

C Programming Language

- Loop
- For - Loop

Dr. Mostafa A. Elhosseini

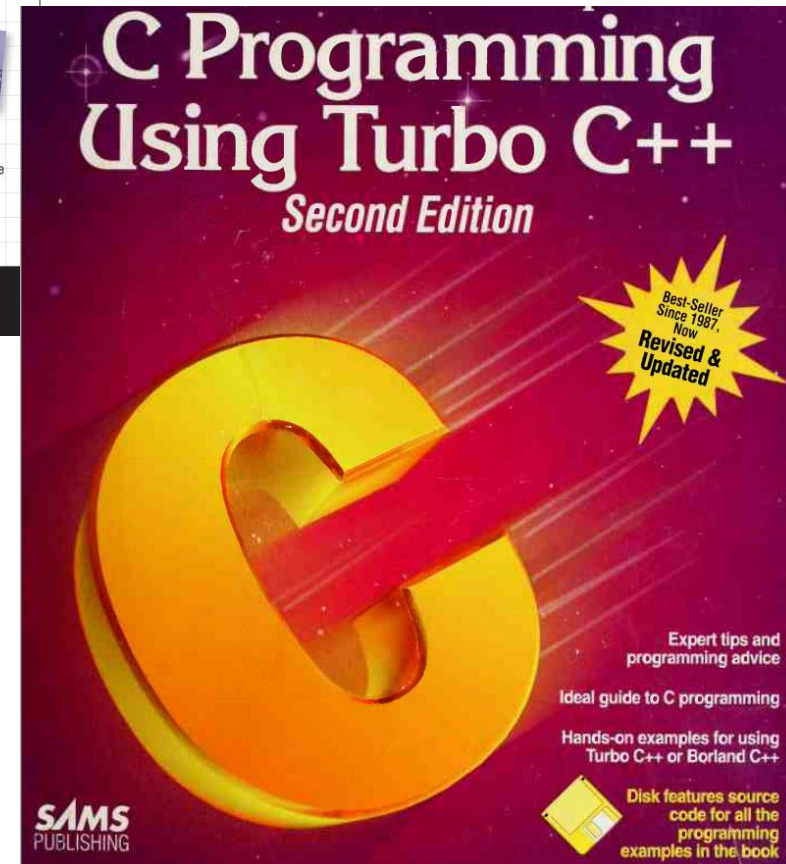
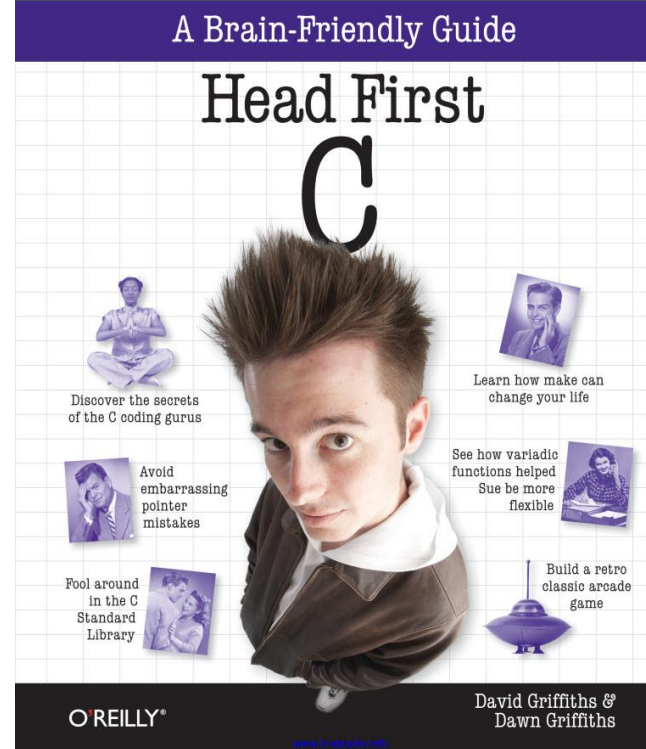
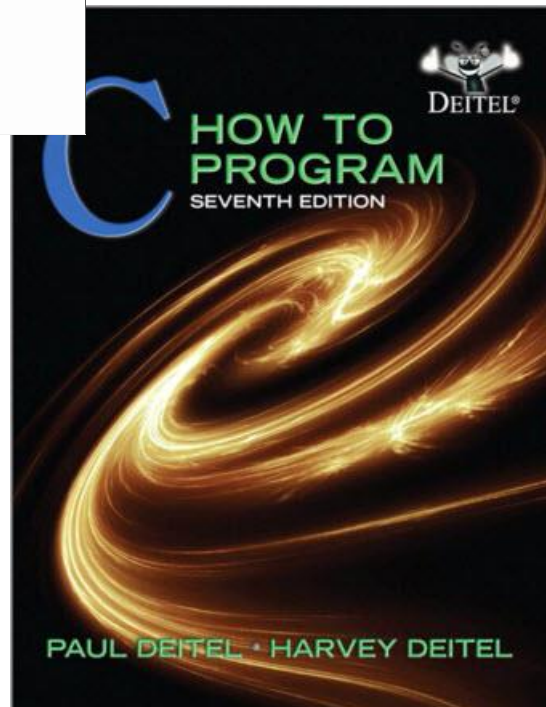
SECOND EDITION

