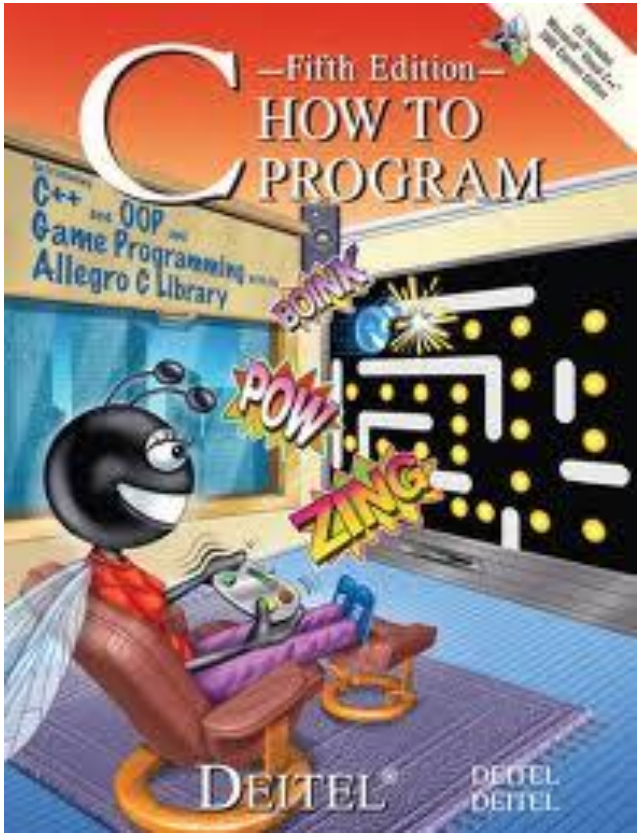


C How to Program,

H. M. Deitel and P. J. Deitel,
Prentice Hall, 5th edition
(3rd edition or above is also OK).

Introduction to C Programming

Dr. Hasan Demirel

Programming Languages

- There are three types of programming Languages

1) Machine Languages (machine codes):

- Strings of 1s and 0s.
- Only unterstood by integrated circuits, such as microprocessors.

Example:
```
10100010
01011011
10101010
```

2) Assembly Languages:

- English-like abbreviations representing elementary computer operations.
- translated to machine code by using assemblers.

Example:
```
MOV AL,3BH
ADD AL, AH
SUB AL,AH
MOV [SI]
```

Programming Languages

- There are three types of programming Languages

1) Machine Language
2) Assembly Languages

3) **High-level Languages:**
   - Codes similar to everyday English
   - Use mathematical notations
   - translated to machine code by using compilers.
   - C, C++, PASCAL, FORTRAN, BASIC are high-level languages.

Example:

```
c=a+b;
if(a<b)
   printf("a is less than b\n");
else
   printf("a is NOT less than b\n");
```

