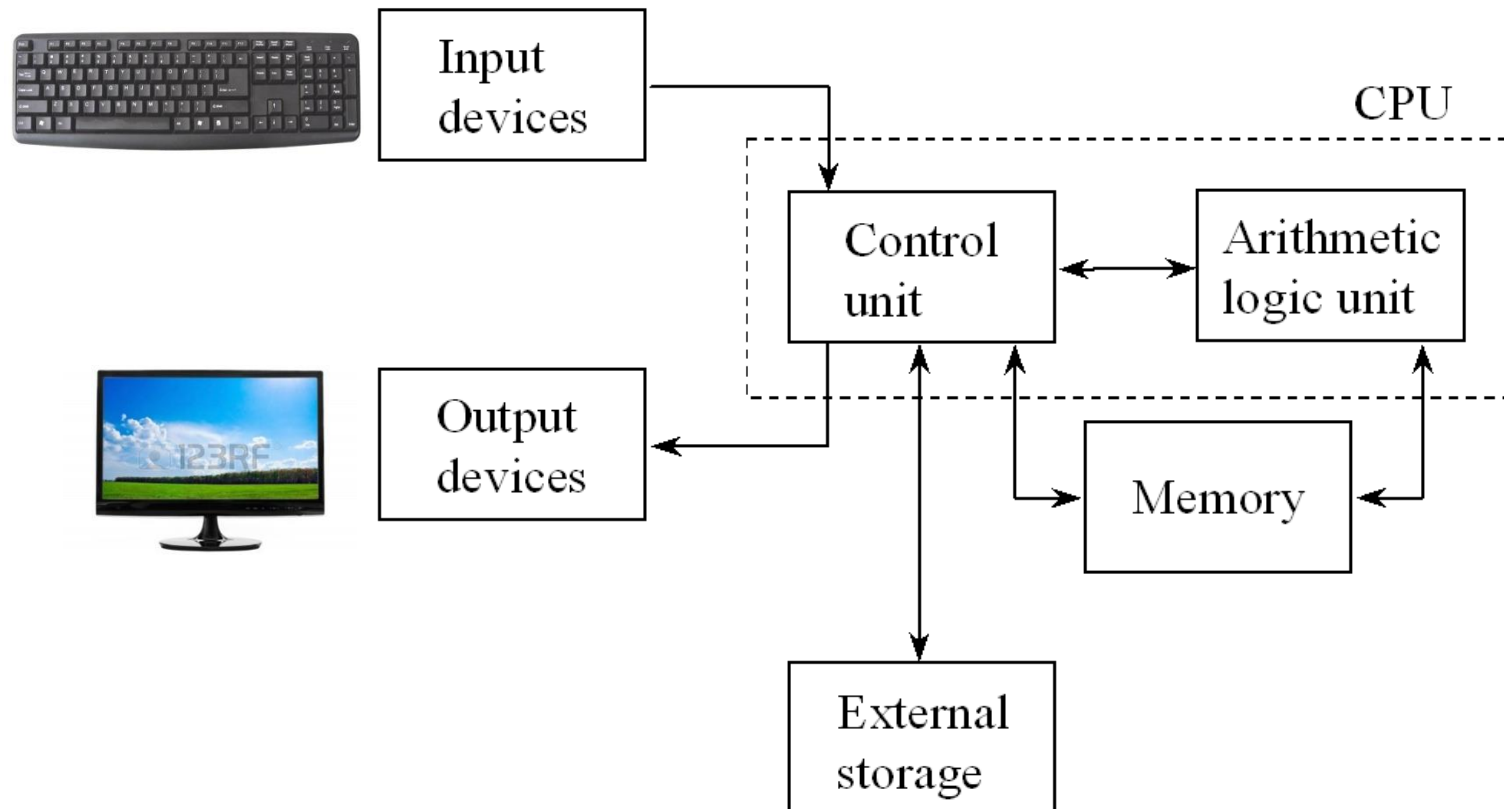
- Various devices comprising a computer
- Keyboard, screen, mouse, disks, memory, CD-ROM, printer, and processing units

- **Software**

- Programs that run on a computer
- Microsoft Windows, Microsoft Office, Internet Explorer



# Conceptual Structure of a Computer System



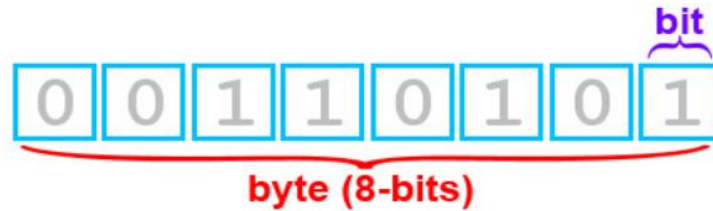


## **CPU (Central Processing Unit)**

- Process and manipulate information stored in memory.
- It can be divided into two units: CU (Control Unit) and ALU (Arithmetic Logic Unit)
- CU coordinates activities of the computer and controls other devices of computer.
- ALU processes arithmetical and logical instructions.



