# Introduction to Programming

Lecture 6:

## Making Decisions

# What We Will Learn

➢ Introduction

➢ Conditions and Boolean operations

➢ `if-else` statement

➢ `switch-case` statement

➢ Conditional expressions

# What We Will Learn

➤ Introduction

➤ Conditions and Boolean operations

➤ `if-else` statement

➤ `switch-case` statement

➤ Conditional expressions

# Decision

➢ Decisions are based on *conditions*

  ➢ Read the "C programming language"

    ➢ Next week you have exam :-O ;-)

➢ In programming

  ➢ Do statements based on conditions

    ➢ True → The statements will be done

    ➢ False → The statement wont be done

# Conditions

➢ Conditions by comparisons; e.g.,

 ➢ If a is greater then b

 ➢ If c equals to d

➢ Comparing numbers: Relational Operators

| Relational Operator | Meaning |
|---|---|
| < | is less than |
| <= | is less than or equal to |
| > | is greater than |
| >= | is greater than or equal to |
| == | is equal to |
| != | is not equal to |

# Relations

➢ Relations are <span style="color:red">not</span> a complete statement

```
int a, b;

a == b;        //ERROR

a <= b;        //ERROR
```

➢ Relations produce a boolean value

```
int a, b;

bool bl;                // #include <stdbool.h>

bl = a == b;

bl = a <= b;
```

# Boolean operations

➤ Multiple conditions in decision making

➤ Logical relation between conditions

   ➤ if you are student and you have the programming course

      → You should read the book

➤ C Boolean operators

   ➤ and     `&&`

   ➤ or       `||`

   ➤ not     `!`

| p | q | p && q | p \|\| q | !p |
|---|---|--------|---------|-----|
| False | False | False | False | True |
| False | True | False | True | True |
| True | False | False | True | False |
| True | True | True | True | False |

# Boolean operations (cont'd)

➢ Examples

```
bool a = true, b = false, c;
c = !a;           // c = false
c = a && b;       // c = false
c = a || b;       // c = true
c = !a || b;      // c = false
```

# Precedence

| Operator | Operation |
|---|---|
| ++ -- | increment, decrement |
| + - | unary plus, minus |
| ! | boolean not |
| (<type>) | cast to <type> |
| * / % | multiplication, division, remainder |
| + - | addition/concatenation, subtraction |
| < <= > >= | relational ordering |
| == != | relational equality, inequality |
| && | boolean and |
| \|\| | boolean or |
| = += -= *= /= %= | assignments |

# Relations, No type effect

```
int a = 10, b = 20;
float f = 54.677f;
double d = 547.775;
char c1 = 'A', c2 = 'a';
bool b1;
b1 = a == f;              // false
b1 = a <= d + 5;          // true
b1 = d < c1 * 10;         // true
b1 = c1 == c2;            // false
b1 = '1' < '2';           // true
b1 = c1 + f < d + a;      // true
```

# Casting

➤ **In logical operations**
  - ➤ 0 → False,  non-zero → True

➤ **In mathematical & comparison operations**
  - ➤ False → 0 , True → 1

```
bool b1, b2;

int i = 0, j = 20;

b1 = i && j;                    // b1 = false

b2 = j || j;                    // b2 = true

i = b1 + b2;                    // i = 1

j = (i < j) + (b1 && b2);       // j = 1
```

# Examples

➤ x ∈ [10 , 20]

➤ Wrong Version

  ➤ 10 <= x <= 20

  ➤ Let x = 30

    ➤ 10 <= 30 <=20 → (10 <= 30) <= 20

                  → true <= 20 → 1 <= 20 → true!!!

➤ Correct Version

  ➤ (10 <= x) && (x <= 20)

  ➤ Let x = 30

    ➤ (10 <= 30) && (30 <= 20) → true && false → false

# Examples

➢ a,b > 0

➢ Wrong version
   ➢ a && b > 0
   ➢ Let a = -10, b = 20
      ➢ -10 && 20 > 0 → -10 && (20 > 0)
      → -10 && true → true && true → true !!!

