CEng 230 Introduction to C Programming

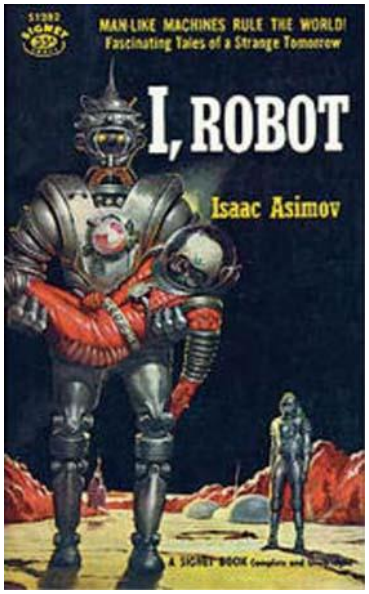
Seyyit Alper SERT

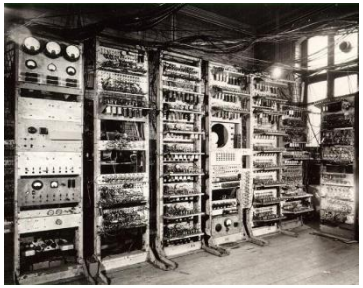
Department of Computer Engineering
2017-2018 Fall

Web Pages
Official Course Page: ceng230.ceng.metu.edu.tr
Learning Management System (LMS): odtuclass.metu.edu.tr
Contact: 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**

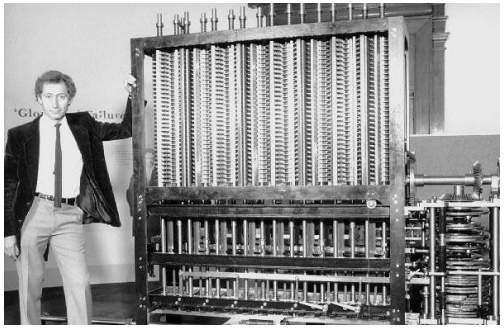
– Device capable of performing computations and making logical decisions
– Computers process data under the control of sets of instructions called *computer programs*

- **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

Input devices

CPU

Control unit

Arithmetic logic unit

Output devices

Memory

External storage

CPU (Central Processing Unit)

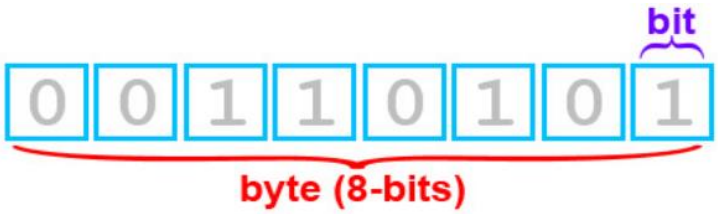
- Process and manipulate information stored in memory.

- 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.