# Bit and Byte



A bit is a single numeric value, either '1' or '0', that encodes a single unit of digital information.

A byte is a sequence of bits; usually eight bits equal one byte.

Byte = 8 bits

KiloByte (KB) = 1,024 Bytes

MegaBytes (MB) = 1,024 KB

GigaByte (GB) = 1,024 MB

TeraByte (TB) = 1,024 GB

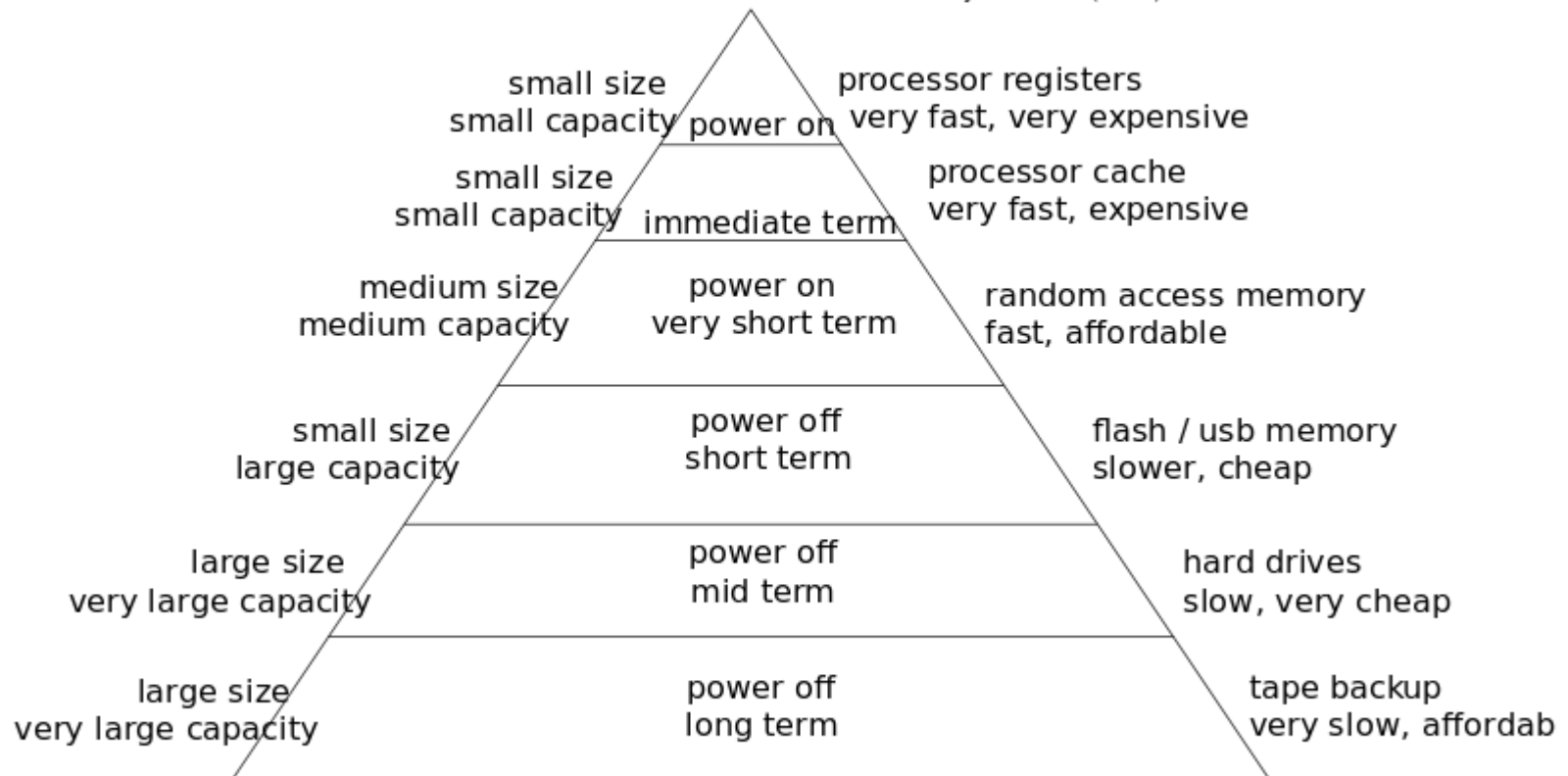
# Memory

- Store information (data + instructions)
- A sequence of memory cells.
- Store, retrieve, update
  - changing the pattern of 0 and 1s in memory cells
  - copying these patterns into some internal registers
- Stored information in memory is **volatile**.



