jgromero@inf.uc3m.es

Universidad
Carlos III de Madrid
www.uc3m.es

Lesson 4
Control Flow and Loops

Programming

Grade in Industrial Technology Engineering

Outline

1. Introduction
2. Conditional instructions
   a. `if else`
   b. `switch`
3. Loop instructions
   a. `while`
   b. `do while`
   c. `for`

Outline

1. **Introduction**
2. Conditional instructions
   a. `if else`
   b. `switch`
3. Loop instructions
   a. `while`
   b. `do while`
   c. `for`

Instructions are executed from top to bottom

- In the order that they appear
- Starting from the first instruction of the `main` method

Control flow instructions break up this sequence

- **Conditional instructions / Decision-making instructions**
  - Blocks of instructions are executed depending on the result of a *boolean* expression (the condition)
- **Loop instructions**
  - Blocks of instructions are repeated while a condition holds

Good practices for developing good programs

Never use "**go to**" instructions!

Any algorithm can be implemented by using only these three structures:

- Sequential
- Conditional
- Loop

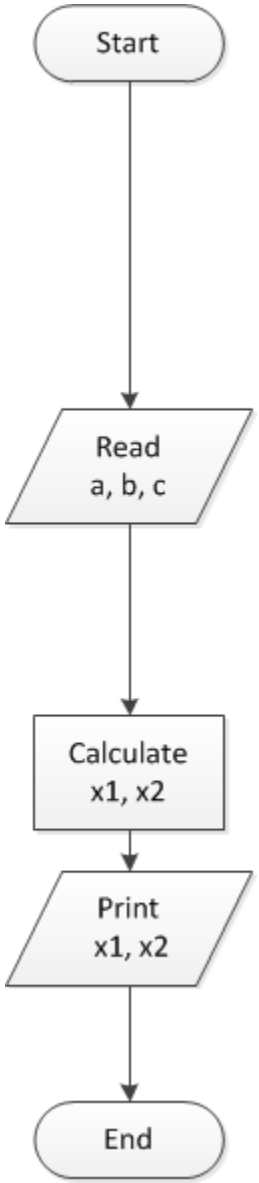
Outline

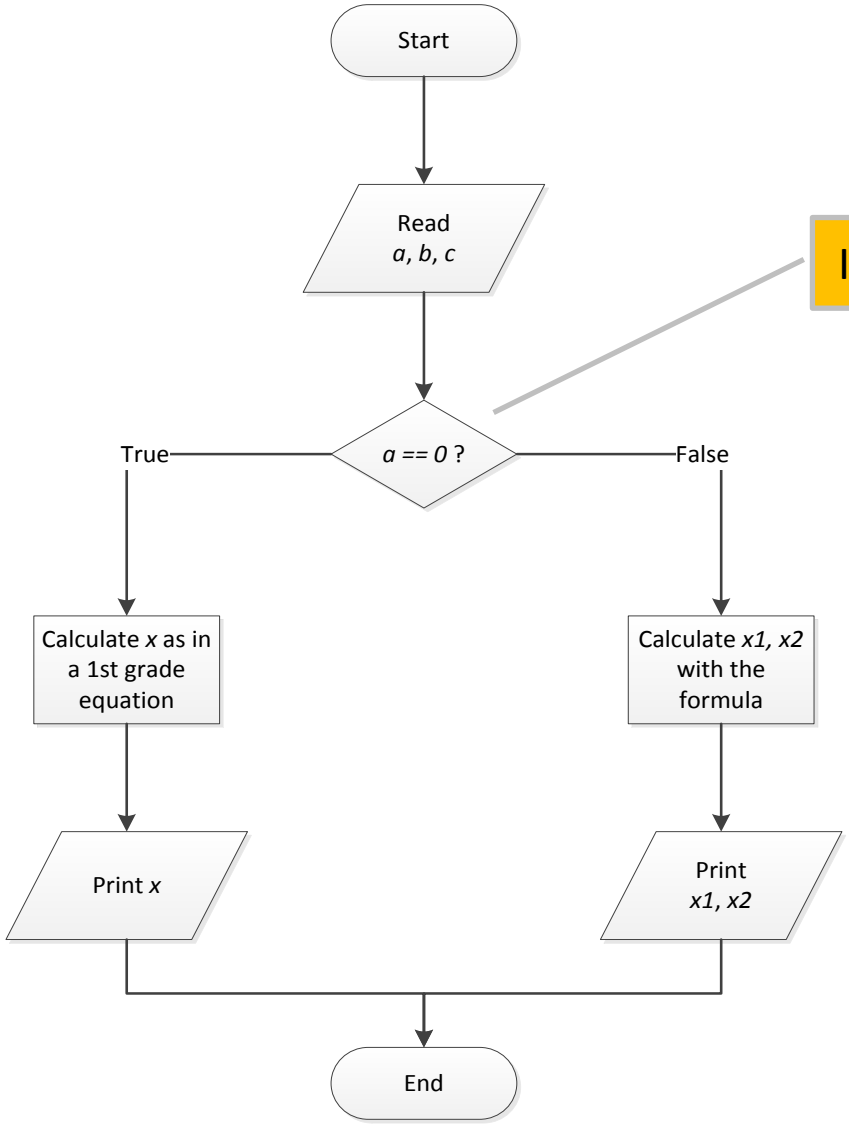
1. Introduction
2. **Conditional instructions**
   a. `if else`
   b. `switch`
3. Loop instructions
   a. `while`
   b. `do while`
   c. `for`

