DATA STRUCTURE & PROGRAMMING I

Topic 4- Introduction to loop and using loop for

Test if a given number is odd or even

Testcondion.c - Code::Blocks 20.03

```c
#include<stdio.h>

main(){
    int n;
    int r;
    int k=1;

    while(10>0){
        printf("\n\n%d Enter a number: ",k);
        scanf("%d" ,&n);

        r = n%2;

        if(r==0){
           printf("\t%d is an even number", n);
        }else if(r!=0){
           printf("\t%d is an odd number", n);
        }

        k=k+1;
    }

}
```

C:\Users\bouch\Desktop\Demo12\Testcondion.c | C/C++ | Windows (CR+LF) | WINDOWS-1252 | Line 19, Col 24, Pos 318 | Insert | Read/Write | default

Loop (iterator structure)

1. `for`
2. `while`
3. `do … while`

Introduction to loop

Types of problems and solutions

- Suppose that we want to display a message "Hi, what is your name" and ask for names of 100 students
- Solution 1: Using 100 repeated instructions

```
var name: Sequence of character
begin
    write("Hi, what is your name?")
    read(name)
    write("Hi, what is your name?")
    read(name)
    ...
    write("Hi, what is your name?")
    read(name)
end
```

100 repeated instructions

- Solution 2 (Better solution): Use iteration structure (loop)
  - Loop allows a block of instruction/codes to be executed repeatedly within a condition. Let i=1 and condition i<=100, we can repeat our code 100 times by each time update i=i+1.

FOR ... DO

FOR loop

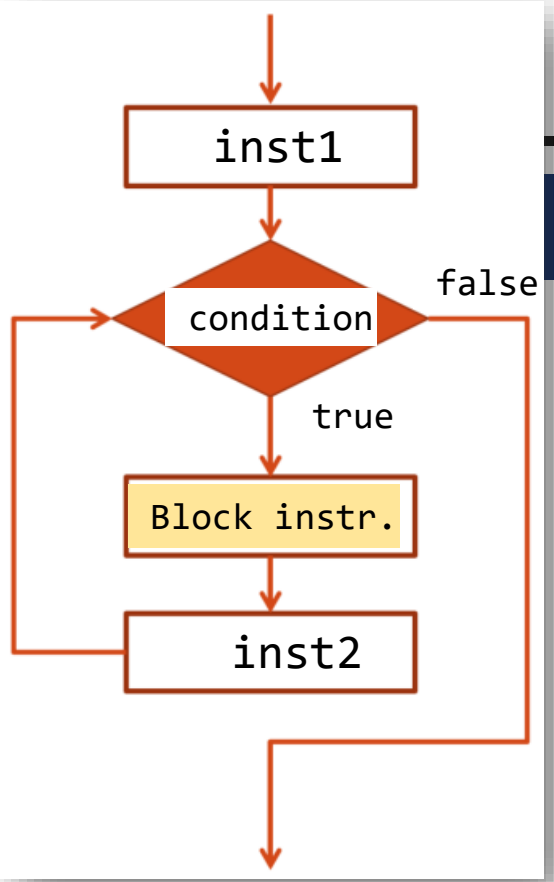
- Syntax:

instruction to initialize value of the control variable

loopback condition for stopping loop when it turns `false`

instruction to modify the value of control variable

```
for(inst1; condition; inst2) do
    block of instructions
end for
```



- Example:

```
var i: integer
begin
    for(i ← 25; i<=30; i ← i+1) do
        write(i, “ ”)
    end for
end
```

Output:

```
25 26 27 28 29 30
```

FOR … DO

Examples

```
var i: integer
begin
    for(i ← 1; i<=7; i ← i+2) do
        write(i, “ ”)
    end for
end
```

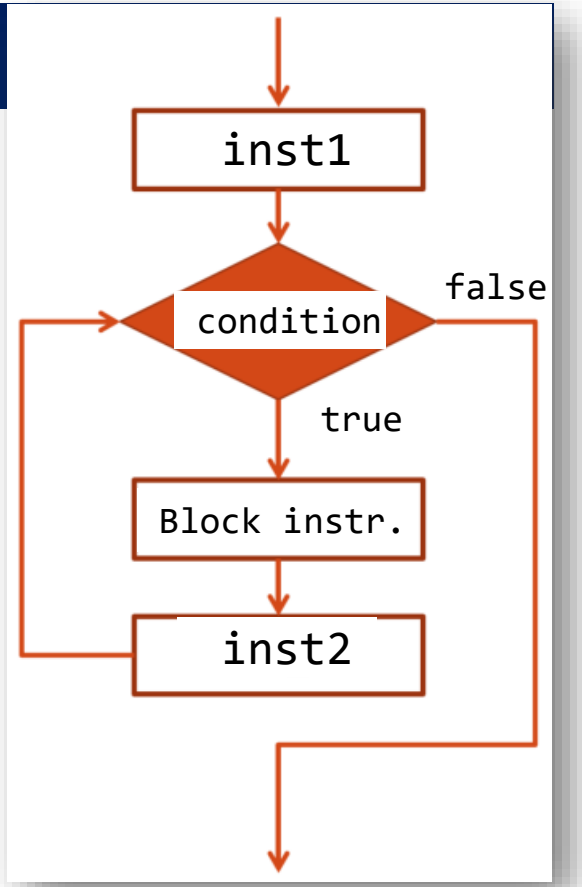
Output:

```
1 3 5 7
```

```
var i: integer
begin
    for(i ← 25; i>0; i ← i-5) do
        write(i, “ ”)
    end for
end
```