# Computer Memory Hierarchy

by Dan Lash (.com)





# Binary System

- Hardware can only deal with **binary digits**, 0 and 1.
- Must represent all numbers, integers or floating point, positive or negative, by binary digits, called ***bits***.
- Can devise electronic circuits to perform arithmetic operations: add, subtract, multiply and divide, on binary numbers.

# Binary System

- **Decimal** system: made of 10 digits,  $\{0,1,2, \dots, 9\}$

$$41 = 4 \times 10^1 + 1 \times 10^0$$

$$255 = 2 \times 10^2 + 5 \times 10^1 + 5 \times 10^0$$

- **Binary** system: made of two digits,  $\{0,1\}$

$$\begin{aligned} 00101001 &= 0 \times 2^7 + 0 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 \\ &\quad + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 32 + 8 + 1 = 41 \end{aligned}$$

$$11111111 = 255, \text{ largest number with 8} \\ \text{binary digits, } 2^8 - 1$$

## Integer Types

Following table gives you details about standard integer types with its storage sizes and value ranges:

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

## Floating-Point Types

Following table gives you details about standard floating-point types with storage sizes and value ranges and their precision:

Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

# ASCII Table

## American Standard Code for Information Interchange

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32; <b>Space</b>		64	40	100	&#64; <b>@</b>		96	60	140	&#96; <b>`</b>	
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33; <b>!</b>		65	41	101	&#65; <b>A</b>		97	61	141	&#97; <b>a</b>	
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34; <b>"</b>		66	42	102	&#66; <b>B</b>		98	62	142	&#98; <b>b</b>	
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35; <b>#</b>		67	43	103	&#67; <b>C</b>		99	63	143	&#99; <b>c</b>	
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36; <b>\$</b>		68	44	104	&#68; <b>D</b>		100	64	144	&#100; <b>d</b>	
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37; <b>%</b>		69	45	105	&#69; <b>E</b>		101	65	145	&#101; <b>e</b>	
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38; <b>&amp;</b>		70	46	106	&#70; <b>F</b>		102	66	146	&#102; <b>f</b>	
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39; <b>'</b>		71	47	107	&#71; <b>G</b>		103	67	147	&#103; <b>g</b>	
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40; <b>(</b>		72	48	110	&#72; <b>H</b>		104	68	150	&#104; <b>h</b>	
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41; <b>)</b>		73	49	111	&#73; <b>I</b>		105	69	151	&#105; <b>i</b>	
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42; <b>*</b>		74	4A	112	&#74; <b>J</b>		106	6A	152	&#106; <b>j</b>	
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43; <b>+</b>		75	4B	113	&#75; <b>K</b>		107	6B	153	&#107; <b>k</b>	
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44; <b>,</b>		76	4C	114	&#76; <b>L</b>		108	6C	154	&#108; <b>l</b>	
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45; <b>-</b>		77	4D	115	&#77; <b>M</b>		109	6D	155	&#109; <b>m</b>	
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46; <b>.</b>		78	4E	116	&#78; <b>N</b>		110	6E	156	&#110; <b>n</b>	
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47; <b>/</b>		79	4F	117	&#79; <b>O</b>		111	6F	157	&#111; <b>o</b>	
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48; <b>0</b>		80	50	120	&#80; <b>P</b>		112	70	160	&#112; <b>p</b>	
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49; <b>1</b>		81	51	121	&#81; <b>Q</b>		113	71	161	&#113; <b>q</b>	
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50; <b>2</b>		82	52	122	&#82; <b>R</b>		114	72	162	&#114; <b>r</b>	
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51; <b>3</b>		83	53	123	&#83; <b>S</b>		115	73	163	&#115; <b>s</b>	
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52; <b>4</b>		84	54	124	&#84; <b>T</b>		116	74	164	&#116; <b>t</b>	
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53; <b>5</b>		85	55	125	&#85; <b>U</b>		117	75	165	&#117; <b>u</b>	
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54; <b>6</b>		86	56	126	&#86; <b>V</b>		118	76	166	&#118; <b>v</b>	
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55; <b>7</b>		87	57	127	&#87; <b>W</b>		119	77	167	&#119; <b>w</b>	
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56; <b>8</b>		88	58	130	&#88; <b>X</b>		120	78	170	&#120; <b>x</b>	
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57; <b>9</b>		89	59	131	&#89; <b>Y</b>		121	79	171	&#121; <b>y</b>	
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58; <b>:</b>		90	5A	132	&#90; <b>Z</b>		122	7A	172	&#122; <b>z</b>	
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59; <b>;</b>		91	5B	133	&#91; <b>[</b>		123	7B	173	&#123; <b>{</b>	
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60; <b>&lt;</b>		92	5C	134	&#92; <b>\</b>		124	7C	174	&#124; <b> </b>	
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61; <b>=</b>		93	5D	135	&#93; <b>]</b>		125	7D	175	&#125; <b>}</b>	
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62; <b>&gt;</b>		94	5E	136	&#94; <b>^</b>		126	7E	176	&#126; <b>~</b>	
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63; <b>?</b>		95	5F	137	&#95; <b>_</b>		127	7F	177	&#127; <b>DEL</b>	