2.1. if else

EquationBasic.c

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

int main()
{
  /* 1. Variable declaration */
  float a, b, c;
  float x1, x2;

  /* 2. Read values from the keyboard */
  printf("Solving a second grade equation...\n");

  printf("Introduce 'a' value: ");
  scanf("%f", &a);

  printf("Introduce 'b' value: ");
  scanf("%f", &b);

  printf("Introduce 'c' value: ");
  scanf("%f", &c);

  /* 3. Calculate solution */
  x1 = (-b + sqrt(b*b - 4*a*c) ) / (2*a) ;
  x2 = (-b - sqrt(b*b - 4*a*c) ) / (2*a) ;

  /* 4. Print solutions */
  printf("\nEquation: %fx^2 + %fx - %f = 0", a, b, c);
  printf("\nx1 = %f", x1);
  printf("\nx2 = %f", x2);

  /* Stop the program and end 'main' function */
  printf("\n\n");
  system("PAUSE");
  return 0;
}
```





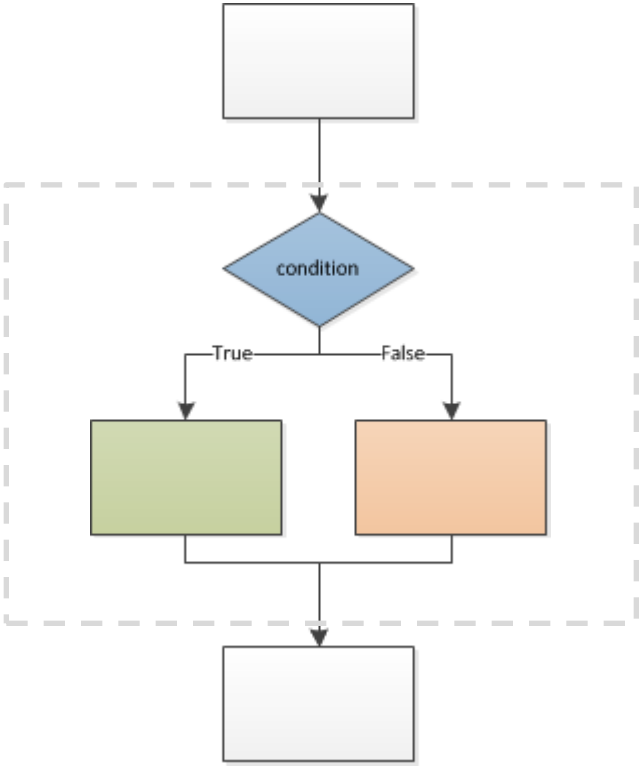
Is it a 2nd grade equation?

2. Conditional instructions
`if else`

The most basic control flow instruction

If the condition is **true**, the block of code associated to the **`if`** part is executed

If the condition is **false**, the block of code associated to the **`else`** part is executed