Structured programming

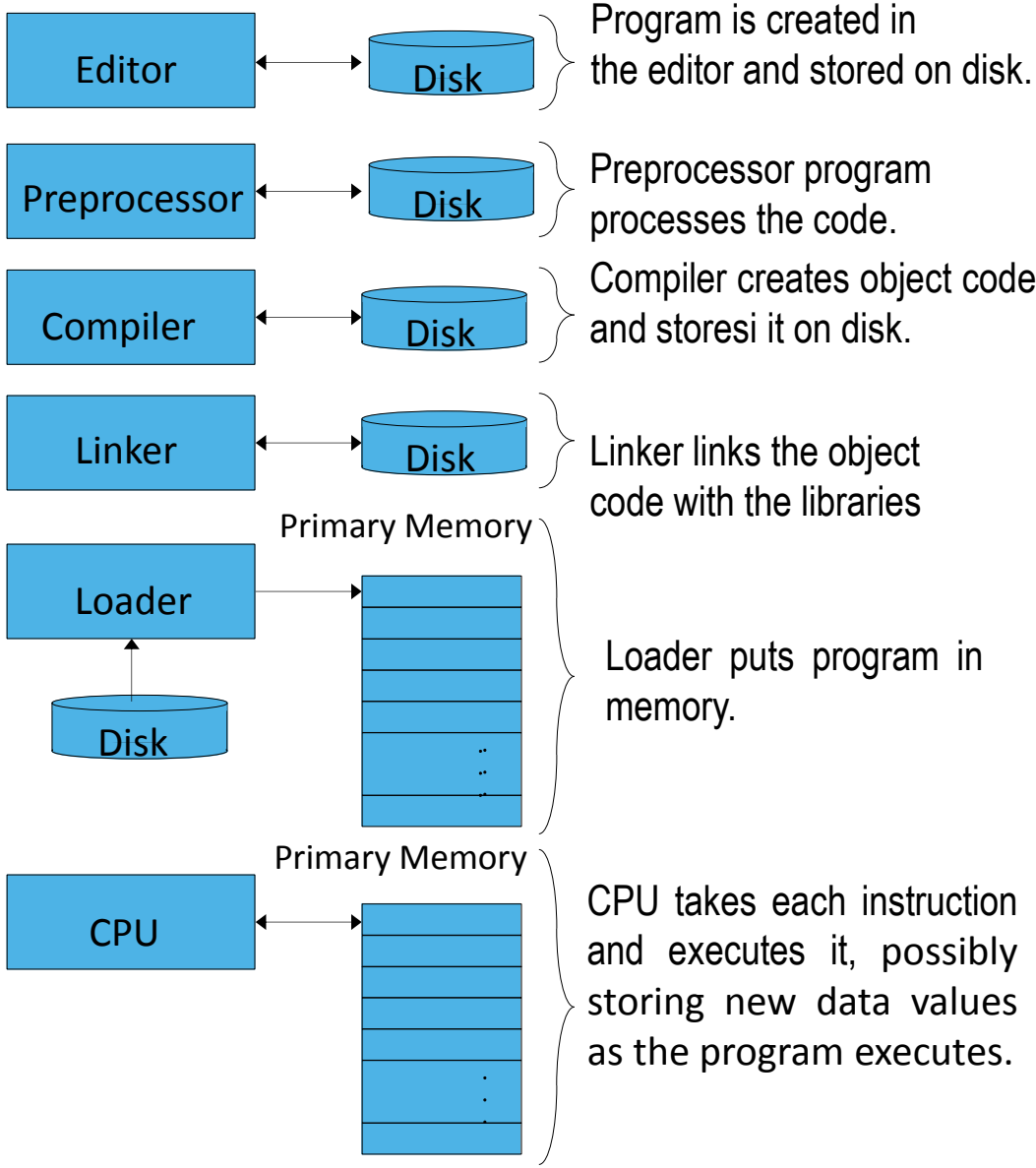
- **Disciplined approach to writing programs**
  - Using flowcharts (graphical representation)
  - Using pseudocodes or step by step algorithms.
- **Clear, easy to test and debug and easy to modify**
  - Using functions for efficient programming.

Multitasking

Specifying that many activities run in parallel.

Basics of a Typical C Program Development Process

- **Phases of C Programs:**
  1. **Edit**
  2. **Preprocess**
  3. **Compile**
  4. **Link**
  5. **Load**
  6. **Execute**



Simple C Program:

- The following program displays `"Hello World"` on the computer screen (monitor).

```
/* This is our first program in C Language */
#include <stdio.h>
int main()
{
   printf("Hello World\n");
return 0;
}
```

- The program output

```
Hello World
```

Simple C Program:

```
/* This is our first program in C Language */
#include <stdio.h>
int main()
{
   printf("Hello World\n");
return 0;
}
```

Comments:

- Text surrounded by **/*** and ***/** is ignored by computer.
- Used to describe program.

#include <stdio.h>

Preprocessor directive:

- Tells computer to load contents of a header file **<stdio.h>**,
- which includes standard input/output functions.
- For example **printf()** is one of the standard input/output functions.

Simple C Program:

```
/* This is our first program in C Language */
#include <stdio.h>
int main()
{
   printf("Hello World\n");
return 0;
}
```

`int main()`

- C programs contain one or more functions,
- One of the functions must be **`main().`**
- Parenthesis used to indicate a function
- **`int`** means that **`main`** "returns" an integer value
- Braces **`({`** and **`})`** indicate a block
- The bodies of all functions must be contained in braces.

Simple C Program:

```
/* This is our first program in C Language */
#include <stdio.h>
int main()
{
   printf("Hello World\n");
return 0;
}
```

`printf("Hello World\n");`

- **`printf()`** function print the string of characters within quotes **`(" ")`**
- All statements must end with a semicolon **`(;)`**
- **`\n`** is the newline character.

`return 0;`

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

`Right brace }`

- Indicates end of main has been reached.

Simple C Program:

- **Example 1:** Write a C program which displays your name and surname in two consecutive lines .

- **Example 2:** Write a C program which displays the following lines.

```
Today
is a
nice

day
```

C Program: Addition of two integer numbers

```
/* This program adds two integer numbers */
#include <stdio.h>
int main()
{
   int a, b, sum;          /* variable declarations */
   printf("Enter first integer\n"); /* prompt the user */
   scanf( "%d", &a);               /* read first integer */
   printf("Enter second integer\n"); /* prompt the user */
   scanf( "%d", &b);               /* read second integer */
   sum = a + b;            /* calculate the sum */
   printf( "Sum = %d\n", sum );    /* print the calculated sum*/
return 0; /* indicate that program ended successfully */
}
```

```
Enter first integer
15
Enter second integer
26
Sum = 41
```

Program Output

C Program: Addition of two integer numbers

```
int a, b, sum;
```

- Declaration of variables
  - Variables: locations in memory where a value can be stored
- `int` means the variables can hold integer numbers (-1, 3, 0, 47)
- Variable names (identifiers)
  - `a, b, sum;`
  - Identifiers: consist of letters, digits (cannot begin with a digit) and underscores(_). They are Case sensitive
- Declarations appear before executable statements
  - If an executable statement references and undeclared variable it will produce a syntax (compiler) error.