## History of C

- C
  - Developed by Denis M. Ritchie at AT&T Bell Labs in **1972** as a systems programming language
  - Used to develop UNIX
  - Used to write modern operating systems
  - Hardware independent (portable)
- Standardization
  - Many slight variations of C existed, and were incompatible
  - Committee formed to create a "unambiguous, machine independent" definition
  - Standard created in 1989, updated in 1999

## **Other High-level Languages**

### **– C++**

- Superset of C, and provides object-oriented capabilities

### **– Java**

- Create web pages with dynamic and interactive content

### **– Fortran**

- Used for scientific and engineering applications

### **– Cobol**

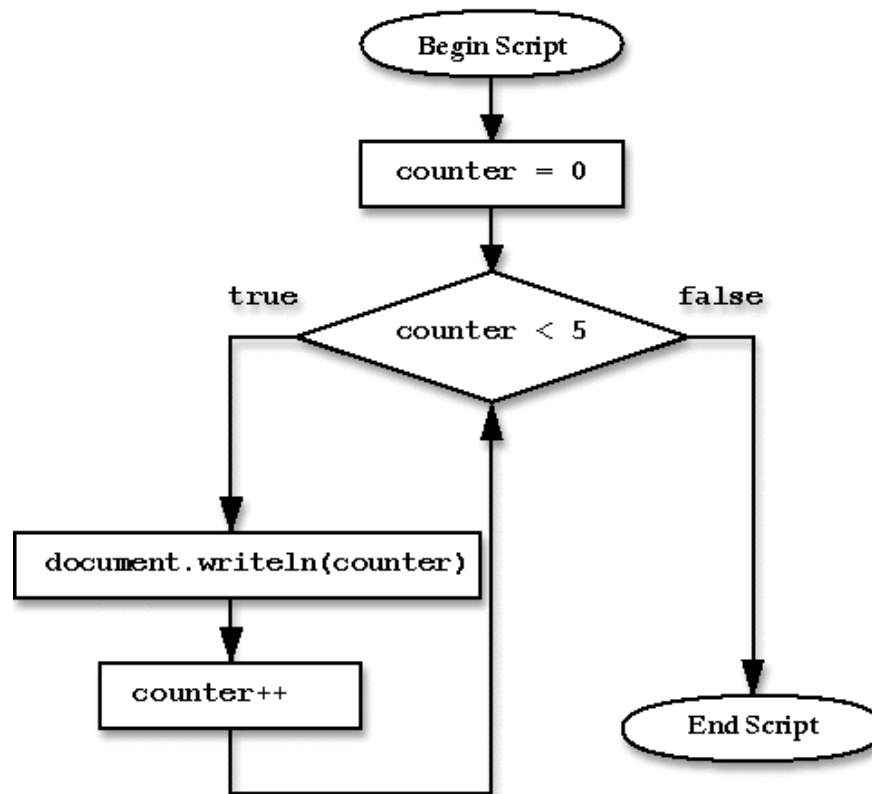
- Used to manipulate large amounts of data

