Computer Programming & Problem Solving
CS100

Mrs Sanga G. Chaki
Department of Computer Science and Engineering
National Institute of Technology, Goa
March, 2023

What are Control Instructions in C

1. Enable us to specify the order in which the various instructions in a program are to be executed.
2. Types of control instructions in C:
   a) Sequence Control Instruction
   b) Selection or Decision Control Instruction
   c) Repetition or Loop Control Instruction
   d) Case Control Instruction

The Decision Control Structure – why needed?

1. Need to alter our actions due to changing circumstances
2. C allows us to implement this using the if statement
3. **if (this condition is true)**

   execute this statement ;

if statement

if (this condition is true)

execute this statement ;

1. keyword **if** tells the compiler that what follows is a decision control instruction.
2. The condition following the keyword if is always enclosed within a pair of parentheses.
3. If the condition is true, then the statement is executed.
4. If the condition is not true then the statement is not executed; instead the program skips past it.

Relational Operators

1. How do we express the condition in C? And how do we evaluate ?
   - Using C's 'relational' operators.
2. The relational operators allow us to compare two values to see whether they are
   - equal to each other,
   - unequal,
   - one is greater than the other.
   - Lesser
   - Greater than equal
   - Lesser than equal

Relational Operators

this expression	is true if
x == y	x is equal to y
x != y	x is not equal to y
x < y	x is less than y
x > y	x is greater than y
x <= y	x is less than or equal to y
x >= y	x is greater than or equal to y

If statement: Code example

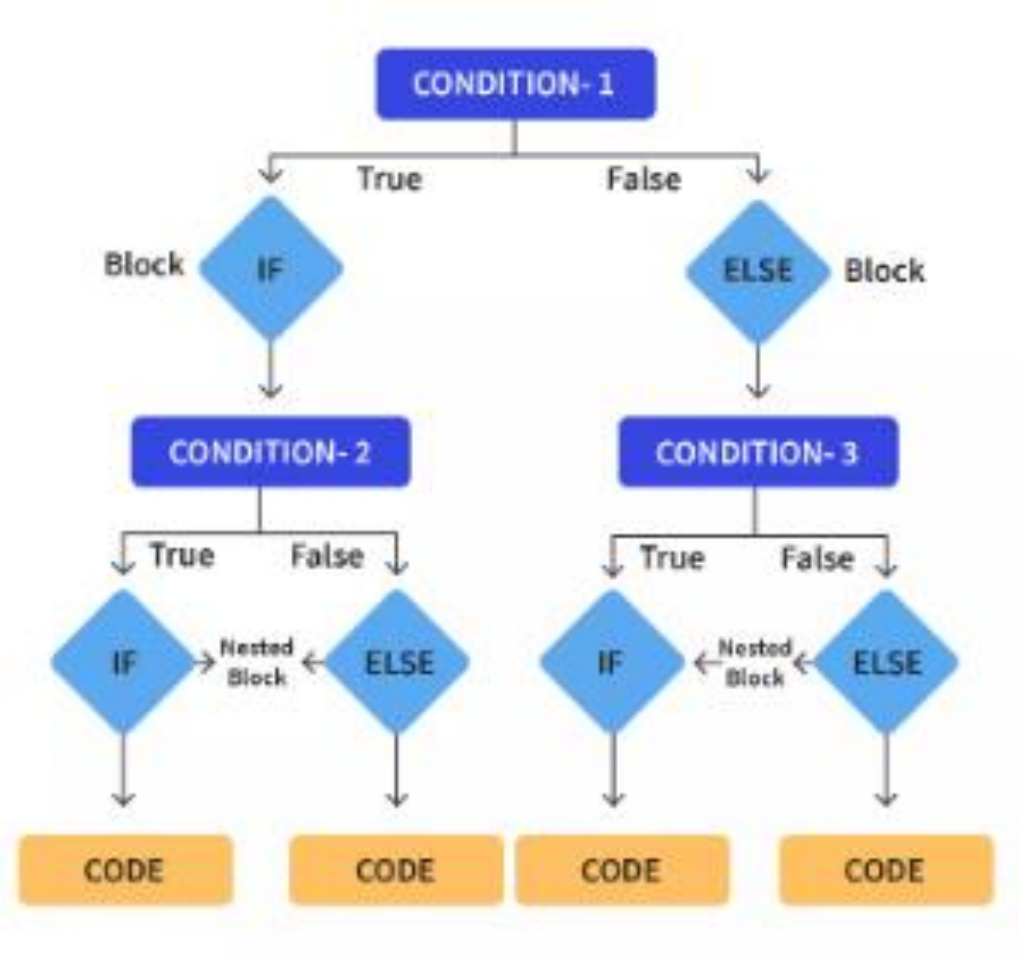
```c
/* Problem: Input an integer from user.
If it is >10, subtract 10 and print the output.*/
#include <stdio.h>

int main() {
    int input_num, output_num;

    printf("Enter an integer number: ");
    scanf("%d", &input_num);

    if(input_num>10)
    {
        output_num = input_num-10;
        printf("Output = %d", output_num);
    }
    printf("----END----");
    return 0;
}
```