➢ Correct version
   ➢ (a > 0) && (b > 0)
   ➢ Let a = -10, b = 20
      ➢ (-10 > 0) && (20 > 0) → false && true → false

13

# Lazy evaluation

➢When final result is found, does not evaluate remaining

```
int i;

bool a = true, b = false, c = true;

bool d = a || b || c

bool d = b && (a || c)

bool d = (i > 0) && (sqrt(i) > 5.6)
```

# What We Will Learn

➢Introduction

➢Conditions and Boolean operations

➢**`if-else`** statement

➢`switch-case` statement

➢Conditional expressions

# Type of statements

➢Expression statement (دستور عبارتی)

  ➢ Single statements

  ```
  x = y + 10; scanf("%d", &i);
  ```

➢Control statement (دستور کنترلی)

  ➢ Control the flow of program

    ➢ Decisions and loops

➢Compound statement (دستور مرکب)

  ➢ Starts with { and ends with }

  ➢ All statements can be between { and }

# if statement

➤ **Decision making in C**

```
if( <expression> )

    <statements1>

else

    <statements2>
```
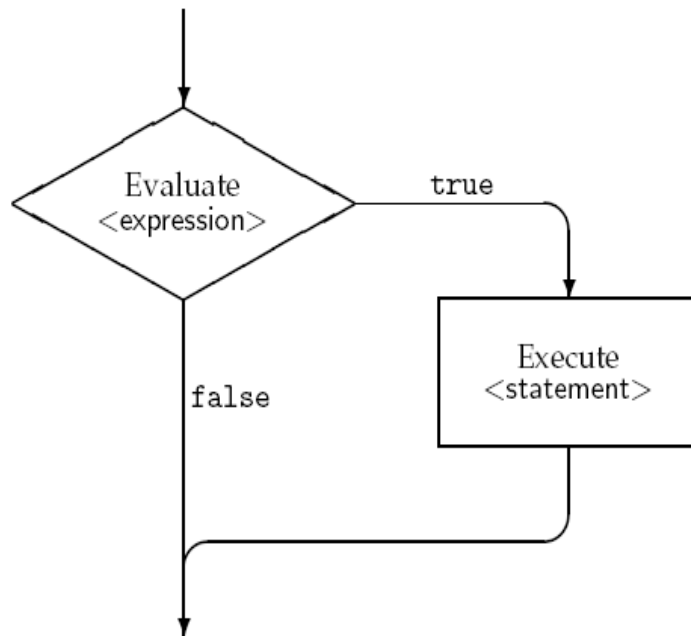
➤ **Expression**

- ➤ A boolean statement: `a <= b`
- ➤ A mathematical statement: `a + b` or a variable: `a`
  - ➤ zero → false
  - ➤ Non-zero → true