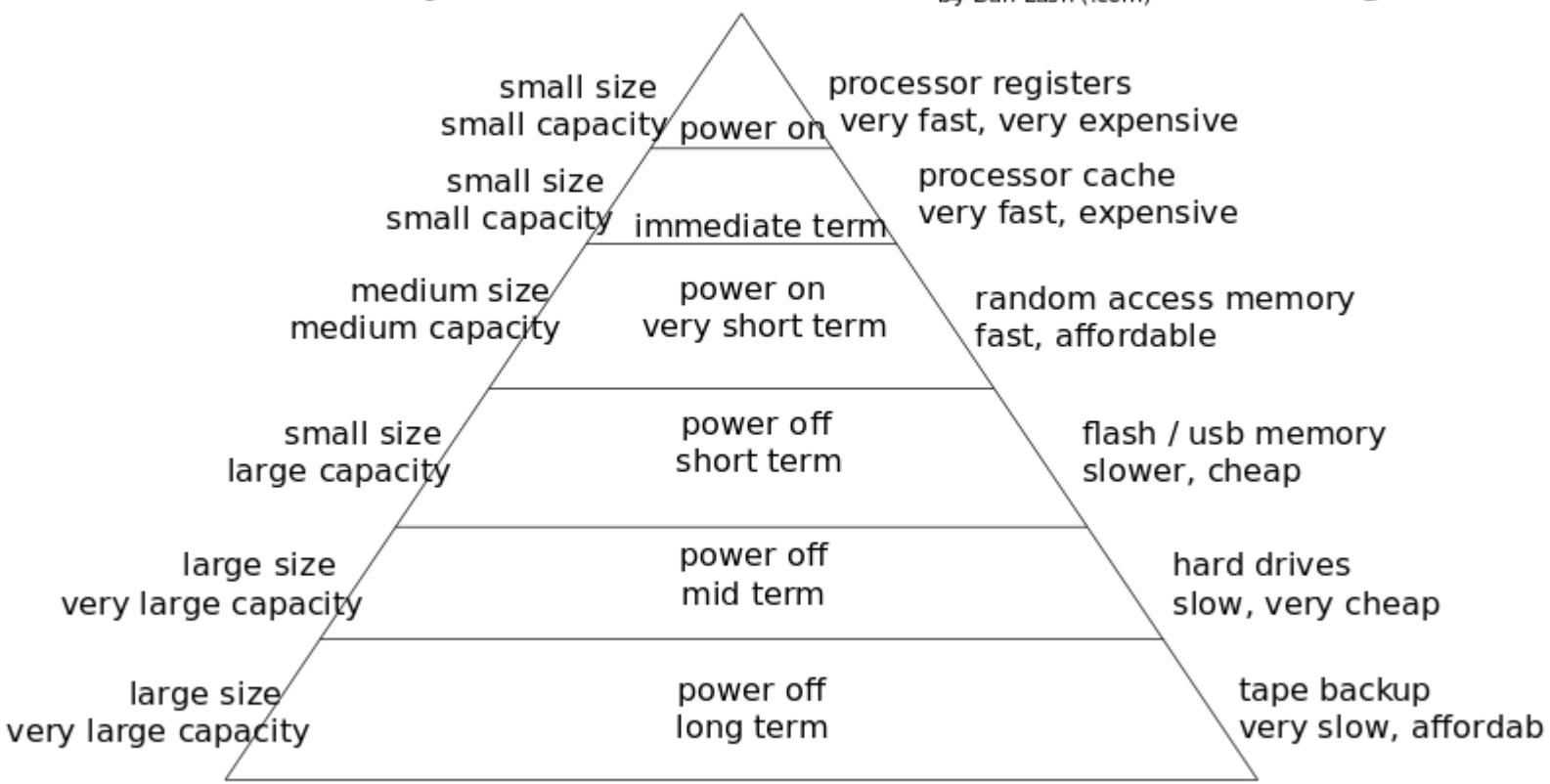
- Store, retrieve, update

– 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



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, . . . , 9}

$$41 = 4\times10^1 + 1\times10^0$$

$$255 = 2\times10^2 + 5\times10^1 + 5\times10^0$$

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

$00101001 = 0\times2^7 + 0\times2^6 + 1\times2^5 + 0\times2^4$
$+1\times2^3 + 0\times2^2 + 0\times2^1 + 1\times2^0$

$= 32 + 8 + 1 = 41$

11111111 = 255, largest number with 8 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	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