Output:

```
25 20 15 10 5
```



FOR … DO

Examples

```
var i: integer
begin
    for(i ← 7 ; i>7; i ← i+2) do
        write(i, “ ”)
    end for
end
```

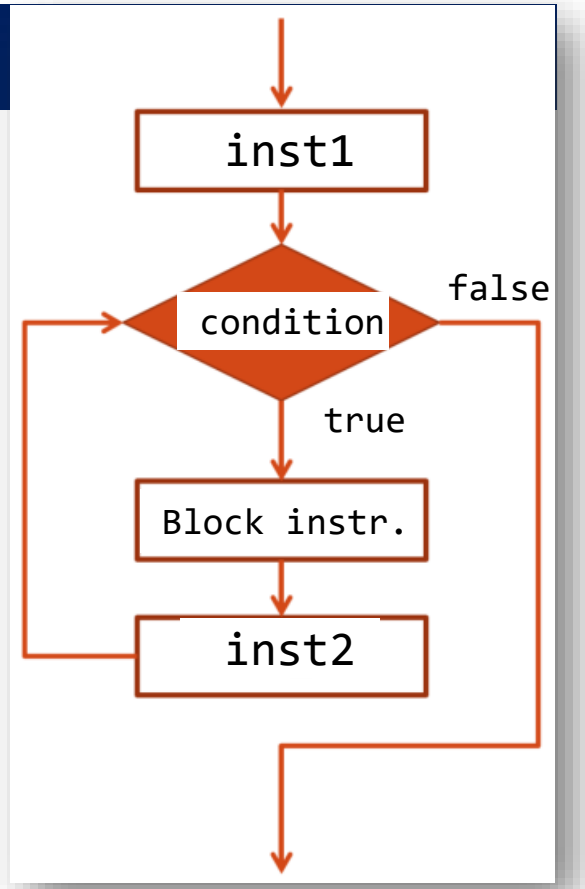
Output:

```
?
```

```
var i: integer
begin
    for(i ← 25; i>0; i ← i+1) do
        write(i, “ ”)
    end for
end
```

Output:

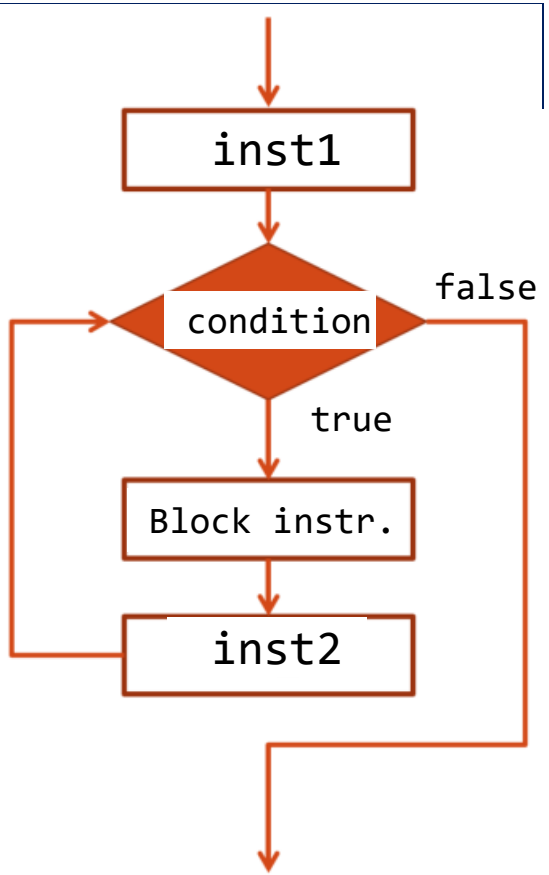
```
25  26  27  …
```



FOR ... DO

More examples

```
var i, j : integer
begin
    for(i ← 1; i <= 3; i ← i+1) do
        for(j ← 1; j <= 4; j ← j+1) do
            write("A")
            write("B")
        end for
        write(" ")
    end for
end
```



