C Developer

Conditional and Iterative Statements

Course Objectives

- ✓ Manage conditions
- ✓ Handle loops

Course Plan

1. Conditional Statements

2. Iterative Statements

1. Conditional Statements

1. Conditional Statements

if

- To make a choice, use **if** statement:
  - One or more conditions
  - Create instruction blocks
  - One or more instruction blocks

- Syntax:

```
if(condition) {

    //Executed if the condition is fulfilled

}
```

1. Conditional Statements

if

```c
#include <stdio.h>

int main()
{
    double x;
    scanf("%lf", &x);
    if(x < 0) {
        x = -x;
    }
    printf("|x| = %lf\n", x);
    return 0;
}
```

```
-42.78
|x| = 42.780000
```

1. Conditional Statements

if … else

- Syntax:

```
if(condition) {

    //Executed if the condition is fulfilled

} else if(condition2) {

    //Executed if the condition2 is fulfilled

} else {

    //Executed if both condition and condition2 are not fulfilled

}
```

1. Conditional Statements

if … else

```c
#include <stdio.h>

int main()
{
    double x, y, max;
    printf("x = ");
    scanf("%lf", &x);
    printf("y = ");
    scanf("%lf", &y);
    if(x < y) {
        max = y;
    } else {
        max = x;
    }
    printf("max = %lf\n", max);
    return 0;
}
```

```
x = 5.6
y = 8.14
max = 8.140000
```

1. Conditional Statements

if … else

```c
#include <stdio.h>

int main()
{
    double x, y, max;
    printf("x = ");
    scanf("%lf", &x);
    printf("y = ");
    scanf("%lf", &y);
    max = (x < y) ? y : x;
    printf("max = %lf\n", max);
    return 0;
}
```

… or use the conditional operator here!

```
x = 5.6
y = 8.14
max = 8.140000
```

1. Conditional Statements

Nested if … else

```c
#include <stdio.h>

int main()
{
    double mark;
    printf("mark = ");
    scanf("%lf", &mark);
    if(mark < 10) {
        printf("Resit!\n");
    } else {
        if(mark > 15) {
            printf("So skilled!\n");
        } else {
            printf("Validated!\n");
        }
    }
    return 0;
}
```

```
mark = 13.5
Validated!
```

1. Conditional Statements

switch

- Can be used for multiple nestings

- Syntax:

```
switch(expression) {
case value1:
    //Executed if the expression matches the value1
case value2:
    //Executed if the expression matches the value2
default:
    //Executed by default
}
```

1. Conditional Statements

switch

- When the value of the expression matches one of the **case** values, the program executes the instructions of this **case**, but also all those that follow

- If you want to perform only the instructions of the concerned **case**, you must add the **break** keyword

1. Conditional Statements

switch

```c
#include <stdio.h>

int main()
{
    int n = 2;
    switch(n) {
    case 1:
        printf("n = 1\n");
    case 2:
        printf("n = 2\n");
    case 3:
        printf("n = 3\n");
    default:
        printf("n = ?\n");
    }
    return 0;
}
```

```
n = 2
n = 3
n = ?
```

1. Conditional Statements

switch

```c
#include <stdio.h>

int main()
{
    int n = 2;
    switch(n) {
    case 1:
        printf("n = 1\n"); break;
    case 2:
        printf("n = 2\n"); break;
    case 3:
        printf("n = 3\n"); break;
    default:
        printf("n = ?\n");
    }
    return 0;
}
```

```
n = 2
```

1. Conditional Statements

Exercise

- Ask an integer to the user
- Display the month name according to the input number
- Manage wrong inputs

1. Conditional Statements

Questions

2. Iterative Statements

2. Iterative Statements

for

- Syntax:

```
for(counter init; condition; counter modifier) {

    //Sequence of instructions to be repeated

}
```

2. Iterative Statements

for

```c
#include <stdio.h>

int main()
{
    for(int i = 0; i <= 10; i++) {
        printf("7*%d=%d\n", i, 7*i);
    }
    return 0;
}
```

```
7*0=0
7*1=7
7*2=14
7*3=21
7*4=28
7*5=35
7*6=42
7*7=49
7*8=56
7*9=63
7*10=70
```

2. Iterative Statements

for

```c
#include <stdio.h>

int main()
{
    for(int i = 30; i >= 0; i-=2) {
        printf("%d\n", i);
    }
    return 0;
}
```

```
30
28
26
24
22
20
18
16
14
12
10
8
6
4
2
0
```

2. Iterative Statements

while

- Syntax:

```
while(condition) {

    //Sequence of instructions to be repeated

}
```

- As long as the condition is verified, it allows the repetition of the instructions

- Do not forget to modify the value of the expression in the instructions to be repeated, otherwise you will have an infinite number of iterations

2. Iterative Statements

while

```c
#include <stdio.h>

int main()
{
    int count = 0, i = 0;
    while(count < 10) {
        printf("Enter a number: ");
        scanf("%d", &i);
        count += i;
        fflush(stdin);
    }
    printf("Total: %d\n", count);
    return 0;
}
```

```
Enter a number: 3
Enter a number: 6
Enter a number: 22
Total: 31
```