THE



PROGRAMMING
LANGUAGE

BRIAN W. KERNIGHAN
DENNIS M. RITCHIE



Revise

✂ switch statement

- How it works
- case
- default
- break

✂ Ternary operator

Agenda

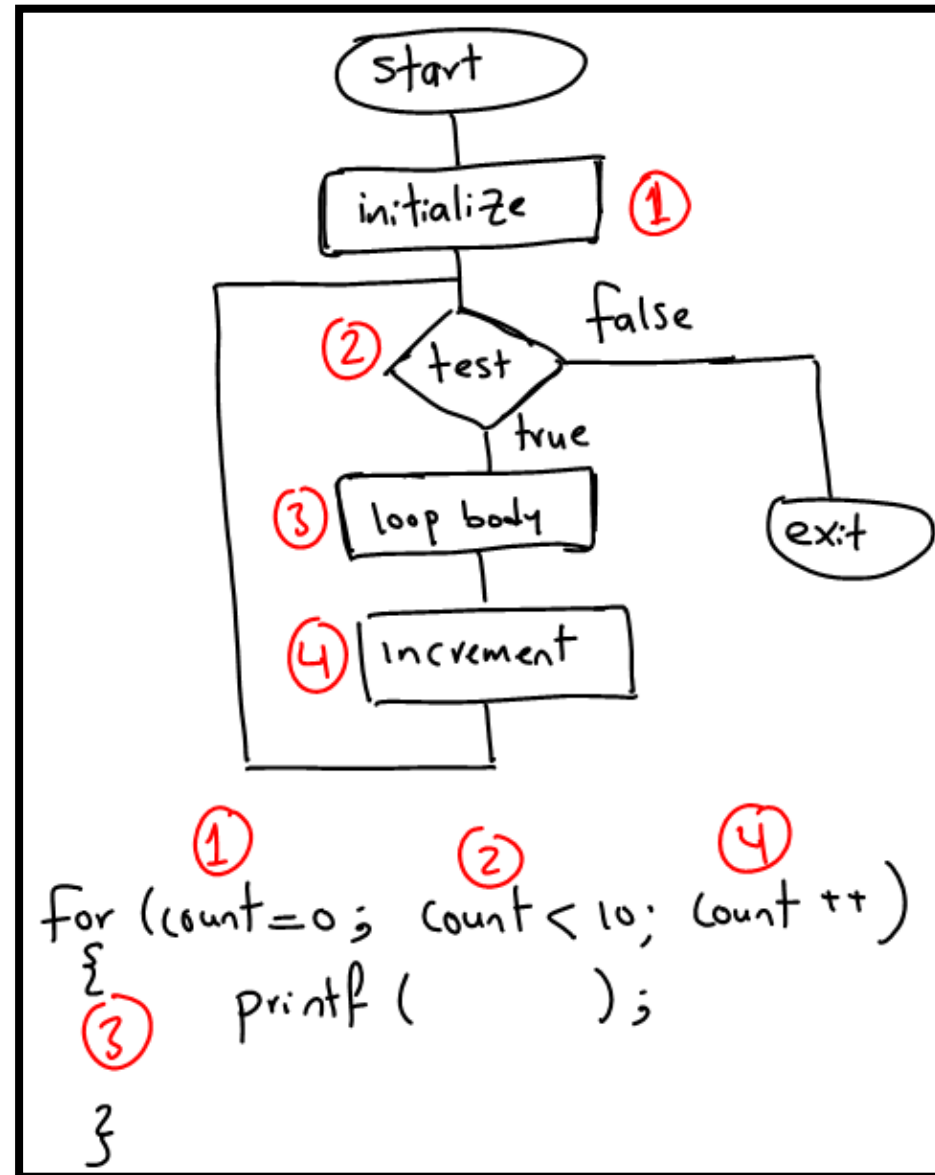
✂ LOOP

- FOR Loop

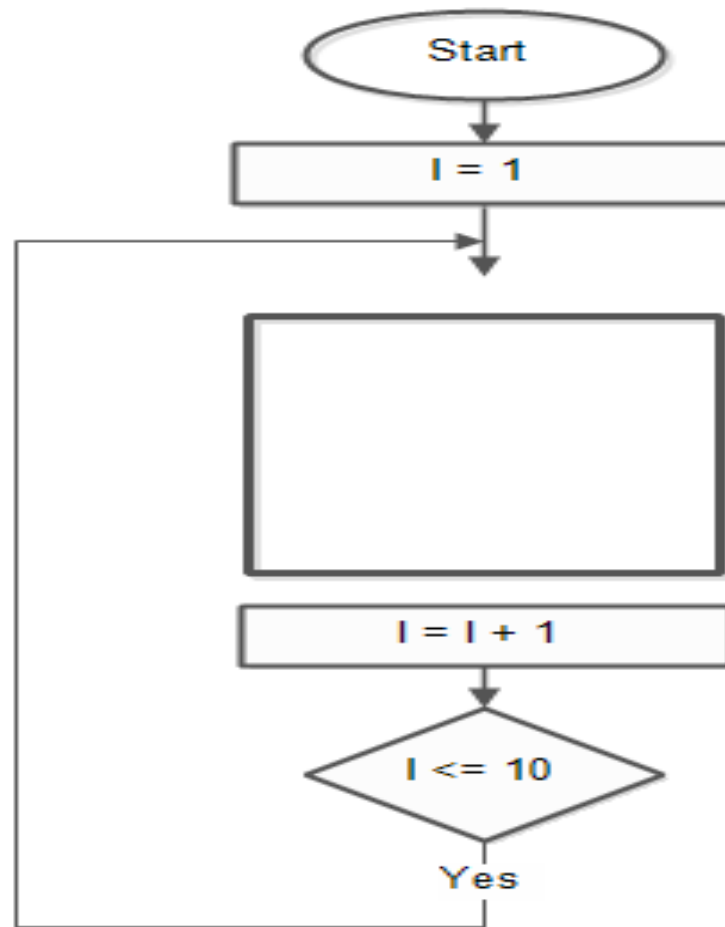
Loop

- ✧ If something is worth doing, it is worth doing more than once.
Programming is the same
 - we frequently need to perform an action over and over, often with variations in the details each time — the mechanism that meets this need is the loop
- ✧ There are three major loop structures in C
 - for loop
 - while loop
 - do-while
- ✧ Each loop must contain: (1) initialize (2) check (3) Update

