

Repeating Statements

Fundamentals of Computer and Programming

Fall 2020

Bahador Bakhshi

CE & IT Department, Amirkabir University of Technology



What We Will Learn

- Introduction
- **while** statement
- **do-while** statement
- **for** statement
- Arrays
- Advanced loops
- Bugs and avoiding them



What We Will Learn

- Introduction
- `while` statement
- `do-while` statement
- `for` statement
- Arrays
- Advanced loops
- Bugs and avoiding them



Repetition

- Example: Write a program that read 3 integer and compute average
 - It is easy. 3 scanf, an addition, a division and, a printf
- Example: Write a program that read 3000 integer and compute average
 - ?? 3000 scanf !!!
- Example: Write a program that read n integer and compute average
 - N??? scanf
-



Repetition: counter controlled

- When we know the number of iteration
 - Average of 10 number

Initialize counter $\leftarrow 0$

Initialize other variables

While (counter < number of loop repetition)
 do something (e.g. read input, take sum)
 counter \leftarrow counter + 1



Repetition: sentinel controlled

- When we do **NOT** know the number of iteration
- But we know, when loop terminates
 - E.g. Average of arbitrary positive numbers ending with <0

Get first input $\rightarrow n$

While (n is not sentinel)

do something (sum, ...)

get the next input $\rightarrow n$

if (there is not any valid input) then S1

else S2



Repetition

- Repetition is performed by loops
 - Put all statements to repeat in a **loop**
- Don't loop to infinity
 - Stop the repetition
 - Based on some conditions (counter, sentinel)
- C has three statements for loops
 - **while** statement
 - **do-while** statement
 - **for** statement



What We Will Learn

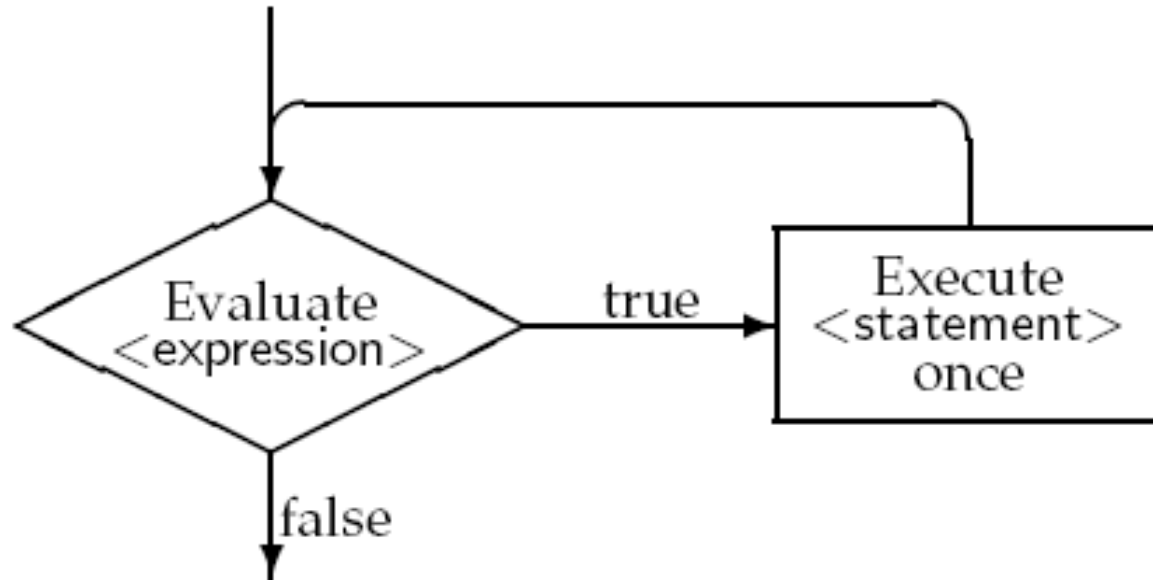
- Introduction
- **while** statement
- `do-while` statement
- `for` statement
- Arrays
- Advanced loops
- Bugs and avoiding them



while statement

while (**<expression>**)

<statements>



```
#include <stdio.h>
```

برنامه‌ای بنویسید که عدد n را از کاربر بگیرد و اعداد 0 تا n را چاپ کند.

```
int main(void) {  
    int n, number;  
    number = 0;  
    printf("Enter n: ");  
    scanf("%d", &n);  
    while(number <= n) {  
        printf("%d \n", number);  
        number++;  
    }  
    return 0;  
}
```