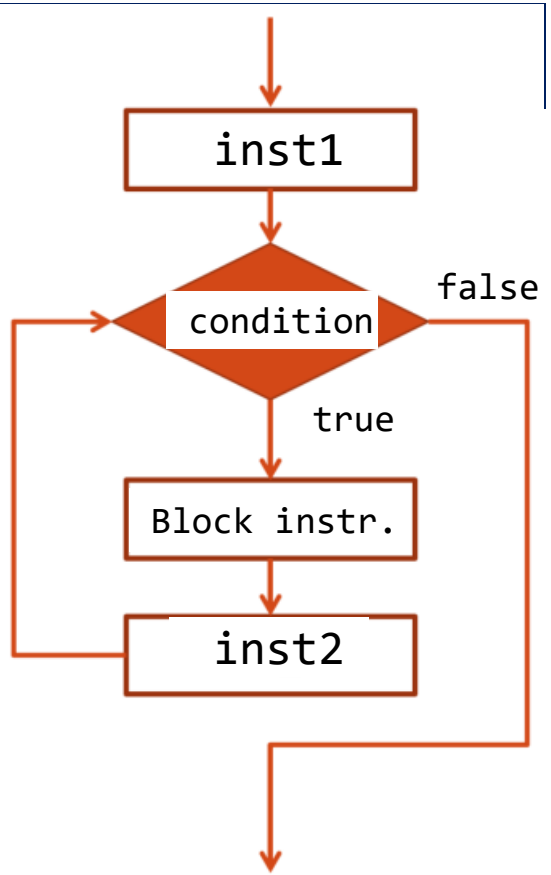
Output:

```
ABABABAB ABABABAB ABABABAB
```

FOR … DO

More examples

```
var i, j, k, n : integer
begin
    n ← 4
    for(i ← 1; i <= n; i ← i+1) do
        for(j ← 1; j <= n-i; j ← j+1) do
            write(“ ”)
        end for
        for(k ← 1; k <= 2*i-1; k ← k+1) do
            write(“*”)
        end for
        write(“\n”)
     end for
end
```



Output:

```
i=1     *
i=2    ***
i=3   *****
i=4  *******
```

Break 10mn
Start: 8:25am

Break Vs. Continue keyword

`break` statement breaks the loop/switch whereas

`continue` skip the execution of current iteration and continue to the next iteration (it does not break the loop/switch)

BREAK and CONTINUE statements

BREAK Vs. CONTINUE

- **Break**: allows to break/terminate the running loop
- **Continue**: allows to skip 1 iteration, so any instructions followed by **continue** will be skipped then the loop continue the next iteration
- Examples

```
var i : integer
begin
    for(i ← 1; i<=9; i ← i+1) do
        if (i==4) then
            continue
        end if
        write(i)
    end for
end
```

Output: 12356789

```
var i : integer
begin
    for(i ← 1; i<=10; i ← i+1) do
        if (i==3) then
            continue
        end if
        if (i==8) then
            break
        end if
        write(i)
    end for
end
```

Output:

124567

C program

Loop: for

- Syntax

```
for(initStatement; condition; updateStatement){
        //codes
}
```

Syntax in C

1

```
#include <stdio.h>
int main(){
   int n,i;

   for(i=0; i<20; i++){
        printf("%d ", i);
   }
}
```

2

```
#include <stdio.h>
int main(){
   int n;

   for(int i=0; i<20; i++){
        printf("%d ", i);
   }
}
```

Examples of using **for** loop in C

3

```
#include <stdio.h>
int main(){
   int num;
   printf("Enter a number: ");
   scanf("%d", &num);

   for(int i=0; i<num; i++){
        printf("%d ", i);
   }
}
```

Output:

```
0 1 2 … (num-1)
```

4

```
#include <stdio.h>
int main(){
   int num;
   printf("Enter a number: ");
   scanf("%d", &num);

   for(int i=num; i>0; i--){
        printf("%d ", i);
   }
}
```

Output:

```
num (num-1) … 1
```

5

```
#include <stdio.h>
int main(){
   int num;
   printf("Enter a number: ");
   scanf("%d", &num);

   for(int i=0; i<num; i--){
        printf("%d ", i);
   }
}
```

Output (infinite loop)**:**

```
0 -1 -2 …
```

Example: A program to display and sum a suit of number

- Using for loop to display and sum all numbers from 1 to 99.