if-else statement : Code example

```c
/* Problem: Input an integer from user. If it is >=10, subtract
    10 and print the output. Else add 10 and print the output*/
#include <stdio.h>
int main() {
    int input_num, output_num;
    printf("Enter an integer number: ");
    scanf("%d", &input_num);
    if(input_num>10){
        output_num = input_num-10;
        printf("Output = %d", output_num);
    }
    else{
        output_num = input_num+10;
        printf("Output = %d", output_num);
    }
    printf("\n----END----");
    return 0;
}
```

Nested if-else statement

1. Check a condition within a condition



Nested if-else statement: What is happening here?

```c
#include <stdio.h>
int main() {
    int marks;
    printf("Please enter marks: ");
    scanf("%d", &marks);

    if(marks>=90){
        if(marks>=95){
            printf("Grade A++");
        }
        else{
            printf("Grade A");
        }
        }
    else{
        printf("Grade B");
        }
    return 0;
}
```

Given this code, can you tell what will be the output if input is:

1. 91
2. 98
3. 70
4. 60.45

Logical Operators

1. AND (&&), OR(||) and NOT(!)
2. Allow two or more conditions to be combined in an if statement
3. Example: Enter marks, output grade following the given rules
   a) >50 - pass, otherwise fail
   b) 75-100 - Grade A
   c) 50-75 - Grade B
4. We can do this using if-else
5. Also using logical operators

Without Using Logical Operators

```c
#include <stdio.h>
int main() {
    int m;
    printf("enter marks: ");
    scanf("%d", &m);
    if(m>50){
        printf("Pass");
        if(m>75){
            printf("\nGrade A");
        }
        else
        printf("\nGrade B");
        }
    else
        printf("Fail");
return 0;
}
```

Using Logical Operators

```c
#include <stdio.h>
int main() {
    int m;
    printf("enter marks: ");
    scanf("%d", &m);
    if(m>50 && m<=75)
        printf("Pass \nGrade B");
    else if(m>75 && m<=100)
        printf("Pass \nGrade A");
    else
        printf("Fail");
    return 0;
}
```

NOT – Unary operator

1. NOT- This operator reverses the result of the expression it operates on.
2. What do you think is the output of the given code?

```c
#include <stdio.h>
int main() {
    int flag = 0;
    if(flag == 0)
        printf("\nFlag is zero");
    if(!flag)
        printf("\nFlag is zero");
return 0;
}
```

Conditional Operators – Ternary Operator

1. This makes use of an expression that is either true or false. An appropriate value is selected, depending on the outcome of the logical expression.
2. General form: **expression 1 ? expression 2 : expression 3**
3. Example:

interest = (balance>5000) ? balance*0.2 : balance*0.1;

This is equivalent to:

```
if (balance > 5000)
    interest = balance * 0.2;
else
    interest = balance * 0.1;
```

Conditional Operators – Ternary Operator

1. Example:

```
int x, y ;
scanf ( "%d", &x ) ;
y = ( x > 5 ? 3 : 4 ) ;
```

Equivalent to:

```
if ( x > 5 )
    y = 3 ;
else
    y = 4 ;
```

1. Example:

```
if (marks >= 60)
    printf("Passed \n");
else
    printf("Failed \n");
```

Is equivalent to:

(marks >= 60) ? printf("Passed\n") : printf("Failed\n");