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main(void){
    int negative_num, positive_num;
    int number;
    negative_num = positive_num = 0;
    printf("Enter Zero to stop \n");
    printf("Enter first number: ");
    scanf("%d", &number);
    while(number != 0){
        if(number > 0)
            positive_num++;
        else
            negative_num++;

        printf("Enter next number: ");
        scanf("%d", &number);
    }
    printf("The number of positive numbers = %d\n", positive_num);
    printf("The number of negative numbers = %d\n", negative_num);
    return 0;
}
```

برنامه‌ای بنویسید که یک سری عدد را از کاربر بگیرد و تعداد اعداد مثبت و منفی آنرا بشمارد. این سری اعداد با صفر تمام می‌شود.

What We Will Learn

- Introduction
- `while` statement
- **`do-while`** statement
- `for` statement
- Arrays
- Advanced loops
- Bugs and avoiding them

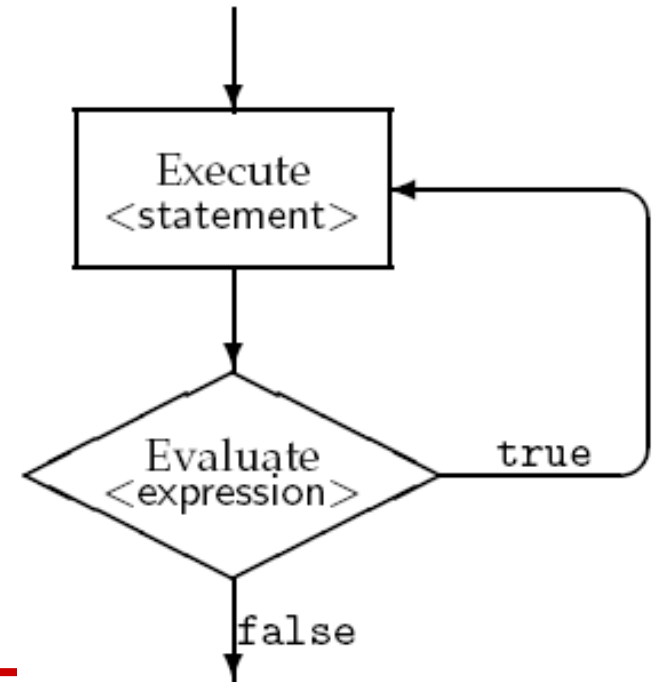


do-while statement

do

<statements>

while (<expression>);



```
#include <stdio.h>
#include <stdlib.h>
```

```
int main(void){
    int n;
    double number, sum;
    printf("Enter n > 0: ");
    scanf("%d", &n);
    if(n < 1){printf("wrong input"); return -1;}

    sum = 0;
    number = 0.0;
    do{
        number++;
        sum += number / (number + 1.0);
    }while(number < n);

    printf("sum = %f\n", sum);
    return 0;
```

برنامه‌ای بنویسید که عدد n را بگیرد و
مجموع n جمله اول رشته زیر را حساب کند

$$1.0/2.0 + 2.0/3.0 + 3.0/4.0 + \dots$$

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int negative_num=0, positive_num=0;
    int number;
    printf("Enter Zero to stop \n");
    do{
        printf("Enter next number: ");
        scanf("%d", &number);
        if(number > 0)
            positive_num++;
        else if(number < 0)
            negative_num++;
    }while(number != 0);

    printf("The number of positive numbers = %d\n", positive_num);
    printf("The number of negative numbers = %d\n", negative_num);
    return 0;
}
```

برنامه‌ای بنویسید که یک رشته
عدد را از کاربر بگیرد و تعداد اعداد
مثبت و منفی آنرا بشمارد. این
رشته اعداد با صفر تمام می‌شود.