# Flowchart

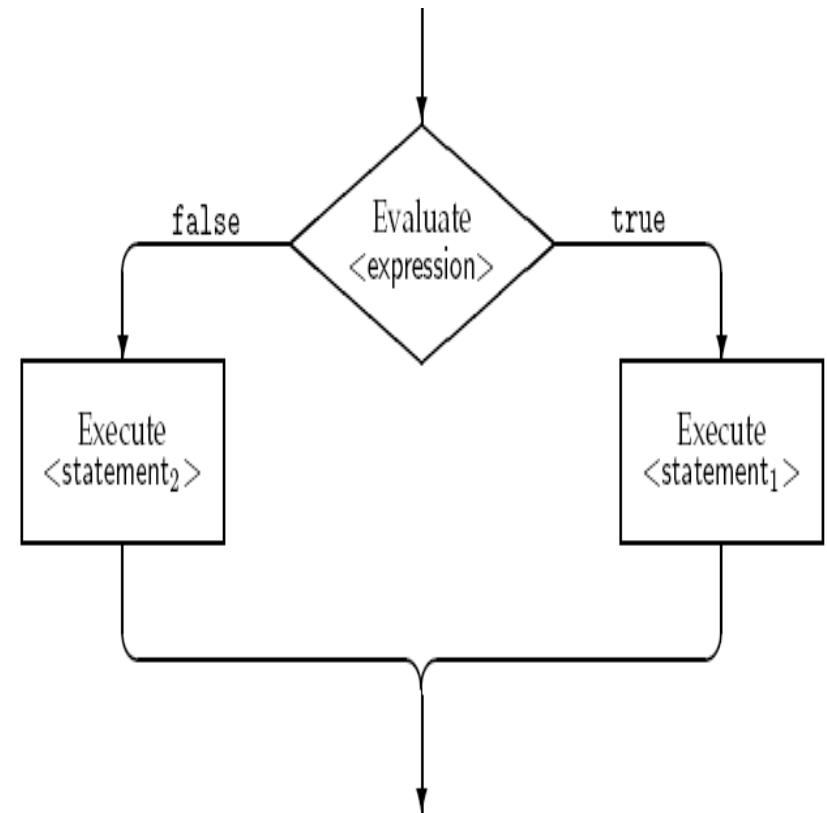if(<expression>)

    <statement>

if(<expression>)

    <statement1>

else

    <statement2>

برنامه‌ای که یک عدد را از کاربر می‌گیرد و مشخص می‌کند که این عدد فرد است یا زوج

```c
#include <stdio.h>
int main(void){
    int number_to_test, remainder;

    printf("Enter your number to be tested: ");
    scanf("%d", &number_to_test);

    remainder = number_to_test % 2;
    if(remainder == 0)
        printf ("The number is even.\n");
    else
        printf ("The number is odd.\n");

    return 0;
}
```

# Statements in if-else

➤ **Empty statement**

```
if(a > b)
   Printf("A is larger \n");

else
   ;
```

➤ **Block statements**

```
if(a <= b){
   printf("A is less than b or ");
   printf("A is equal b\n");

}

else
   printf("A is greater than b\n");
```

```c
#include <stdio.h>

int main(void){
    int i;
    char c;

    printf("Enter a char: ");
    scanf(" %c", &c);
    printf("Enter an int: ");
    scanf("%d", &i);

    if(i > 0)
        printf("Your number is larger than 0\n");
    else
        printf("Your number is less than or equal 0\n");

    if((c >= '0') && (c <= '9'))
        printf("Your char is Numeric \n");
    return 0;
}
```

برنامه‌ای که یک حرف و یک عدد را می‌گیرد. در مورد عدد مشخص می‌کند که آیا بزرگتر صفر است یا نه. در مورد حرف اگر حرف عددی باشد پیغام چاپ می‌کند.

# More than two choices

➤ If statement: 2 choices

  ➢ If conditions are true → if statements

  ➢ If conditions are false → else statements


➤ How to make decisions when there are multiple choices?

# Map numeric grade to alphabetic

```
int numg;
char alphag;
if((numg < 25)
  alphag = 'D';
if((numg >= 25) && (numg < 50))
  alphag = 'C';
if((numg >= 50) && (numg < 75))
  alphag = 'B';
if(numg >= 75)
  alphag = 'A';
```

# More than two choices

➢ To avoid repeating conditions in if statements

➢ To avoid running unnecessary statements

➢ Nested if: check multiple conditions

  ➢ \<Statements 1\> becomes an if-else statement

  ➢ \<Statements 2\> becomes an if-else statement

  ➢ Repeat it as many as needed

# Nested if

```
if(<condition 1>)

   <statement 1>

else{

   if(<condition 2>)

      <statement 2>

   else

      <statement 3>

}
```

```
if(<condition 1>)

   <statement 1>

else if(<condition 2>)

      <statement 2>

else

      <statement 3>
```