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

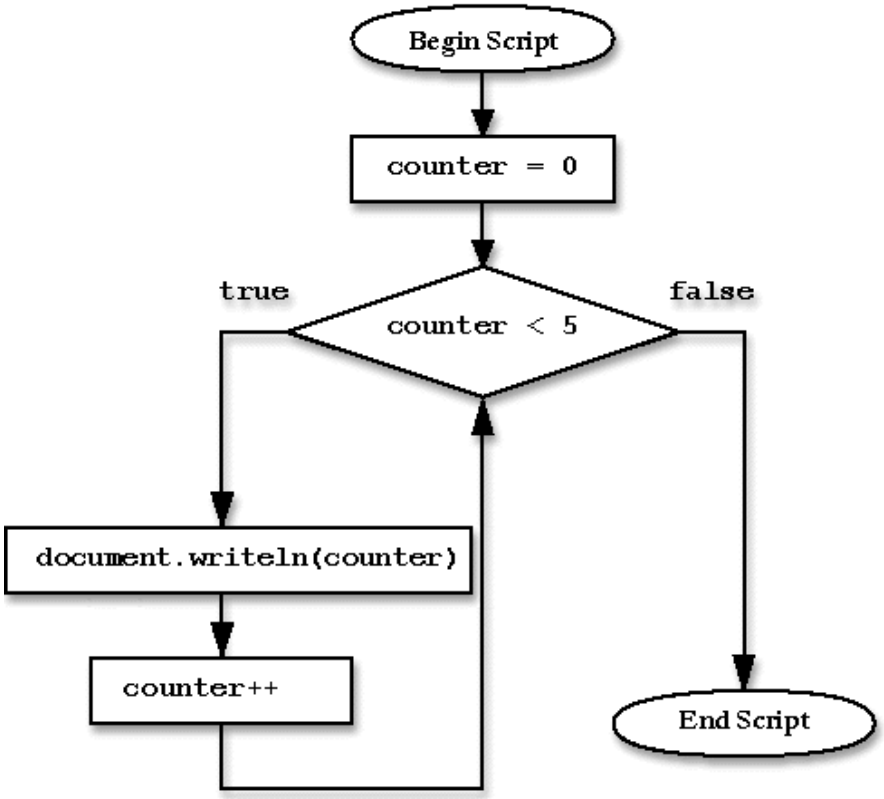
- Used to manipulate large amounts of data