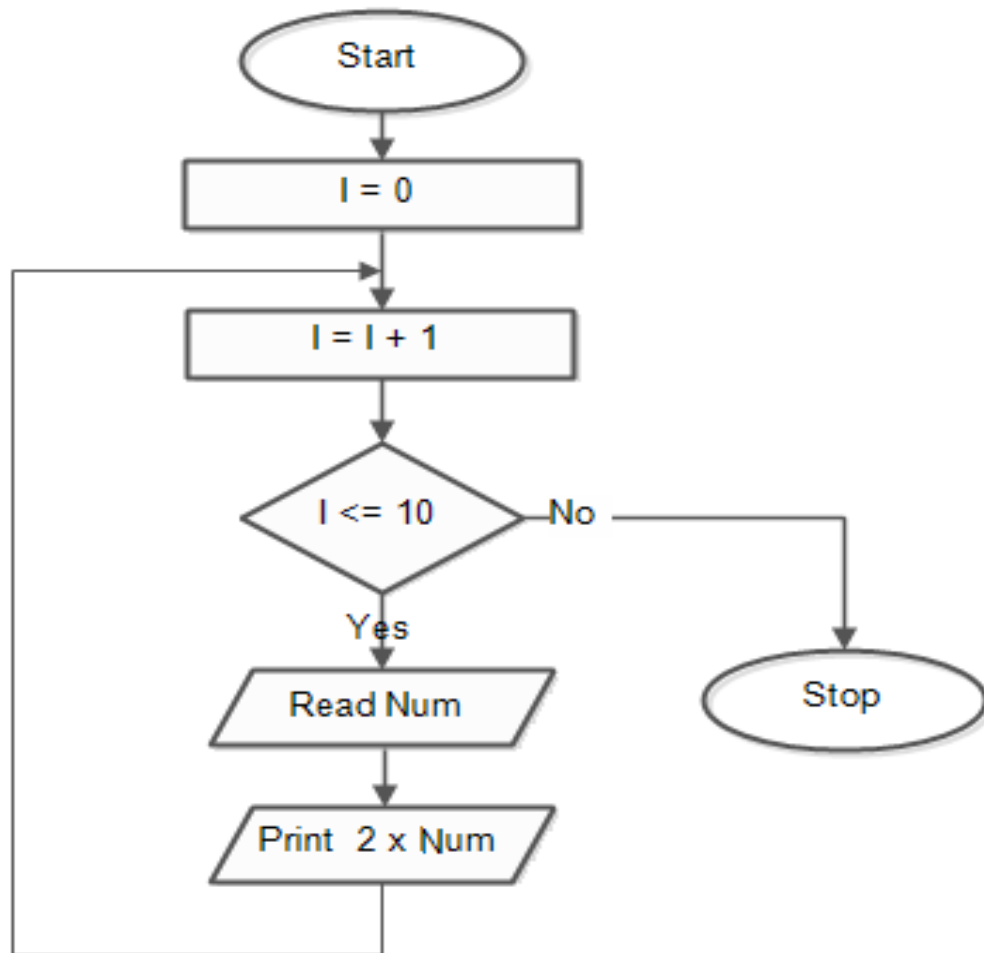
■ loop



Flowchart



Flowchart



for loop

- ✧ Encapsulates the essential elements of a loop into one statement
- ✧ You **want to do something a fixed number of times** — perhaps you want to calculate the **pay checks for 120 employees** or **print out the squares of all the numbers from 1 to 50**
- ✧ Here you should know **before executing** the program, **how many times you need to repeat your task**
- ✧ It is particularly well suited for loops in which the number of passes is known in advance

for loop

The diagram illustrates the components of a C for loop. The code is written as follows:

```
for (count = 0; count < 10; count++)  
{  
    printf("count = %d \n", count);  
    :  
}
```

Annotations in red:

- init expr**: An arrow points from this label to the initialization part `count = 0`.
- test expr**: An arrow points from this label to the condition part `count < 10`.
- no Semi Colon**: An arrow points from this label to the increment part `count++`.
- Keyword**: An arrow points from this label to the `for` keyword.
- body of the loop**: A bracket on the right side of the loop body (from the opening curly brace to the closing curly brace) is labeled with this text.

for loop

- ✧ The **fOR loop** may be **started at any value**,
 - here **count = 0**, however, **loops are often started at 1** or some other number
- ✧ **Count<10** — tests each time through the loop to see if count is less than 10.
 - if the **test expression is true** (count is less than 10), **the body of the loop will be executed**. If the expression becomes false, the loop will be terminated and control will pass to the statements following the loop
- ✧ **Count++**, increments the variable count each time the loop is executed