```
if (<boolean expression>) {
  <statement(s)>
} else {
  <statement(s)>
}
```

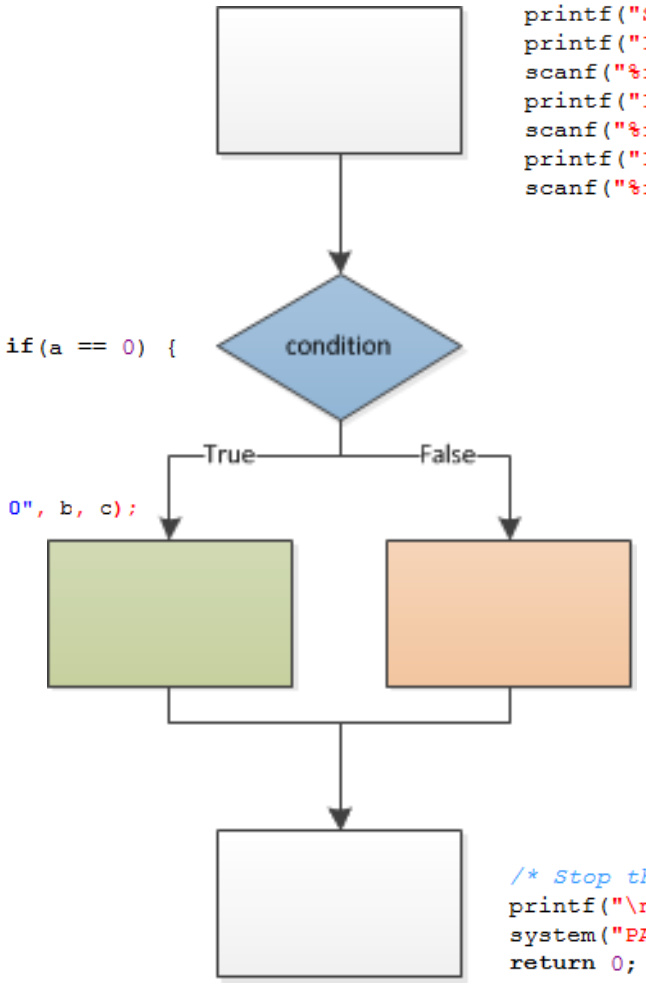
2. Conditional instructions
`if else`

```c
/* Test 'a' value */
if(a == 0) {
  /* First grade equation*/
  printf("1st grade equation: %fx + %f = 0", b, c);
  x = -c / b;

  /* Print solutions */
  printf("\nx = %f", x);

} else {
  /* Second grade equation */
  printf("2nd grade equation: %fx^2 + %fx + %f = 0", a, b, c);
  x1 = (-b + sqrt(b*b - 4*a*c) ) / (2*a) ;
  x2 = (-b - sqrt(b*b - 4*a*c) ) / (2*a) ;

  /* Print solutions */
  printf("\nx1 = %f", x1);
  printf("\nx2 = %f", x2);
}
```

Is it a 2nd grade equation?

2. Conditional instructions

```
/* Read values from the keyboard */
printf("Solving a second grade equation...\n");
printf("Introduce 'a' value: ");
scanf("%f", &a);
printf("Introduce 'b' value: ");
scanf("%f", &b);
printf("Introduce 'c' value: ");
scanf("%f", &c);
```



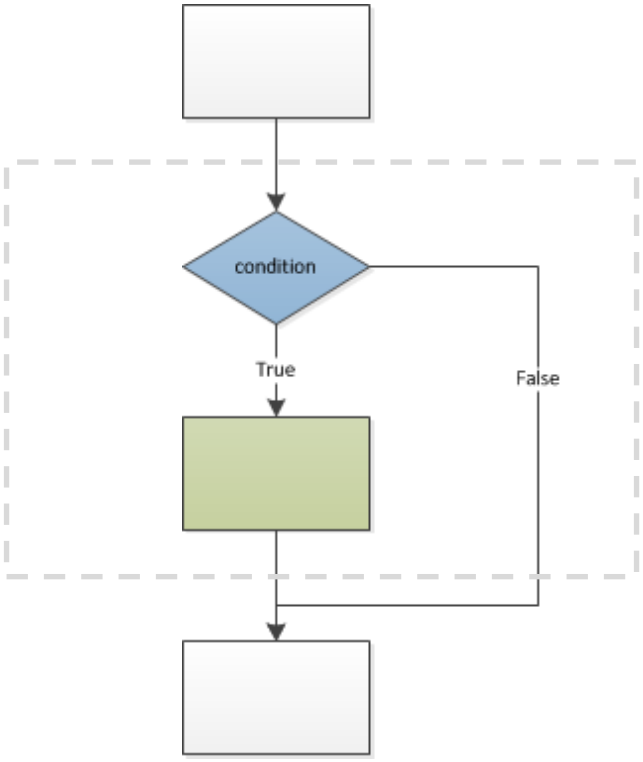