2. Iterative Statements

do … while

- Syntax:

```
do {
    //Sequence of instructions to be repeated
} while(condition);
```

- In the **while** structure the condition is evaluated before the iteration and in the **do ... while** structure it is evaluated after; so, we always have at least one iteration with the **do ... while** structure

2. Iterative Statements

do … while

```c
#include <stdio.h>

int main()
{
    int count = 0, i = 0;
    do {
        printf("Enter a number: ");
        scanf("%d", &i);
        count += i;
        fflush(stdin);
    } while(count < 10);
    printf("Total: %d\n", count);
    return 0;
}
```

```
Enter a number: 3
Enter a number: 6
Enter a number: 22
Total: 31
```

2. Iterative Statements

do … while

```c
#include <stdio.h>

int main()
{
    int count = 14, i = 0;
    do {
        printf("Enter a number: ");
        scanf("%d", &i);
        count += i;
        fflush(stdin);
    } while(count < 10);
    printf("Total: %d\n", count);
    return 0;
}
```

```
Enter a number: 0
Total: 14
```

```c
#include <stdio.h>

int main()
{
    int count = 14, i = 0;
    while(count < 10) {
        printf("Enter a number: ");
        scanf("%d", &i);
        count += i;
        fflush(stdin);
    }
    printf("Total: %d\n", count);
    return 0;
}
```

```
Total: 14
```

2. Iterative Statements

Nested loops

```c
#include <stdio.h>

int main()
{
    int i, j;
    for(i = 0; i <= 10; i++) {
        printf("Table of %d\n", i);
        for(j = 0; j <= 10; j++) {
            printf("%d*%d=%d\n", i, j, i*j);
        }
    }
    return 0;
}
```

```
7*9=63
7*10=70
Table of 8
8*0=0
8*1=8
8*2=16
8*3=24
8*4=32
8*5=40
8*6=48
8*7=56
8*8=64
8*9=72
8*10=80
Table of 9
9*0=0
9*1=9
```

2. Iterative Statements

break and continue

- Use the **break** keyword to exit a loop and cause the early termination of its instructions

```c
#include <stdio.h>

int main()
{
    for(int i = 0; i <= 10; i++) {
        if(i == 3) {
            break;
        }
        printf("%d\n", i);
    }
    printf("Done!\n");
    return 0;
}
```

```
0
1
2
Done!
```

2. Iterative Statements

break and continue

- Use the **continue** keyword to cause a jump to the next iteration of the loop

```c
#include <stdio.h>

int main()
{
    for(int i = 0; i <= 10; i++) {
        if(i == 3) {
            continue;
        }
        printf("%d\n", i);
    }
    printf("Done!\n");
    return 0;
}
```

```
0
1
2
4
5
6
7
8
9
10
Done!
```

2. Iterative Statements

Bonus: emptying the input buffer

- Remember the following line:

```
fflush(stdin);
```

- The **scanf** function works with a buffer: the entered characters are first stored in a memory area, and are available for the **scanf** function only after pressing the “**Enter**” key

- The function leaves in the buffer the "**Enter**" character necessary for the validation of the input, and the next input will start with this character

2. Iterative Statements

Bonus: emptying the input buffer

```c
#include <stdio.h>

int main()
{
    int i = 0;
    char c;
    while(i < 5) {
        printf("Enter a char: ");
        scanf("%c", &c);
        printf("c%d: %c\n\n", i, c);
        i++;
    }
    return 0;
}
```

```
Enter a char: A
c0: A

Enter a char: c1:


Enter a char: B
c2: B

Enter a char: c3:


Enter a char: C
c4: C
```

2. Iterative Statements

Bonus: emptying the input buffer

- You must empty the input buffer!

```c
#include <stdio.h>

int main()
{
    int i = 0;
    char c;
    while(i < 5) {
        printf("Enter a char: ");
        scanf("%c", &c);
        printf("c%d: %c\n\n", i, c);
        fflush(stdin);
        i++;
    }
    return 0;
}
```

or

```c
while(getchar() != '\n');
```

```
Enter a char: A
c0: A

Enter a char: B
c1: B

Enter a char: C
c2: C

Enter a char: D
c3: D

Enter a char: E
c4: E
```

2. Iterative Statements

Exercise

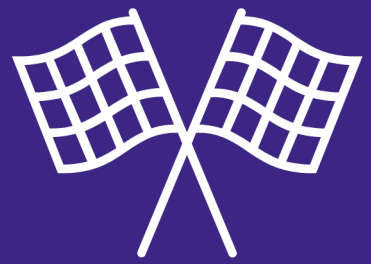
- Ask the user to enter several numbers
- Stop the input process when the user enters “**0**”
- Display the sum of the numbers

2. Iterative Statements

Questions

C Developer

Conditional and Iterative Statements



Thank you for your attention

SUPINFO