– Pascal

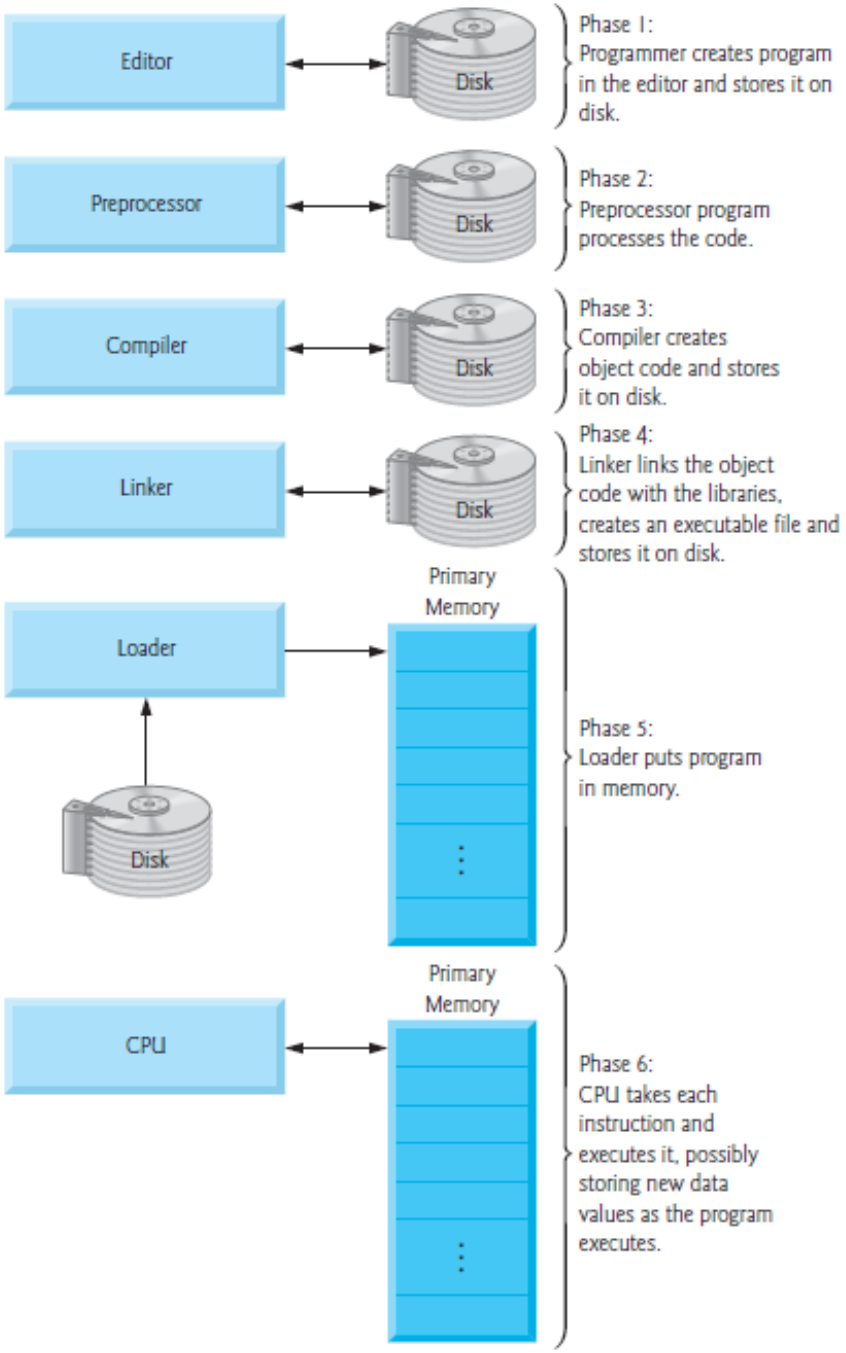
- Intended for academic use

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

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

```c
/* Fig. 2.1: fig02_01.c
   A first program in C */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   printf( "Welcome to C!\n" );

   return 0; /* indicate that program ended successfully */
} /* end function main */
```

```
Welcome to C!
```

Fig. 2.1 | A first program in C.

```c
/* Fig. 2.3: fig02_03.c
   Printing on one line with two printf statements */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   printf( "Welcome " );
   printf( "to C!\n" );

   return 0; /* indicate that program ended successfully */
} /* end function main */
```

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

```
Welcome to C!
```

```c
/* Fig. 2.4: fig02_04.c
   Printing multiple lines with a single printf */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   printf( "Welcome\nto\nC!\n" );

   return 0; /* indicate that program ended successfully */
} /* end function main */
```

```
Welcome
to
C!
```