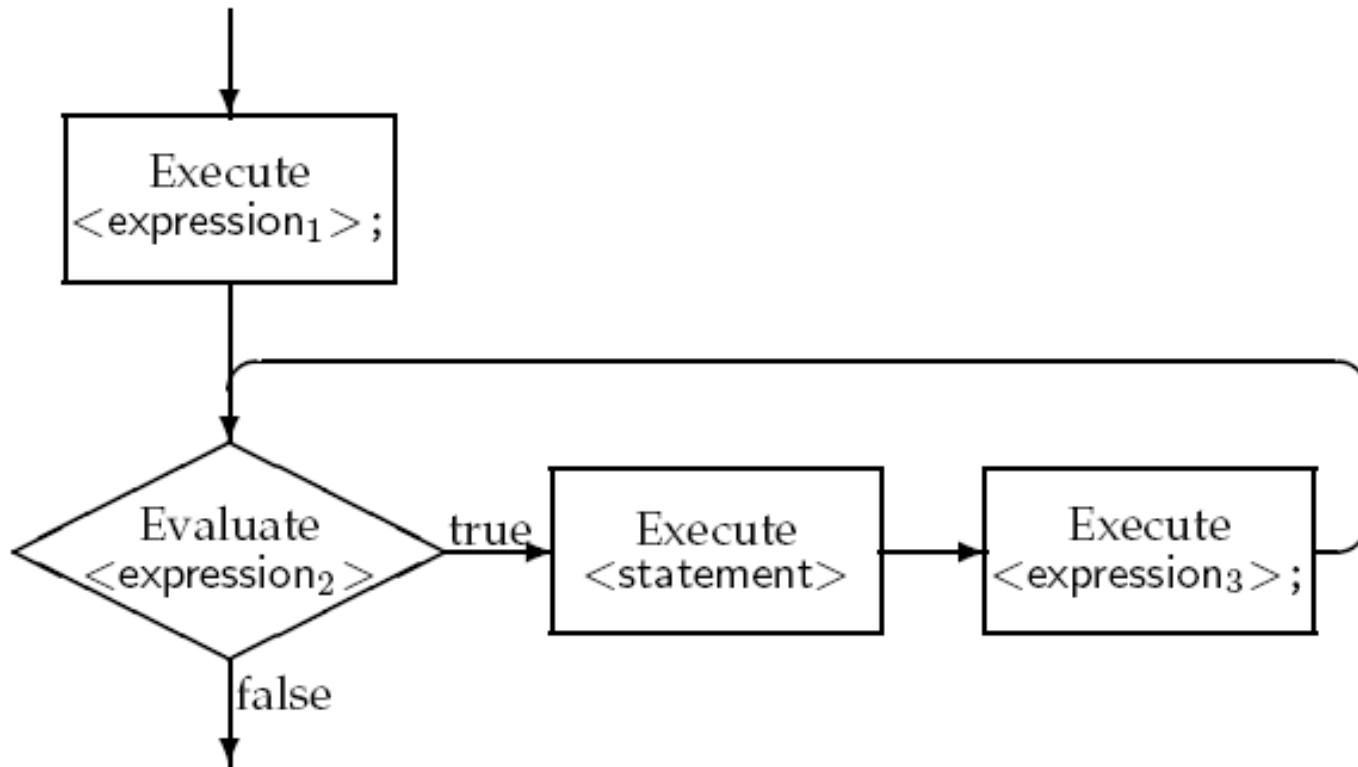
What We Will Learn

- Introduction
- `while` statement
- `do-while` statement
- **`for`** statement
- Arrays
- Advanced loops
- Bugs and avoiding them



for statement

```
for (<expression1>;  
    <expression2>; <expression3>)  
    <statements>
```



```
#include <stdio.h>
```

```
int main(void){
```

```
    int grade, count, i;
```

```
    double average, sum;
```

```
    sum = 0;
```

```
    printf("Enter the number of students: ");
```

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

```
    for(i = 0; i < count; i++){
```

```
        printf("Enter the grade of %d-th student: ", (i + 1));
```

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

```
        sum += grade;
```

```
    }
```

```
    average = sum / count;
```

```
    printf("The average of your class is %0.3f\n", average);
```

```
    return 0;
```

برنامه‌ای که تعداد دانشجویان و نمره‌های آنها را خوانده و میانگین را محاسبه کند.

```
#include <stdio.h>
```

```
int main(void){
```

```
    int n, number;
```

```
    printf("Enter n: ");
```

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

```
    for(number = 2; number <= n; number += 2)
```

```
        printf("%d \n", number);
```

```
    return 0;
```

برنامه‌ای که عدد n را از کاربر بگیرد و همه اعداد زوج کوچکتر مساوی آن را چاپ کند.

```
#include <stdio.h>
```

```
int main(void){
```

```
    int n, number;
```

```
    printf("Enter n: ");
```

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

```
    for(number = 1; number <= n; number++)
```

```
        if((number % 2) == 0)
```

```
            printf("%d \n", number);
```

```
    return 0;
```

برنامه‌ای که عدد n را از کاربر بگیرد و همه اعداد زوج کوچکتر مساوی آن را چاپ کند.