C Program: Addition of two integer numbers

```
scanf( "%d", &a );
```

- Obtains(reads/inputs) a value from the user
  - **scanf** uses standard input (usually keyboard)

- This **scanf** statement has two arguments
  %d - indicates data should be a decimal integer
  &a – location (address) in memory to store variable **a.**
  & is confusing in beginning – for now, just remember to include it with the variable name in **scanf** statements.

- When executing the program the user responds to the **scanf** statement by typing in a number, then pressing the *enter* (return) key.

C Program: Addition of two integer numbers

= (assignment operator)

- Assigns a value to a variable
- Is a binary operator (has two operands)

```
sum = a + b;
```

sum gets a + b;

- Variable receiving value on left

```
printf( "Sum is %d\n", sum );
```

- Similar to **scanf**
  - **%d** means decimal integer will be printed
  - **sum** specifies what integer will be printed
- Calculations can be performed inside **printf** statements

```
printf( "Sum is %d\n", a + b );
```

C Program: Addition of two integer numbers

- **Example 3:** Write a C program which calculates and displays the addition of integers 7, 8 and 14..

- **Example 4:** Write a C program which asks the user to enter 3 integer numbers and outputs the sum of these three numbers.

Arithmetic Operations in C

- **Arithmetic Calculations:**
  - Use `*` for multiplication and `/` for division
  - Integer division truncates remainder
    `7 / 5` evaluates to `1`
  - Modulus operator(%) returns the remainder of modular division.
    `7 % 5` evaluates to `2`

- **Operator precedence:**
  - Some arithmetic operators act before others (i.e., multiplication before addition)
  - Use parenthesis when needed

  - Example: Find the average of three variables a, b and c
    Do not use: `a + b + c / 3`
    Use: `(a + b + c ) / 3`

Arithmetic Operations in C

- **Arithmetic operators:**

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	`x / y`
Modulus	%	$r \bmod s$	`r % s`

Arithmetic Operations in C

- **Rules of Operator Presidence:**

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 are evaluated left to right.
`*`, `/`, or `%`	**Multiplication,Division, Modulus**	Evaluated second. If there are several, they are evaluated left to right.
`+` or `-`	**Addition Subtraction**	Evaluated last. If there are several, they are evaluated left to right.

Decision Making: Equality and Relational Operators

- **Executable statements**
  - Perform actions (calculations, input/output of data)
  - Perform decisions
    - May want to print `"pass"` or `"fail"` given the value of a test grade
- **if control structure**
  - Simple version in this section, more detail later
  - If a condition is `true`, then the body of the `if` statement executed
    - `0` is `false`, non-zero is `true`
  - Control always resumes after the `if` structure
- **Keywords**
  - Special words reserved for C
  - Cannot be used as identifiers or variable names

Decision Making: Equality and Relational Operators

Standard algebraic equality operator 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`
not =	!=	`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`

Decision Making: Equality and Relational Operators

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

Decision Making: Equality and Relational Operators

```
1  /* Fig. 2.13: fig02 13.c
2     Using if statements, relational
3     operators, and equality operators */
4  #include <stdio.h>
5
6  int main()
7  {
8     int num1, num2;
9
10    printf( "Enter two integers, and I will tell you\n"
11    printf( "the relationships they satisfy: " );
12    scanf( "%d%d", &num1, &num2  );   /* read two
13
14    if ( num1 == num2 )
15       printf( "%d is equal to %d\n", num1, num2 );
16
17    if ( num1 != num2 )
18       printf( "%d is not equal to %d\n", num1, num2 );
19
20    if ( num1 < num2 )
21       printf( "%d is less than %d\n", num1, num2 );
22
23    if ( num1 > num2 )
24       printf( "%d is greater than %d\n", num1, num2 );
25
26    if ( num1 <= num2 )
27       printf( "%d is less than or equal to %d\n",
28               num1, num2 );
```

Program Outline

1. Declare variables

2. Input

2.1 `if` statements

3. Print

Decision Making: Equality and Relational Operators

Program Outline

```
29
30    if ( num1 >= num2 )
31       printf( "%d is greater than or equal to %d\n",
32               num1, num2 );
33
34    return 0;   /* indicate program ended successfully */
35 }
```

3.1 Exit `main`

Program Output

```
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
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

Decision Making: Equality and Relational Operators

- **Example 5:** Write a C program which asks the user to enter two integers, compare them and perform the following actions:
  - if the first value is greater -> add the two numbers,
  - if the second value is greater -> multiply the integers
  - if they are equal -> divide their multiplication with their sum.