```
/* First grade equation*/
printf("1st grade equation: %fx + %f = 0", b, c);
x = -c / b;

/* Print solutions */
printf("\nx = %f", x);
```

```
} else {
  /* Second grade equation */
  printf("2nd grade equation: %fx^2 + %fx + %f = 0", a, b
  x1 = (-b + sqrt(b*b - 4*a*c) ) / (2*a) ;
  x2 = (-b - sqrt(b*b - 4*a*c) ) / (2*a) ;

  /* Print solutions */
  printf("\nx1 = %f", x1);
  printf("\nx2 = %f", x2);
}
```

```
/* Stop the program and end 'main' function */
printf("\n\n");
system("PAUSE");
return 0;
```

2. Conditional instructions
if else

The **else** part is optional




```
if (<boolean expression>) {
  <statement(s)>
}
```

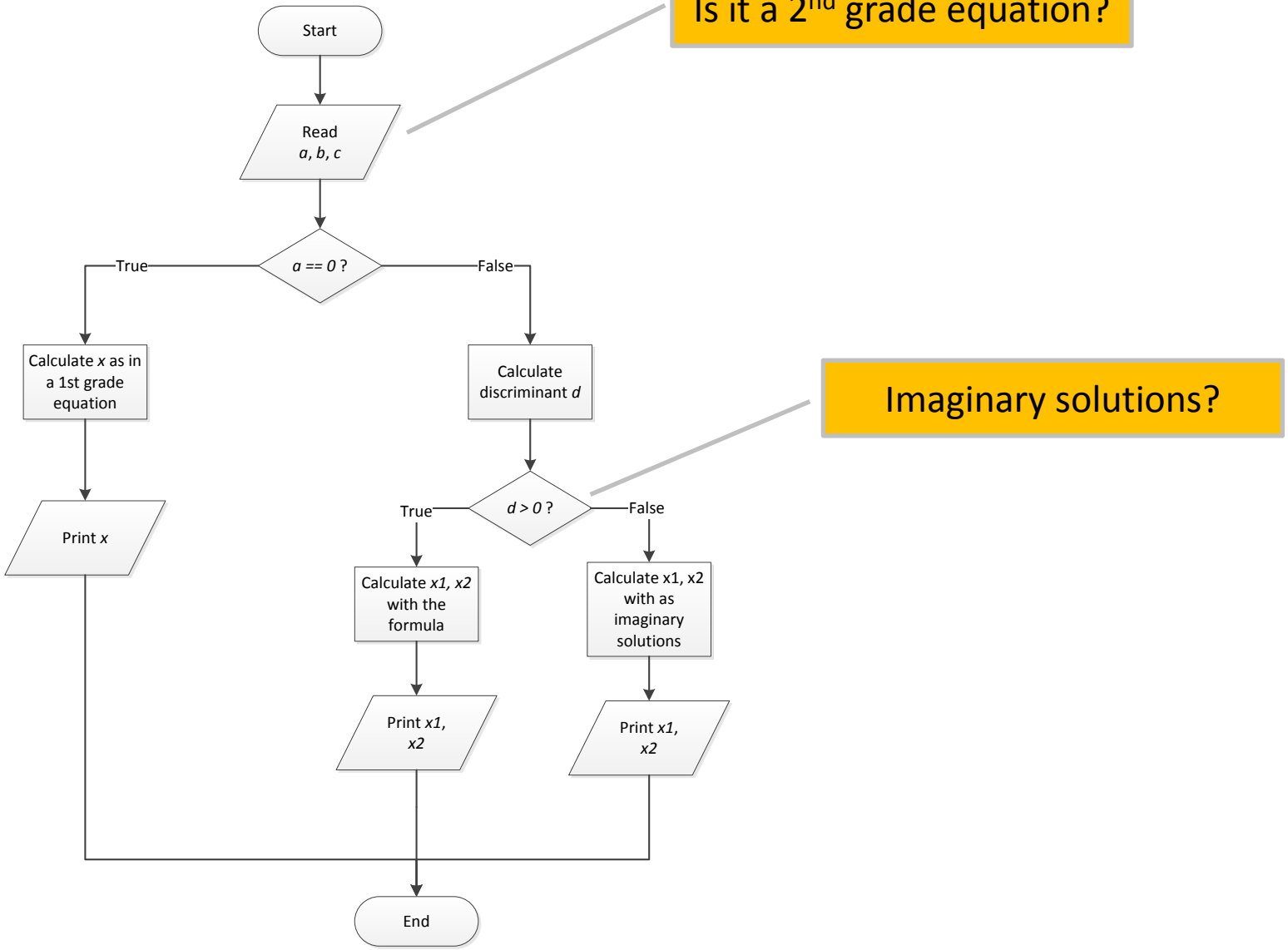
If there is only one instruction inside the block, the **braces** can be removed