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

```c
/* Fig. 2.5: fig02_05.c
   Addition program */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   int integer1; /* first number to be input by user  */
   int integer2; /* second number to be input by user */
   int sum; /* variable in which sum will be stored */

   printf( "Enter first integer\n" ); /* prompt */
   scanf( "%d", &integer1 ); /* read an integer */

   printf( "Enter second integer\n" ); /* prompt */
   scanf( "%d", &integer2 ); /* read an integer */

   sum = integer1 + integer2; /* assign total to sum */

   printf( "Sum is %d\n", sum ); /* print sum */

   return 0; /* indicate that program ended successfully */
} /* 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.

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	+	$f + 7$	`f + 7`
Subtraction	–	$p - c$	`p - c`
Multiplication	*	bm	`b * m`
Division	/	x / y or $\frac{x}{y}$ or $x \div y$	`x / y`
Remainder	%	$r \bmod s$	`r % s`

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	

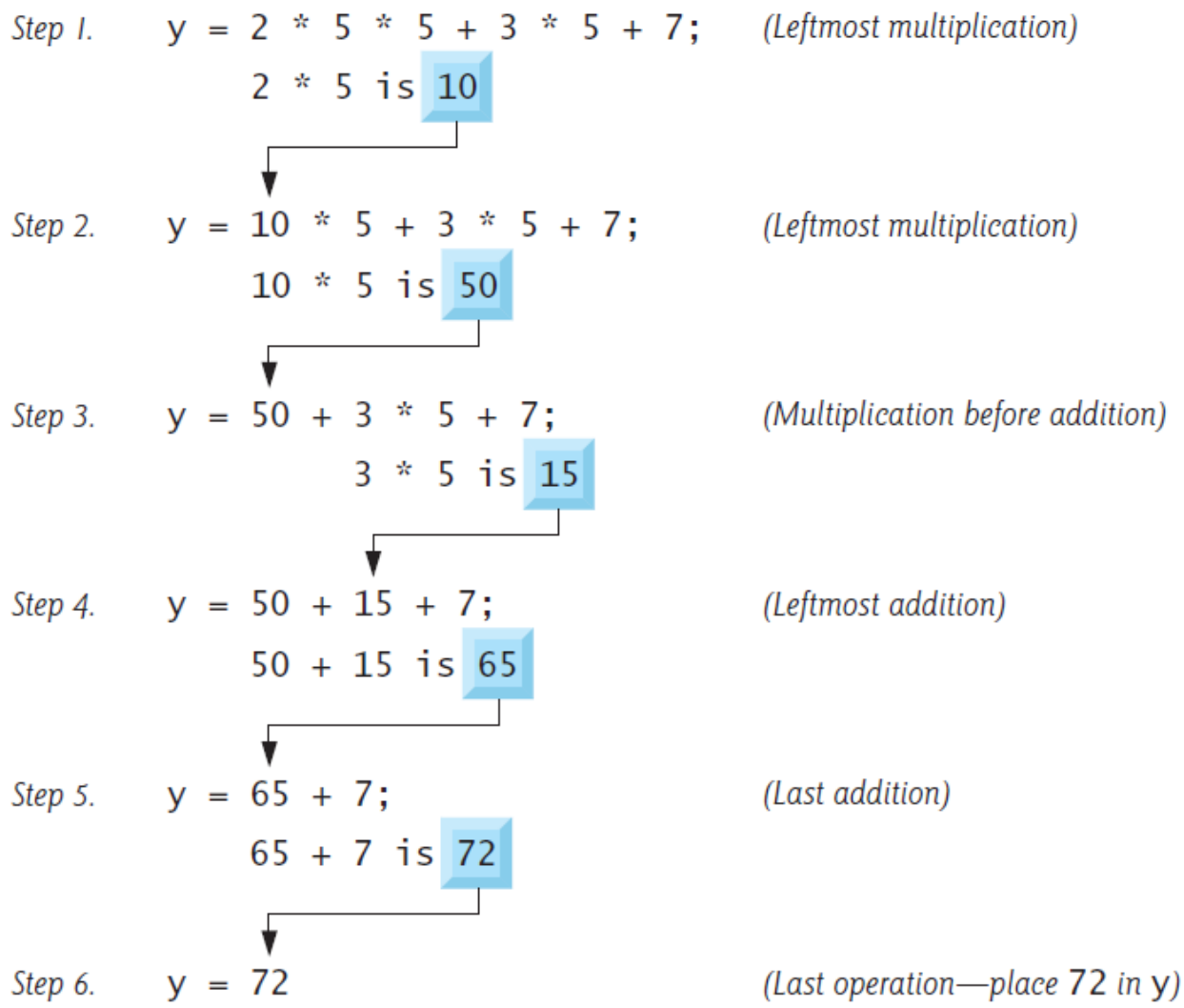


Fig. 2.11 | Order in which a second-degree polynomial is evaluated.

Algebraic equality or relational operator	C equality or relational operator	Example of C condition	Meaning of C condition
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y
Relational operators			
>	>	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

```c
/* Fig. 2.13: fig02_13.c
   Using if statements, relational
   operators, and equality operators */
#include <stdio.h>

/* function main begins program execution */
int main( void )
{
   int num1; /* first number to be read from user  */
   int num2; /* second number to be read from user */

   printf( "Enter two integers, and I will tell you\n" );
   printf( "the relationships they satisfy: " );

   scanf( "%d%d", &num1, &num2 ); /* read two integers */

   if ( num1 == num2 ) {
      printf( "%d is equal to %d\n", num1, num2 );
   } /* end if */

   if ( num1 != num2 ) {
      printf( "%d is not equal to %d\n", num1, num2 );
   } /* end if */

   if ( num1 < num2 ) {
      printf( "%d is less than %d\n", num1, num2 );
   } /* end if */

   if ( num1 > num2 ) {
      printf( "%d is greater than %d\n", num1, num2 );
   } /* end if */

   if ( num1 <= num2 ) {
      printf( "%d is less than or equal to %d\n", num1, num2 );
   } /* end if */

   if ( num1 >= num2 ) {
      printf( "%d is greater than or equal to %d\n", num1, num2 );
   } /* end if */

   return 0; /* indicate that program ended successfully */
} /* 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
`\n`	Newline. Position the cursor at the beginning of the next line.
`\t`	Horizontal tab. Move the cursor to the next tab stop.
`\a`	Alert. Sound the system bell.
`\\`	Backslash. Insert a backslash character in a string.
`\"`	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