### **– Pascal**

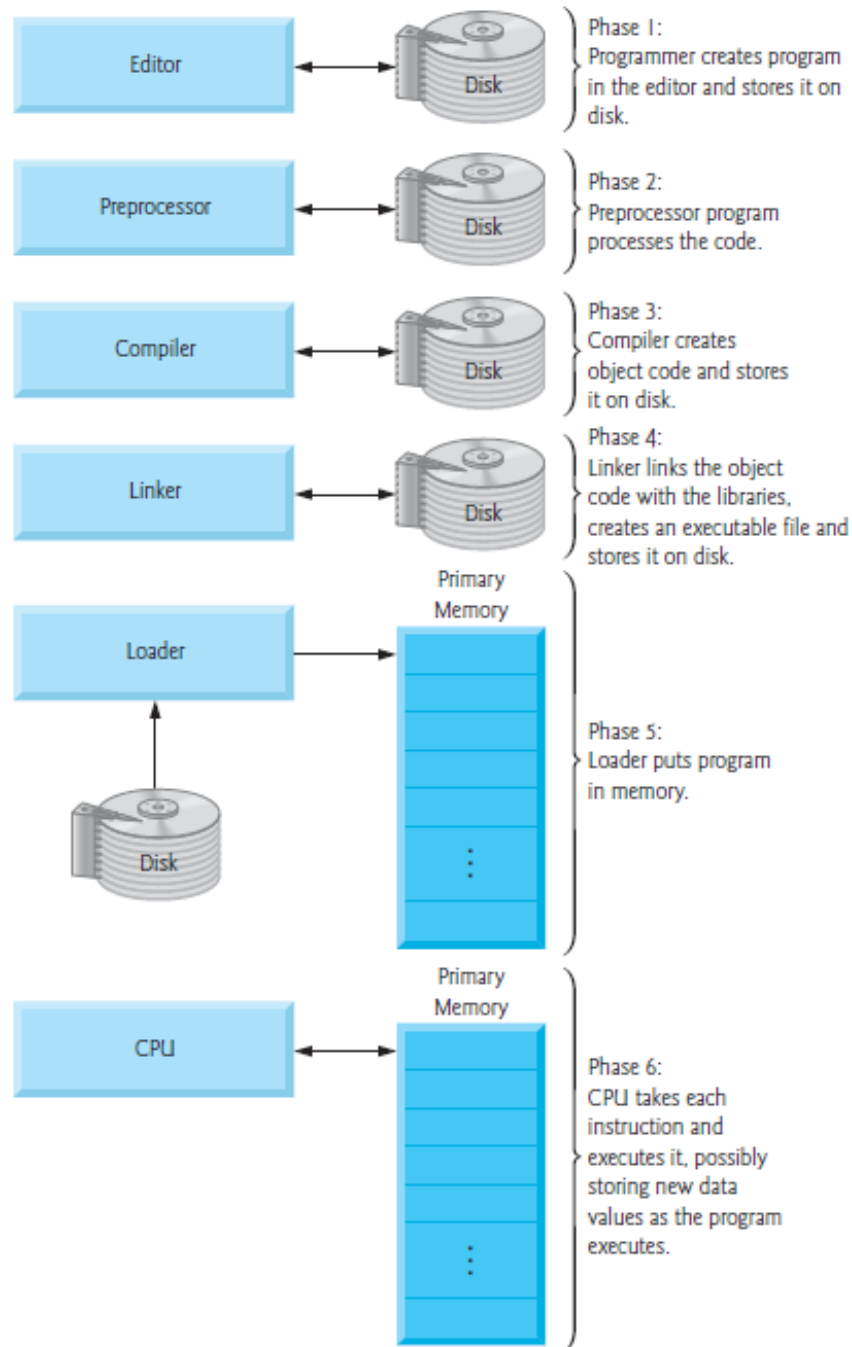
- Intended for academic use

## Flow chart of a program (Algorithm)

In mathematics and computer science, an *algorithm* is a step-by-step procedure for calculations.



## C language development environment





# DevC++ to write C code

**BloodshedSoftware**

Providing Free Software to the internet community

Site Map

Home

Download

Email us

Get Bloodshed CD

Programmers

Free compilers list

Delphi

C and C++

Linux/Unix

Resources

Free URL submission

Add a resource

Dev-C++ 5 (currently beta)

 [Screenshot]

Bloodshed Dev-C++ is a full-featured Integrated Development Environment (IDE) for the C/C++ programming language. It uses Mingw port of GCC (GNU Compiler Collection) as it's compiler. Dev-C++ can also be used in combination with Cygwin or any other GCC based compiler.

Features are :

- Support GCC-based compilers
- Integrated debugging (using GDB)
- Project Manager
- Customizable syntax highlighting editor
- Class Browser
- Code Completion
- Function listing
- Profiling support
- Quickly create Windows, console, static libraries and DLLs
- Support of templates for creating your own project types
- Makefile creation
- Edit and compile Resource files
- Tool Manager
- Print support
- Find and replace facilities
- CVS support

**Source code :** Delphi 6 Source code of Dev-C++ is available for free under the GNU General Public License (GPL)

**Authors :** Colin Laplace, Mike Berg, Hongli Lai : Development  
Mingw compiler: Mumit Khan, Jan Jaap van der Heidjen, Colin Hendrix and GNU coders.