```
if (<boolean expression>)
  <statement>
```

```
if (<boolean expression>)
  <statement>
else
  <statement>
```

Do not forget the ;

if else instructions can be **nested**


Start
Read
a, b, c
a == 0 ?
True
False
Calculate x as in a 1st grade equation
Print x
Calculate discriminant d
d > 0 ?
True
False
Calculate x1, x2 with the formula
Print x1, x2
Calculate x1, x2 with as imaginary solutions
Print x1, x2
End
Is it a 2nd grade equation?
Imaginary solutions?

```c
/* Test 'a' value */
if(a == 0) {
  /* First grade equation*/
  printf("1st grade equation: %fx + %f = 0", b, c);
  x = -c / b;

  /* Print solutions */
  printf("\nx = %f", x);

} else {
  /* Second grade equation */
  printf("\nEquation: %fx^2 + %fx + %f = 0", a, b, c);

  /* Calculate discriminant */
  discriminant = b*b - 4*a*c;

  /* Test discriminant */
  if(discriminant >= 0) {
    /* Real solutions */
    x1 = (-b + sqrt(discriminant) ) / (2*a) ;
    x2 = (-b - sqrt(discriminant) ) / (2*a) ;
    printf("\nx1 = %f", x1);
    printf("\nx2 = %f", x2);

  } else {
    /* Imaginary solutions */
    r = -b / (2*a);
    complex = sqrt(-discriminant) / (2*a);
    printf("\nx1 = %f + %fi", r, complex);
    printf("\nx2 = %f - %fi", r, complex);
  }

}
```

First grade eq.

Second grade eq.

Real solutions

Imaginary solutions

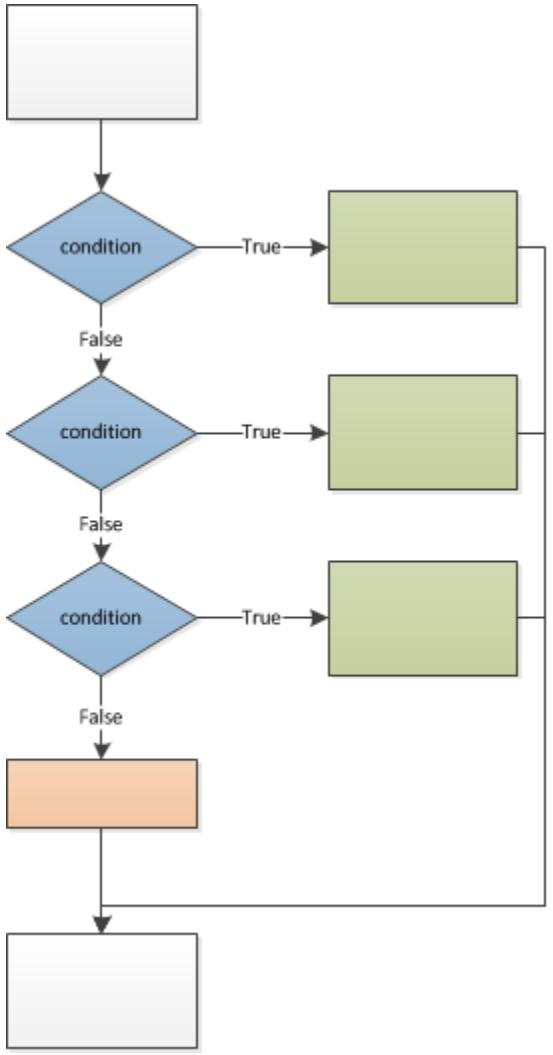
if else if

Multiple alternative choices

Mutually exclusive

If all the conditions are false, the final *else* block is executed