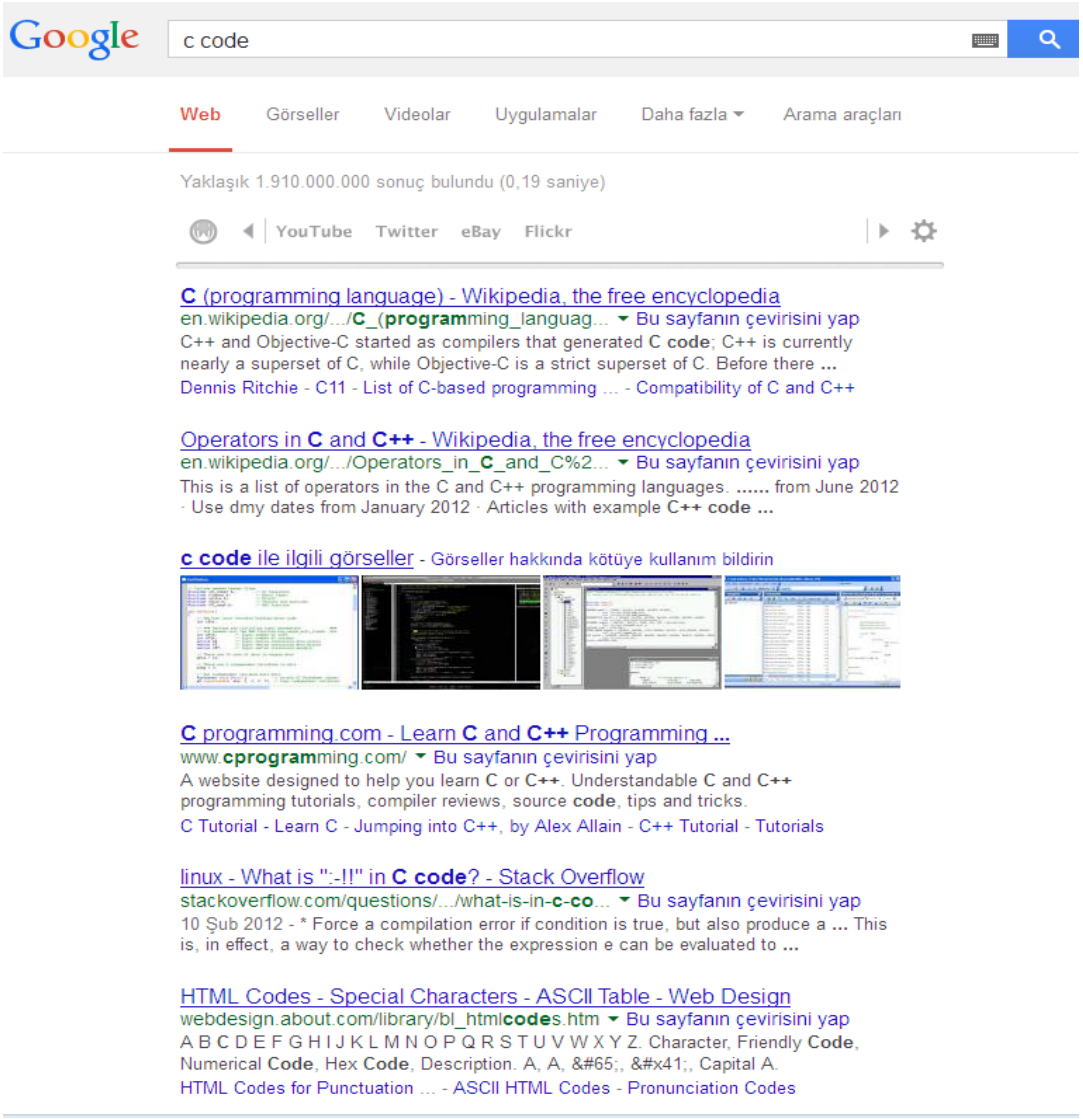
• C programs consist of pieces/modules called functions

– 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



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.