Expressions in `for` statements

- Expression1 and Expression3 can be any number of expressions, they execute in the order
 - `for(i = 0, j = 0; i < 10; i++, j--)`
- Expression2 at most should be a single expression
 - If multiple expressions → the value of the last one is evaluated as True/False
 - `for(i = 0, j = 0; i < 10, j > -100; i++, j--)`
- Any expression can be empty expression
 - `for(; i < 10; i++)`
 - `for(; ;)`



What We Will Learn

- Introduction
- `while` statement
- `do-while` statement
- `for` statement
- **Arrays**
- Advanced loops
- Bugs and avoiding them



Introduction

- Algorithms usually work on large data sets
 - Sort a set of numbers
 - Search a specific number in a set of numbers
- How to read and store a set of data?
- To read
 - Repeat the scanf statement
 - Use the loop statements
- To store the data
 - Save each data in a single variable??
 - 3000 int variables! ! !



Array

- An **ordered** collection of **same type** variables
- A $n \times 1$ vector of
 - Integers, chars, floats, ...

- Example

- An array of 8 integer

0	1	2	3	4	5	6	7
3	1	5	11	10	19	0	12

- An array of 5 chars

0	1	2	3	4
'a'	'z'	'F'	'z'	'k'



Arrays in C

➤ Array declaration in C

<Elements' Type> <identifier>[<size>]

➤ <Elements' Type>: int, char, float, ...

➤ <size>

➤ Old compilers (standard): it should be constant

➤ New compilers (standard): it can be variable

➤ Elements in array

➤ From 0 to (size – 1)



Example

```
int num[20];
```

- **num** is array of 20 **integers**
- **num[0]** is the first integer variable
- **num[19]** is the last integer

```
float farr[100];
```

- **farr** is array of 100 **floats**
- **farr[0]** is the first float
- **farr[49]** is the 50th **float**
- **farr[99]** is the last float



Example

```
int number[10];
```

```
int i, j = 3;
```

```
i = 5; // -1 < i < 10
```

```
number[i] = 0;           //6th number is 0
```

```
number[i + j] = 1;       //??
```

```
j = number[i];           //?
```

```
j = number[i + 1];       //?
```

```
j = number[i] + 1;       //?
```



```
#include <stdio.h>

#define SIZE 20

void main(void) {
    int number[SIZE];
    double average;
    int sum, large_size, small_size, i;
    sum = large_size = small_size = 0;
    for(i = 0; i < SIZE; i++){
        int tmp;
        scanf("%d", &tmp);
        number[i] = tmp;
        sum += number[i];
    }
    average = (1.0 * sum) / SIZE;
    for(i = 0; i < SIZE; i++)
        if(number[i] >= average)
            large_size++;
        else
            small_size++;
    printf("average = %f\n", average);
    printf("Small Size = %d, Large Size = %d\n", small_size, large_size);
}
```

برنامه‌ای که ۲۰ عدد را بگیرد و
تعداد اعداد بزرگتر و کوچکتر از
میانگین را حساب کند.

```
#include <stdio.h>
```

```
void main(void){
```

```
    int n;
```

```
    printf("Enter n: ", n);
```

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

```
    int number[n];
```

```
    double average;
```

```
    int sum, large_size, small_size, i;
```

```
    sum = large_size = small_size = 0;
```

```
    for(i = 0; i < n; i++)
```

```
        scanf("%d", &(number[i]));
```

```
    for(i = 0; i < n; i++)
```

```
        sum += number[i];
```

```
    average = (1.0 * sum) / n;
```

```
    for(i = 0; i < n; i++)
```

```
        if(number[i] >= average)
```

```
            large_size++;
```

```
        else
```

```
            small_size++;
```

```
    printf("average = %f\n", average);
```

```
    printf("Small Size = %d, Larg Size = %d\n", small_size, large_size);
```

```
}
```

برنامه‌ای که تعداد اعداد و یک رشته
عدد را بگیرد و تعداد اعداد بزرگتر و
کوچکتر از میانگین را حساب کند.

Array Initialization: Known Length

```
int num[3]={10, 20, 60};
```

- **num** is the array of **3** integers, **num[0]** is 10, ...

```
int num[]={40, 50, 60, 70, 70, 80};
```

- **num** is the array of **6** integers

```
int num[10]={40, 50, 60};
```

- **num** is the array of **10** integers
- **num[0]** is 40, **num[1]** is 50, **num[2]** is 60
- **num[3]** , **num[4]** , ... , **num[9]** are 0



Array Initialization (cont'd)

```
int num[2]={40, 50, 60, 70};  
/* Compile warning */
```

```
int num[5]={ [0] = 3, [4] = 6 };  
/* num[5] = {3, 0, 0, 0, 6} */
```



Initializing Variable Length Arrays

```
int n;  
scanf("%d", &n);  
int num[n]={0}; /* Compile error */
```

Variable length arrays cannot be initialized!