# Map numeric grade to alphabetic

```
int numg;
char alphag;
if(numg < 25)
   alphag = 'D';
else{
   if(numg < 50)
      alphag = 'C';
   else{
      if(numg < 75)
            alphag = 'B';
      else
            alphag = 'A'
   }
}
```

# Map numeric grade to alphabetic

```
int numg;
char alphag;
if(numg < 25)
  alphag = 'D';
else if(numg < 50)
  alphag = 'C';
else if(numg < 75)
  alphag = 'B';
else
  alphag = 'A';
```

# Map numeric grade to alphabetic

```
int numg;
char alphag;
if(numg < 50){
    if(numg < 25)
        alphag = 'D';
    else
        alphag = 'C';
}
else{
    if(numg < 75)
        alphag = 'B';
    else
        alphag = 'A';
}
```

# Nested if: Example 2

> Determine a char is alphabetic, Uppercase or not, numeric, less or greater than 5 or none of them

```
/* '0': 48, '9': 57, 'A': 65, 'Z': 90, 'a': 97, 'z': 122 */

char c;

if(((c >= 'a') && (c <= 'z')) || ((c >= 'A') && (c <= 'Z'))){

    if(c >= 'a')

        printf("The char is Lowercase \n");

    else

        printf("The char is Uppercase \n");

}
else if((c >= '0') && (c <= '9')){

    if(c > '5')

        printf("The char is greater than 5\n");

    else

        printf("The char is less than or equal 5\n");

}
else

    printf("The char is not either alphabetic or numeric");
```

This can be written in other ways

# Equivalent if-else of Nested if-else

```
if(c1)
  if(c2)
    s1
  else
    s2
else
  if(c3)
    s3
  else
    s4
```

```
if(c1 && c2)
  s1
if (c1 && !(c2))
  s2
if (!(c1) && c3)
  s3
if (!(c1) && !(c3))
  s4
```

# Nested if: Incomplete branch

➢ 1) **else** part is optional

➢ 2) **else** always associates with the nearest **if**
  ➢ 1 + 2 can be dangerous specially in incomplete branchs

➢ Example: Tell user to move or game over

```
if(gameIsOver == 0)

    if(playerToMove == YOU)

        printf ("Your Move\n");

else

    printf ("The game is over\n");
```

➢ To avoid error you should
  ➢ Close off you code or Use Empty statements

# Nested if: close off & empty statement

```
if(gameIsOver == 0){
    if(playerToMove == YOU)
        printf ("Your Move\n");
}
else
    printf ("The game is over\n");
//--------------------------------
if(gameIsOver == 0)
    if(playerToMove == YOU)
        printf ("Your Move\n");
    else
        ;
else
    printf ("The game is over\n");
```

This one is better.

# Print in base-2, 0 <= input < 16

# Duplicate zero, input is 3 digit

# What We Will Learn

➢ Introduction

➢ Conditions and Boolean operations

➢ `if-else` statement

➢ **switch-case statement**

➢ Conditional expressions

# switch-case: Multiple choices

➢ **Multiple conditions**
  ➢ If-else if-else if-….

➢ **Select from alternative values of a variable**
  ➢ switch-case
  ➢ Values should be constant not expression: `i, i+j,`
  ➢ Values & Variables should be `int` or `char`

```
switch(variable){
    case value1:
            <statements 1>
    case value2:
            <statements 2>
}
```

# How does switch-case work?

➤ Each switch-case can be rewritten by If-else
  ➤ if-else version of switch-case in the previous slide

```
if(variable == value1)}
  <statements 1>
  <statements 2>
}
else if(variable == value2){
  <statements 2>
}
```

# switch-case: complete version

```
switch(variable){
    case value1:
        <statements 1>
        break;
    case value2:
        <statements 2>
        break;
    default:
        <statements 3>
}
```

```
If(variable == value1)
{
    <statements 1>
}
else if(variable == value2)
{
    <statements 2>
}
else{
    <statements 3>
}
```