**System :** Windows 95/98/NT/2000/XP

**Status :** Free Software (under the GNU General Public License)

**Size :** 13.5 Mb

**Downloads :** [Go to Download Page](#)

[Dev-C++ resources page \(libraries, sources, updates...\)](#)

Dev-C++ 4

 [Screenshot]

Dev-C++ is a full-featured integrated development environment (IDE), which is able to create Windows or console-based C/C++ programs using the Mingw compiler system (version MSVCRT 2.95.2-1 included with this package), or the Cygwin compiler. It can also handle the Insight Debugger, which you can also download here. - C and C++ compiler for Win32 (Mingw)

- Debugger (GDB or Insight)

Get Bloodshed CD



Get it on Bloodshed CD

You can get all the software available on this site on a CD. This is the list of all software included on ONE CD:

Dev-C++ 5 beta  
Dev-C++ 4 CD version  
Dev-C++ 4.01 update  
Dev-C++ for Linux  
QuickInstall 2.0  
Avi Creator 1.0  
Dev-C++ 4 sources  
Dev-Pascal 1.9 sources  
Multibox 4.0  
Fast Cleaner 1.0  
LaserWar  
LaserWar sources for Delphi  
Calc-By-Step 1.0  
Dev-C++ Packages  
+ Goodies !

Copyright Bloodshed Software

If you find any broken links in this site, please [send me an email](#).



<http://www.bloodshed.net/devcpp.html>

---

```
1  /* Fig. 2.1: fig02_01.c
2     A first program in C */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main( void )
7  {
8     printf( "Welcome to C!\n" );
9
10     return 0; /* indicate that program ended successfully */
11 } /* end function main */
```

Welcome to C!

**Fig. 2.1** | A first program in C.

---

```
1  /* Fig. 2.3: fig02_03.c
2     Printing on one line with two printf statements */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main( void )
7  {
8     printf( "Welcome " );
9     printf( "to C!\n" );
10
11     return 0; /* indicate that program ended successfully */
12 } /* end function main */
```

---

**Fig. 2.3** | Printing on one line with two printf statements. (Part 1 of 2.)

Welcome to C!

---

```
1  /* Fig. 2.4: fig02_04.c
2     Printing multiple lines with a single printf */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main( void )
7  {
8     printf( "Welcome\nto\nC!\n" );
9
10     return 0; /* indicate that program ended successfully */
11 } /* end function main */
```

```
Welcome
to
C!
```

**Fig. 2.4** | Printing multiple lines with a single printf.

```
1  /* Fig. 2.5: fig02_05.c
2     Addition program */
3  #include <stdio.h>
4
5  /* function main begins program execution */
6  int main( void )
7  {
8     int integer1; /* first number to be input by user */
9     int integer2; /* second number to be input by user */
10    int sum; /* variable in which sum will be stored */
11
12    printf( "Enter first integer\n" ); /* prompt */
13    scanf( "%d", &integer1 ); /* read an integer */
14
15    printf( "Enter second integer\n" ); /* prompt */
16    scanf( "%d", &integer2 ); /* read an integer */
17
18    sum = integer1 + integer2; /* assign total to sum */
19
20    printf( "Sum is %d\n", sum ); /* print sum */
21
22    return 0; /* indicate that program ended successfully */
23 } /* end function main */
```

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

**Fig. 2.5** | Addition program (Part 2 of 2)

Variable names such as `integer1`, `integer2` and `sum` actually correspond to locations in the computer's memory. Every variable has a name, a **type** and a **value**.

In the addition program of Fig. 2.5, when the statement (line 13)

```
scanf( "%d", &integer1 ); /* read an integer */
```

is executed, the value typed by the user is placed into a memory location to which the name `integer1` has been assigned. Suppose the user enters the number 45 as the value for `integer1`. The computer will place 45 into location `integer1` as shown in Fig. 2.6.

---

`integer1`



A light blue rectangular box with a slight 3D effect, containing the number 45.

---

**Fig. 2.6** | Memory location showing the name and value of a variable.

Whenever a value is placed in a memory location, the value replaces the previous value in that location; thus, placing a new value into a memory location is said to be **destructive**.