for-loop.c - Code::Blocks 20.03

File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help

Practice1.c | for-loop.c | while1.c

```c
#include<stdio.h>
main(){
    int sum=0;
    for(int k=1; k<=99; k=k+1){
      //codes belong to for loop
      //
      printf("%d ", k);
      sum=sum+k;
    }
    printf("\n\tTotal sum: %d", sum);
}
```

"G:\My Drive\Working\CADT\Advance Algo (T3 2022-2023)\Exam\for-loop.exe"

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84
85 86 87 88 89 90 91 92 93 94 95 96 97 98 99
        Total sum: 4950
Process returned 0 (0x0)   execution time : 0.058 s
Press any key to continue.
```

Example: A program to check if a given input number is a prime number

```c
int number;
int state=1;  //suppose 1

printf("\n*** A program to check a prime number\n");
printf("Enter a number: ");
scanf("%d", &number);

for(int p=2; p<number; p=p+1){
    printf("%d ", p);
    if(number%p == 0){ //finding remainder to check divisible
        state = 0;
        printf("\n\t Divider %d",p);
        break;
    }
}
if(state == 0){
    printf("\n\t %d is not a prime number\n", number);
}else{
    printf("\n\t %d is a prime number\n", number);
}
```

```
*** A program to check a prime number
Enter a number: 119
2 3 4 5 6 7
        Divider 7
        119 is not a prime number

*** A program to check a prime number
Enter a number: _
```

Assignment (part 1)

Loop exercises: Write a C program to …

1. Display all numbers from 99 to 1.
2. Display all numbers from 1 to 100 except the number 50.
3. Display odd numbers between 8 to 1000 except the numbers 11, 17 and 21.
4. Show all integer divisible by 3 between 1 to 100 except 30, 60, and 90.
5. Sum all numbers from 1 to 100 then display the result.
6. Multiply all numbers from 1 to 100 then display the result.

Assignment (part 1)

Loop exercises: Write a C program to …

7. Display the words “Hi” 20 times and then “bye” 10 time using **For loop**. One line for displaying the word “Hi”, and another line for displaying the word “bye”.
8. Display all even numbers ***between*** 0 to 30.
9. Calculate factorial of integer number n, where n is a positive number entered by a user.
10. Write an algorithm to sum suite number from 1 to n, where n is a positive number entered by a user.

Assignment (part 2)

Loop exercises: Write a C program to …

5. Compute and display the summation of the suit cube number starting from 1 up to n, where n is the input number entered by a user, n is greater than 1.
   5. Ex: Suppose the input is 3, then display $1^3 + 2^3 + 3^3 = 36$
6. Check whether an input number is a primary number or not. The program runs indefinitely so that we can always check another input number.
7. Display all primary numbers in between 2 to 500.
8. Read 10 input numbers from a user and then find the maximum number and display it on screen.

Assignment (part 2)

Loop exercises: Write a C program to …

9. Compute and display the summation of the suit cube number starting from n up to 1, where n is the input number entered by a user, n is greater than 1.
   9. Ex: Suppose the input is 3, then display 3^3 + 2^3 + 1^3 = 36
10. Check whether an input number is a primary number or not. The program runs indefinitely so that we can always check another input number.
11. Display all primary numbers in between 2 to 500.
12. Read 20 input numbers from a user and then find the maximum number and display it on screen.

Assignment (part 2)

Loop exercises: Write a C program to …

13. Check whether an input number is a **perfect number or not.** The program runs indefinitely so that we can always check another input number.

Perfect number is a positive integer that is equal to the sum of its proper divisors, excluding the number itself.
E.g: 6 is a perfect number.

- 6 has divisors 1, 2 and 3
- Since the sum of its divisors 1, 2, and 3 are equal to 6.

Perfect Number	Sum of its Divisors
6	1 + 2 + 3
28	1 + 2 + 4 + 7 + 14
496	1 + 2 + 4 + 8 + 16 + 31 + 62 + 124 + 248
8,128	1 + 2 + 4 + 8 + 16 + 32 + 64 + 127 + 254 + 508 + 1,016 + 2,032 + 4,064

Assignment (part 2)

Loop exercises: Write a C program to …

14. Read 10 input numbers from a user and then find the minimum number and display it on screen.

15. Display the first n numbers of suit Fibonacci, where n is a number entered by a user.

The Fibonacci Sequence is the series of numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...
The next number is found by adding up the two numbers before it.

- The 2 is found by adding the two numbers before it (1+1)
- The 3 is found by adding the two numbers before it (1+2),
- And the 5 is (2+3),
- and so on!

Assignment (part 2)

Loop exercises: Write a C program to …

16. Ask a user to input many numbers as possible. When the user inputs 0 or -1, stop asking the user for the number and display the total summation of all input numbers on the screen.