- ✧ The **loop variable** in a FOR loop **does not have to be increased by 1** each time through the loop
 - `count = count +3`

for loop

- ✧ Do not place a semicolon between the loop expression and the body of the loop, since the keyword for and the loop expression do not constitute a complete C statement
- ✧ It is not important to consider the braces when the for loop is one statement
- ✧ The for loop can execute zero time, if the condition is not satisfied at the beginning

Examples

```
1  #include <stdio.h>
2  int main()
3  {
4      int i;
5      for(i=0;i<10;i++)
6      {
7          printf("\n # iterations: %d",i);
8      }
9      return 0;
10 }
11
```

"E:\Courses\Taibah\Programming es

```
# iterations: 0
# iterations: 1
# iterations: 2
# iterations: 3
# iterations: 4
# iterations: 5
# iterations: 6
# iterations: 7
# iterations: 8
# iterations: 9
```

Examples

```
1  #include <stdio.h>
2  int main()
3  {
4      int i;
5      for(i=0;i<-2;i++)
6          printf("\n # iterations: %d",i);
7      printf("\n i value = %d",i);
8      return 0;
9  }
10
```

"E:\Courses\Taibah\Programming essential using C\Code\test\bin\Debug\test.exe

i value = 0

Examples

```
1  #include <stdio.h>
2  int main()
3  {
4      int i;
5      for(i=0; i<10; i+=2)
6          printf("\n # iterations: %d", i);
7      return 0;
8  }
```

"E:\Courses\Taibah\Programming es:

```
# iterations: 0
# iterations: 2
# iterations: 4
# iterations: 6
# iterations: 8
```

Examples

```
1  #include <stdio.h>
2  int main()
3  {
4      float i;
5      for(i=0;i<1;i+=0.1)
6          printf("\n # iterations: %f",i);
7      return 0;
8  }
9
```

```
# iterations: 0.000000
# iterations: 0.100000
# iterations: 0.200000
# iterations: 0.300000
# iterations: 0.400000
# iterations: 0.500000
# iterations: 0.600000
# iterations: 0.700000
# iterations: 0.800000
# iterations: 0.900000
```