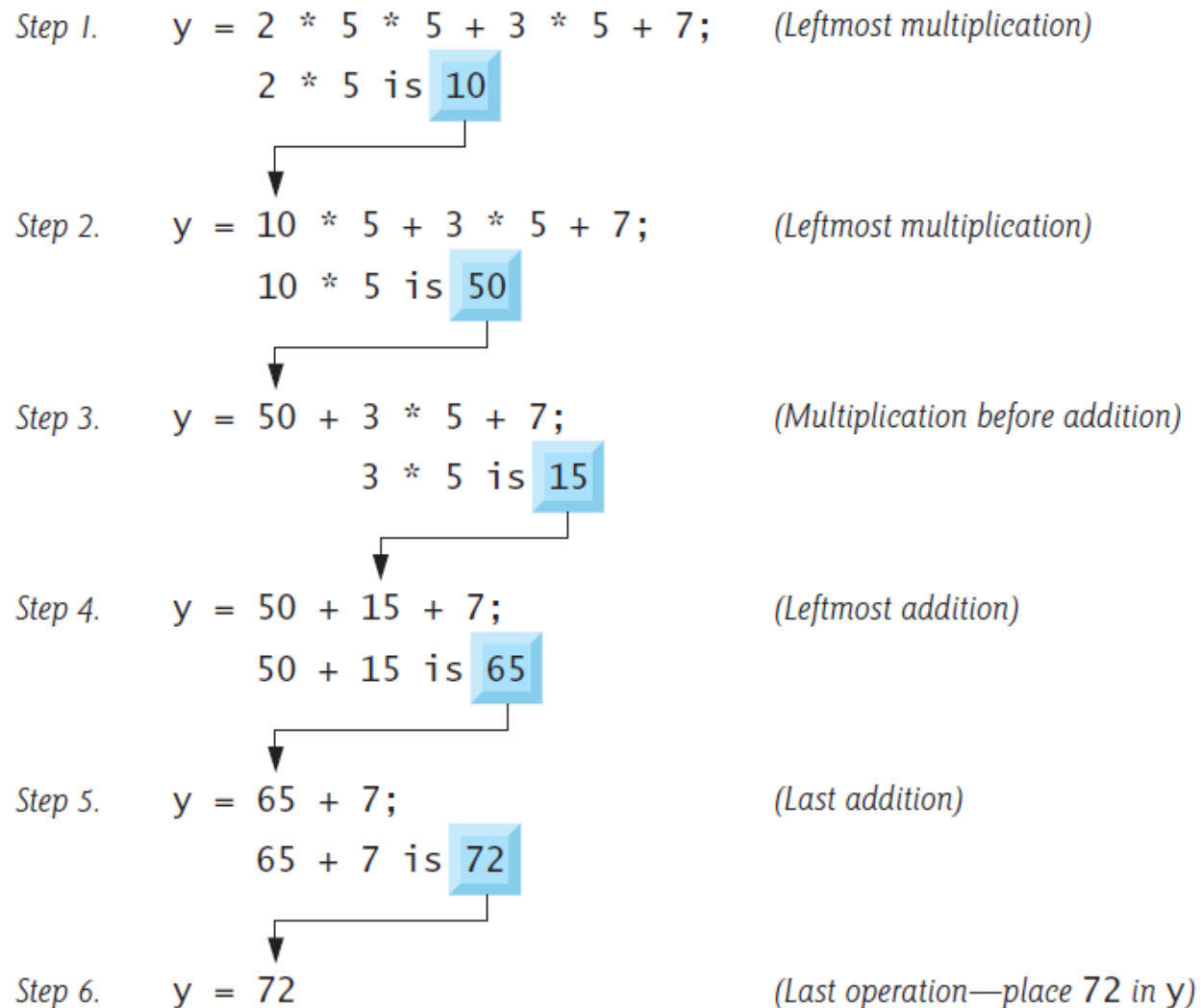
C operation	Arithmetic operator	Algebraic expression	C expression
Addition	+	$f + 7$	<code>f + 7</code>
Subtraction	-	$p - c$	<code>p - c</code>
Multiplication	*	$bm$	<code>b * m</code>
Division	/	$x / y$ or $\frac{x}{y}$ or $x \div y$	<code>x / y</code>
Remainder	%	$r \bmod s$	<code>r % s</code>

```
printf( "Welcome to \\\%d", (3/2) );
```

Output is : 1

Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses “on the same level” (i.e., not nested), they’re evaluated left to right.
*	Multiplication	Evaluated second. If there are several, they’re evaluated left to right.
/	Division	
%	Remainder	
+	Addition	Evaluated last. If there are several, they’re evaluated left to right.
-	Subtraction	





**Fig. 2.11** | Order in which a second-degree polynomial is evaluated.

Algebraic equality or relational operator	C equality or relational operator	Example of C condition	Meaning of C condition
<i>Equality operators</i>			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y
<i>Relational operators</i>			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y