```
if (<boolean expression>) {
  <statement(s)>
} else if (<boolean expression>)  {
  <statement(s)>
} else if (<boolean expression>)  {
  <statement(s)>
} else {
  <statement(s)>
}
```



2. Conditional instructions
if else

$$f(x) = \begin{cases} -1 & \dots & x < -1 \\ -x^2 & \dots & -1 \le x < 0 \\ x^2 & \dots & 0 \le x < 1 \\ 1 & \dots & x \ge 1 \end{cases}$$

```
13  float x, y;
14
15  /* Read x value */
16  printf("Enter x value: ");
17  scanf("%f", &x);
18
19  /* Calculate ouput */
20  if( x < -1 ) {
21    y = -1;
22
23  } else if( (x >= -1) && (x < 0) ) {
24    y = - (x*x);
25
26  } else if( (x >= 0) && (x < 1) ) {
27    y = x*x;
28
29  } else {
30    y = 1;
31  }
32
33  /* Print output */
34  printf("y = %f\n", y);
```

Alternative 1

Alternative 2

Notice that x >= -1 is not necessary

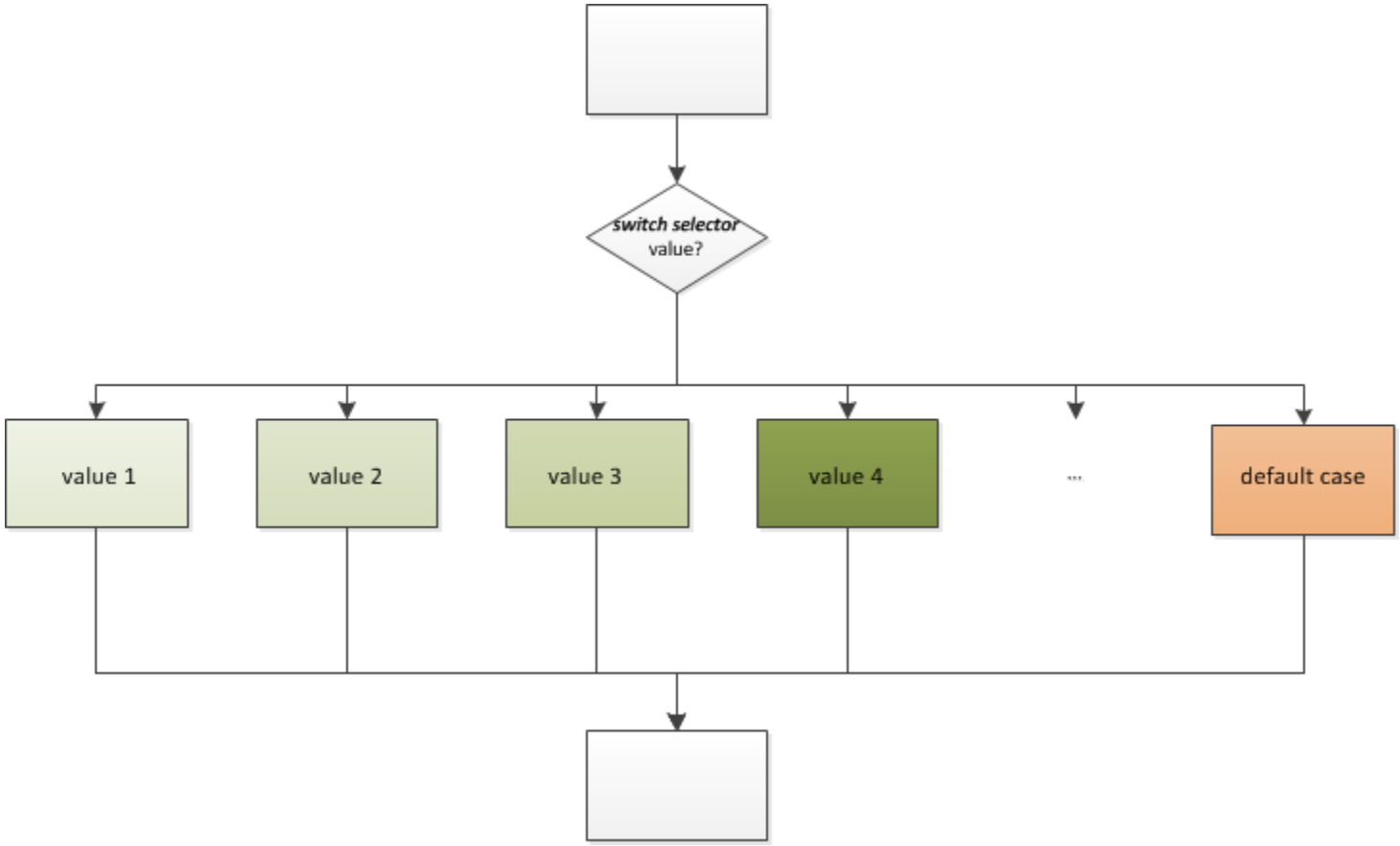
Alternative 3

Default

Outline

1. Introduction
2. **Conditional instructions**
   a. `if else`
   b. **`switch`**
3. Loop instructions
   a. `while`
   b. `do while`
   c. `for`

Allows for multiple execution paths, depending on the value of the ***switch selector***



2. Conditional instructions

switch