برنامه‌ای که دو عدد و یک عملگر را می‌گیرد، عملگر را بر روی اعداد اعمال و نتیجه را چاپ می‌کند.

```c
#include <stdio.h>
#include <stdlib.h>
int main(void){
    int res, opd1, opd2;
    char opr;
    printf("Operand1 : ");
    scanf("%d", &opd1);
    printf("Operand2 : ");
    scanf("%d", &opd2);
    printf("Operator : ");
    scanf(" %c", &opr);
    switch(opr){
    case '+':
        res = opd1 + opd2;
        break;
```

```c
case '-':
    res = opd1 - opd2;
    break;
case '/':
    res = opd1 / opd2;
    break;
case '*':
    res = opd1 * opd2;
    break;
default:
    printf("Invalid operator \n");
    return -1;
}
printf("%d %c %d = %d\n", opd1, opr, opd2, res);
return 0;
}
```

# switch-case (cont'd)

➢ All values used in case should be different

```
switch(i){        //Error

case 1:

…

case 2:

…

case 1:

…
```

# switch-case (cont'd)

➢ All values must be value, not expression of variables

```
switch(i){        //Error

case j:

…

case 2:

…

case k+10:
```

# switch-case: multiple matches

```
switch(variable){

   case value1:

   case value2:

      <statements 1>

      break;

   case value3:

      <statements 2>

}
```

```
If(

(variable == value1) ||

(variable == value2)

){

   <statements 1>

}
else if

(variable == value3)

{

   <statements 2>

}
```

# switch-case vs. if-else

➢ **if-else** is more powerful than **switch-case**

➢ **switch-case** is only for checking the values of a variable and the values must be constant

 ➢ Some if-else cannot be rewritten by switch-case

```
double var1, var2;
if(var1 <= 1.1)
  <statements 1>

if(var1 == var2)
  <statements 2>
```

# Nested switch-case

```
bool b;   //b = x && y
switch (x){
  case 0:
     b = 0;
     break;
  case 1:
     switch(y){
          case 0:
              b = 0;
              break;
          case 1:
              b = 1;
              break;
     }
  break;
}
```

# What We Will Learn

➢Introduction

➢Conditions and Boolean operations

➢`if-else` statement

➢`switch-case` statement

➢Conditional expressions

# Conditional Expression

➢Assign value according to conditions

➢A ternary (سه تایی) operator

```
int i, j, k;
bool b;
...
i = b ? j : k;          /* if(b)
                          *        i = j
                          * else
                          *        i = k;
                          */
```

# Conditional Expression: Examples

$$y = abs(x)$$

```
y = (x > 0) ? x : -x;
```

$$\text{signum}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases}$$

```
signum = (x < 0) ? -1 : (x > 0 ? 1 : 0)
```

# Map Alphabetic Grade to Numeric

int d = numg / 25

charg = (d == 0) ? 'D' : ((d == 1) ? 'C' : (d == 2) ? 'B' : 'A');

# Common Bugs

➤ Equality of floating point numbers
  ➤ Two float numbers may or may <span style="color:red">NOT</span> be equal

```
double d1, d2;
d1 = 1e20 + 1;
d2 = 1e20 - 1;
if(d1 == d2)
    printf("They are equal :-o \n");
else
    printf("They are not equal :D \n");
```

<span style="color:red">They are equal :-o</span>

# Common Bugs

➢ Danger of empty statement

➢ Danger of assignment (=) and equality (==)

```
int a = 10;

int b = 20;

if(a=b) // Logical Error
```

➢ Danger of similarity between C and mathematic

➢ `if(a < b < c)     // Logical Error`

➢ `if(a && b > 0)    // Logical Error`

# Avoiding Bugs

➤ Precedence of operators

`if(!a && b)`　　　or　　　`if(!(a && b))`

➤ Use parenthesis in conditions

➤ Close-off code as much as you can

# Reference

➢ Reading Assignment: Chapter 3 of "C How to Program"

# Homework

➢Homework 3