**Fig. 2.12** | Equality and relational operators.

! exclamation mark

```

1  /* Fig. 2.13: fig02_13.c
2     Using if statements, relational
3     operators, and equality operators */
4  #include <stdio.h>
5
6  /* function main begins program execution */
7  int main( void )
8  {
9     int num1; /* first number to be read from user */
10    int num2; /* second number to be read from user */
11
12    printf( "Enter two integers, and I will tell you\n" );
13    printf( "the relationships they satisfy: " );
14
15    scanf( "%d%d", &num1, &num2 ); /* read two integers */
16
17    if ( num1 == num2 ) {
18        printf( "%d is equal to %d\n", num1, num2 );
19    } /* end if */
20
21    if ( num1 != num2 ) {
22        printf( "%d is not equal to %d\n", num1, num2 );
23    } /* end if */
24
25    if ( num1 < num2 ) {
26        printf( "%d is less than %d\n", num1, num2 );
27    } /* end if */
28
29    if ( num1 > num2 ) {
30        printf( "%d is greater than %d\n", num1, num2 );
31    } /* end if */
32
33    if ( num1 <= num2 ) {
34        printf( "%d is less than or equal to %d\n", num1, num2 );
35    } /* end if */
36
37    if ( num1 >= num2 ) {
38        printf( "%d is greater than or equal to %d\n", num1, num2 );
39    } /* end if */
40
41    return 0; /* indicate that program ended successfully */
42 } /* end function main */

```

```

Enter two integers, and I will tell you
the relationships they satisfy: 3 7
3 is not equal to 7
3 is less than 7
3 is less than or equal to 7

```

Escape sequence	Description
<code>\n</code>	Newline. Position the cursor at the beginning of the next line.
<code>\t</code>	Horizontal tab. Move the cursor to the next tab stop.
<code>\a</code>	Alert. Sound the system bell.
<code>\\</code>	Backslash. Insert a backslash character in a string.
<code>\"</code>	Double quote. Insert a double-quote character in a string.

**Fig. 2.2** | Some common escape sequences .

## Keywords

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

*Keywords added in C99*

\_Bool \_Complex \_Imaginary inline restrict

**Fig. 2.15** | C's keywords.

## **The C Standard Library**

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

# Use search engines while studying

Google

Web Görseller Videolar Uygulamalar Daha fazla ▾ Arama araçları


Yaklaşık 1.910.000.000 sonuç bulundu (0,19 saniye)

YouTube Twitter eBay Flickr

[C \(programming language\) - Wikipedia, the free encyclopedia](#)  
[en.wikipedia.org/.../C\\_\(programming\\_languag...](#) ▾ Bu sayfanın çevirisini yap  
C++ and Objective-C started as compilers that generated **C code**; C++ is currently nearly a superset of C, while Objective-C is a strict superset of C. Before there ...  
[Dennis Ritchie](#) - [C11](#) - [List of C-based programming ...](#) - [Compatibility of C and C++](#)

[Operators in C and C++ - Wikipedia, the free encyclopedia](#)  
[en.wikipedia.org/.../Operators\\_in\\_C\\_and\\_C%2...](#) ▾ Bu sayfanın çevirisini yap  
This is a list of operators in the C and C++ programming languages: ..... from June 2012  
- Use dmy dates from January 2012 - Articles with example **C++ code** ...

[c code ile ilgili görseller](#) - Görseller hakkında kötüye kullanım bildirin



[C programming.com - Learn C and C++ Programming ...](#)  
[www.cprogramming.com/](#) ▾ Bu sayfanın çevirisini yap  
A website designed to help you learn C or C++. Understandable C and C++ programming tutorials, compiler reviews, source **code**, tips and tricks.  
[C Tutorial](#) - [Learn C](#) - [Jumping into C++](#), by Alex Allain - [C++ Tutorial](#) - [Tutorials](#)

[linux - What is ":-!" in C code? - Stack Overflow](#)  
[stackoverflow.com/questions/.../what-is-in-c-co...](#) ▾ Bu sayfanın çevirisini yap  
10 Şub 2012 - \* Force a compilation error if condition is true, but also produce a ... This is, in effect, a way to check whether the expression e can be evaluated to ...

[HTML Codes - Special Characters - ASCII Table - Web Design](#)  
[webdesign.about.com/library/bl\\_htmlcodes.htm](#) ▾ Bu sayfanın çevirisini yap  
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z. Character, Friendly **Code**, Numerical **Code**, Hex **Code**, Description: A, A, &#65;, &#x41;, Capital A.  
[HTML Codes for Punctuation ...](#) - [ASCII HTML Codes](#) - [Pronunciation Codes](#)

# Useful links

<http://www.programmingsimplified.com/c-program-examples>

<http://www.tutorialspoint.com/cprogramming/>

<http://www.wikihow.com/Learn-to-Program-in-C>

<http://www.programiz.com/c-programming/examples>

<http://www.lynda.com/C-tutorials/C-Essential-Training/164457-2.html>



# Homework

- Install BloodshedC++ compiler to your computer
- Write a C program that prints your name, surname and mail address to the screen and upload the .c file (Source file) to LMS (Assignment-1) until next week.
- Cover the related material from reference books.