Solution

```
for (i = 0; i < n; i++)  
    num[i] = 0;
```



What We Will Learn

- Introduction
- `while` statement
- `do-while` statement
- `for` statement
- Arrays
- Advanced loops
- Bugs and avoiding them



Empty statements

➤ <statement> in loops can be empty

```
while (<expression>) ;
```

E.g. ,

```
while (i++ <= n) ;
```

```
for (<expression1>; <expression2>;  
    <expression3>) ;
```

E.g. ,

```
for (i = 0; i < 10; printf ("%d\n", i), i++) ;
```



Nested loops

➤ <statement> in loops can be loop itself

```
while (<expression0>)
```

```
    for (<expression1>; <expression2>;  
        <expression3>)
```

```
        <statements>
```

```
for (<expression1>; <expression2>;  
    <expression3>)
```

```
do
```

```
    <statements>
```

```
while (<expression>)
```



Nested loops example

➤ A program that takes n and m and prints

**** (m * in each line)

****

...

****

(n lines)



```
#include <stdio.h>
```

```
int main(void) {
```

```
    int i, j, n, m;
```

```
    printf("Enter n & m: ");
```

```
    scanf("%d%d", &n, &m);
```

```
    for(i = 0; i < n; i++) {
```

```
        for(j = 0; j < m; j++)
```

```
            printf("*");
```

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

```
    }
```

```
    return 0;
```

```
}
```

Nested loops example

➤ A program that takes n and prints

* (i * in i-th line)

**

*** *
....

(n lines)



```
#include <stdio.h>
```

```
int main(void){
```

```
    int i, j, n;
```

```
    printf("Enter n: ");
```

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

```
    i = 1;
```

```
    while(i <= n){
```

```
        for(j = 0; j < i; j++)
```

```
            printf("*");
```

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

```
        i++;
```

```
    }
```

```
    return 0;
```

```
}
```

Example

- A program that takes a number and generates the following pattern

Input = 5

```
*
**
***
****
*****
****
***
**
*
```

```
for(i= 1; i <= n; i++){
    for(j = 0; j < i-1; j++)
        printf(" ");

    for(j = 1; j <= i; j++)
        printf("*");

    printf("\n");
}

for(i= n-1; i >= 1; i--){
    for(j = 1; j < i; j++)
        printf(" ");
    for(j = 1; j <= i; j++)
        printf("*");
    printf("\n");
}
```



break statement

- Exit from loop based on some conditions

```
do{  
    scanf ("%d", &a) ;  
    scanf ("%d", &b) ;  
    if (b == 0)  
        break ;  
    res = a / b ;  
    printf ("a / b = %d\n", res) ;  
}while (b > 0) ;
```



continue statement

- Jump to end of loop and continue repetition

```
do{  
    scanf ("%f", &a) ;  
    scanf ("%f", &b) ;  
    if (b == 0)  
        continue ;  
    res = a / b ;  
    printf ("a / b = %f\n", res) ;  
}while (a > 0) ;
```



Which loop?

- When you know the number of repetition
 - Counter-controlled loops
 - Usually, **for** statements
- When you don't know the number of repetitions (sentinel loop)
 - Some condition should be check before starting loop
 - Usually, **while** statement
 - The loop should be executed at least one time
 - Usually, **do-while**



What We Will Learn

- Introduction
- `while` statement
- `do-while` statement
- `for` statement
- Arrays
- Advanced loops
- Bugs and avoiding them



Common bugs and avoiding them

➤ Loop should terminate

- E.g., in `for` loops, after each iteration, we should approach to the stop condition

```
for (i = 0; i < 10; i++) //OK
```

```
for (i = 0; i < 10; i--) //Bug
```

➤ Initialize loop control variables

```
int i;
```

```
for ( ; i < 10; i++)
```



Common bugs and avoiding them

- Don't modify `for` loop controller in loop body

```
for(i = 0; i < 10; i++) {  
    ...  
    i--; //Bug  
}
```

- Take care about wrong control conditions

- < VS. <=

- = VS. ==

```
int b = 10;  
while(a = b) { //it means while(true)  
    scanf("%d", &a)
```



Reference

- **Reading Assignment:** Chapter 4 of “C How to Program”