```c
#include <stdio.h>

int main(void) {
    char c;

    printf("enter a single character: ");
    scanf("%c", &c);

    switch (c){
        case 'a':
            printf ("Vowel a");
            break;
        case 'e':
            printf ("Vowel e");
            break;
        case 'i':
            printf ("Vowel i");
            break;
        case 'o':
            printf ("Vowel o");
            break;
        case 'u':
            printf ("Vowel u");
            break;
        default:
            printf ("Consonant, uppercase vowel or other symbol");
    }

    printf("\n");
    system("PAUSE");
    return 0;
}
```

If the *selector* is equal to the value in a **`case`**, the next sentences are executed **until a `break` is found**

- `break` terminates the **`switch`**
- `break` is not compulsory

If a `case` block does not have a `break`, the execution continues in the next `case`, even if it is false

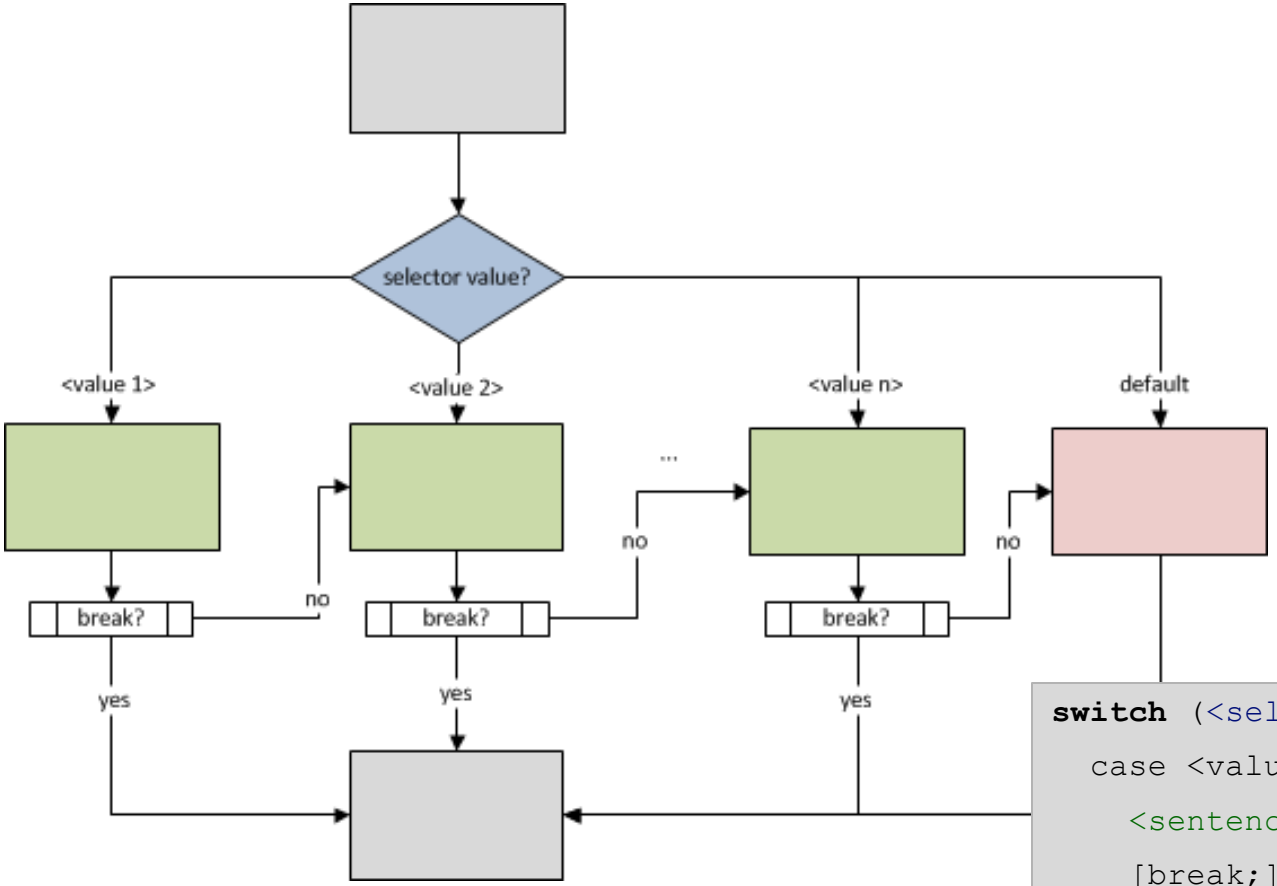
If no **`case`** is true, the **`default`** block (if defined) is executed

- `default` is not compulsory!

The switch *selector* must be an integer, *boolean* or character expression (floating point values are not allowed)

The *case* values must be integer, *boolean* or character constants

2. Conditional instructions




