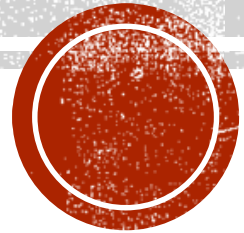
CONTROL STATEMENTS

Muhammad Ali Nayeem

Student of Knowledge

http://teacher.buet.ac.bd/ali_nayeem/



RECAP

- We have already learned 2 types of statements
  - Declaration statement
  - Assignment statements
- Insha'Allah we will learn a new type of statement today
  - Control Statement

INTRODUCTION

- Control Statements determine your program's flow of execution
  - Order of execution of statements in a program
- Default order ?
- Often we need to alter/change order
  - When?

`if` STATEMENT

- **General form** `if(expression)`
  `statement;`
- A new keyword
- *expression*:
  - any valid C expression
  - know as target

```
if ( 3 + 2 % 5 )
    printf ( "This works" ) ;

if ( a = 10 )
    printf ( "Even this works" )  ;

if ( -5 )
    printf ( "Surprisingly even this works" ) ;
```

- If *expression* is **non-zero** *statement* will be executed
- If *expression* is **zero** *statement* will be bypassed/skipped
- Normally *expression* consists of **relational & logical** operator

RELATIONAL OPERATORS

- Allow us to compare two values

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

EXAMPLE

```
#include<stdio.h>
main()
{
    int num;
    scanf("%d", &num);
    if(num>=0)
       printf("num is positive");/*if(num>-1)*/
    if(num<0)
       printf("num is negative");
}
```

COMMON PROGRAMMING ERROR!!!

- Placing ; (semicolon) immediately after condition in **if**
  - if(*expression*); *statement*;
- Confusing equality operator (==) with assignment operator (=)
  - if(a=b)
    - The assignment operators return the value of the variable specified by the left operand after the assignment
    - The resultant type is the type of the left operand
  - if(a=5)

ADD `else`

- `if` does nothing when the expression evaluates to false
- Can we execute one statement if the expression evaluates to true and another statement if the expression evaluates to false?
- Of course! This is what is the purpose of the else statement

```
if(expression)
    statement1;
else
    statement2;
```

  - If *expression* is **true** *statement1* will be evaluated and *statement1* will be skipped
  - If *expression* is **false** *statement1* will be bypassed and *statement2* will be executed
- Under no circumstances both the statements will execute
  - Mutually exclusive
- **else** part is optional

IF-ELSE EXAMPLE

```
#include<stdio.h>
main()
{
    int num;
    scanf("%d", &num);
    if(num>=0)
        printf("num is positive");//if(num>-1)
    else
        printf("num is negative");
}
```

BLOCK OF CODE

- Statements enclosed within { }
- Group two or more statements into one unit
- Can be used anywhere a single statement can
- Common programming error:
  - Forgetting braces of **compound** statements/blocks

BLOCK OF CODE

- ```
  if(expression)
  {
      statement1;
      statement2;
          ⋮
      statementN;
   }
  else
  {
      statement1;
      statement2;
          ⋮
      statementN;
   }
  ```
- If *expression* is **true** all the statements with if will be executed
- If *expression* is **false** all the statements with else will be executed


  ```

LOGICAL OPERATOR

- Connect together true/false results
- ‘&&’ logical AND binary
- ‘||’ logical OR binary
- ‘!’ logical NOT unary

p	q	p && q	p || q	!p
0	0	0	0	1
0	1	0	1	1
1	1	1	1	0
1	0	0	1	0

NESTED `if-else`

- It is perfectly all right if we write an entire if-else construct within
  - either the body of the if statement
  - or the body of an else statement.
- This is called ‘nesting’of ifs

```
#include<stdio.h>
main()
{
 int id;
 printf("Please enter last 3 digits of your id:\n");
 scanf("%d", &id);
 if(id>130 && id<196)
 {
    if(id<163)
     printf("You are in C1\n");
    else
     printf("You are in C2\n");
 }
 else
 {
    printf("You are in A or B\n");
 }
}
```

FORMS OF `if`

(a)
```
if ( condition )
    do this ;
```

(b)
```
if ( condition )
{
    do this ;
    and this ;
}
```

(c)
```
if ( condition )
    do this ;
else
    do this ;
```

(d)
```
if ( condition )
{
    do this ;
    and this ;
}
else
{
    do this ;
    and this ;
}
```

(e)

```
if ( condition )
    do this ;
else
{
    if ( condition )
        do this ;
    else
    {
        do this ;
        and this ;
    }
}
```

(f)

```
if ( condition )
{
    if ( condition )
        do this ;
    else
    {
        do this ;
        and this ;
    }
}
else
    do this ;
```

if-else if STATEMENT

- ```
  if(expression)
      statement;
  else if (expression)
      statement;
  else if (expression)
      statement;
  else
      statement;
  ```


  ```

if-else if STATEMENT

- Multi-way decision
- *expressions* are evaluated in order
- If any *expression is true*
  - the *statement* associated with it is executed
    - Multiple *statements* can be associated using { }
  - the whole chain is terminated
- If none of the *expressions* are true
  - **else** part is executed
  - Handles none of the above/default case
  - Optional

A PRACTICE PROBLEM

- Input: The marks obtained by a student in 5 different subjects
  - Use scanf( ) to get 5 numbers from the keyboard.
- Find $Average$ of the 5 marks
- Output: The grade of the student as per the following rules:
  - $Average \geq 60 \rightarrow$ **A**
  - $50 \leq Average < 60 \rightarrow$ **B**
  - $40 \leq Average < 50 \rightarrow$ **C**
  - $Average < 40 \rightarrow$ Fail
- Use only `if` statement

PRECEDENCE/PRIORITY AND ASSOCIATIVITY OF OPERATORS

- Operators on the same line have the same precedence
- Rows are in order of decreasing precedence
- Unary & +, -, and * have higher precedence than the binary forms

Operators	Associativity
`() [] -> .`	left to right
`! ~ ++ -- + - * (type) sizeof`	right to left
`* / %`	left to right
`+ -`	left to right
`<< >>`	left to right
`< <= > >=`	left to right
`== !=`	left to right
`&`	left to right
`^`	left to right
`|`	left to right
`&&`	left to right
`||`	left to right
`?:`	right to left
`= += -= *= /= %= &= ^= |= <<= >>=`	right to left
`,`	left to right

THANKS TO

- Johra Muhammad Moosa
  - Lecturer
  - Department of Computer Science & Engineering
  - Bangladesh University of Engineering & Technology