```
switch (<selector expression>) {
  case <value 1>:
    <sentences>
    [break;]
  case <value 2>:
    <sentences>
    [break;]
  [default:
    <sentences>
    [break;]]
}
```

2. Conditional instructions

switch

```c
#include <stdio.h>

int main(void) {
    char c;

    printf("enter a single character: ");
    scanf("%c", &c);

    switch (c){
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
            printf ("Vowel %c", c);
            break;
        default:
            printf ("Consonant, uppercase vowel or other symbol");
    }

    printf("\n");
    system("PAUSE");
    return 0;
}
```

2. Conditional instructions

switch

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

int main(void) {

    int n_edges;
    scanf("%i", &n_edges);

    switch(n_edges) {       // n_edges is the 'selector'
        case 0:
        case 1:
        case 2:             // Grouped values
            printf("Not a polygon");
            break;
        case 3:
            printf("Triangle");
            break;
        case 4:
            printf("Quadrilateral");
            break;
        case 5:
            printf("Pentagon");
            break;
        default:
            printf("More than 5 edges");
            break;
    }

    printf("\n");
    system("pause");
    return 0;
}
```

- Example
  - Implement the basic functioning of an ATM machine
    - Basic: It requires the use of a password and has only one operation:
      - Withdraw money
    - Advanced: Three operations:
      - Withdraw money
      - Consult credit
      - View password

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

#define PASSWORD         1234
#define INITIAL_AMOUNT   3000

int main(void) {
  int number;
  int amount;
  int withdraw;

  printf("Welcome to the ATM services!\n");


  /* Ask user for password */
  printf("Enter password: ");
  scanf("%i", &number);

  if(number == PASSWORD) {
    /* Password is correct */
    amount = INITIAL_AMOUNT;    /* Initialize amount */

    /* Withdraw option */
    printf("Enter amount to withdraw: ");
    scanf("%i", &withdraw);

    if(withdraw <= amount) {
      printf("[OK] Withdraw %i Euros. ", withdraw);
      amount -= withdraw;
    } else {
      printf("[ERROR] Insufficient credit.");
    }

    printf("\nRemaining credit: %i Euros\n", amount);
    printf("\nThanks for using our services. Goodbye.\n\n");

  } else {
   /* Password is not correct */
    printf("[ERROR] Wrong password.\n");
  }

  /* End application */
  system("PAUSE");
  return 0;
}
```

Use `switch` here to implement different options

```c
if(number == PASSWORD) {
  /* Password is correct */
  amount = INITIAL_AMOUNT;   /* Initialize amount */

  /* Ask user for choice */
  printf("Please enter your option: \n");
  printf("1: Withdraw money\n");
  printf("2: Consult credit\n");
  printf("3: View password\n");
  printf("4: Exit\n");
  printf("Your selection: ");
  scanf("%i", &choice);

  /* Switch on choice */
  switch(choice) {
    case 1:
      /* Withdraw option */
      printf("Enter amount to withdraw: ");
      scanf("%i", &withdraw);

      if(withdraw <= amount) {
        printf("[OK] Withdraw %i Euros. ", withdraw);
        amount -= withdraw;

      } else
        printf("[ERROR] Insufficient credit.");

      printf("\nRemaining credit: %i Euros\n", amount);
      break;

    case 2:
      /* Print credit */
      printf("\nRemaining credit: %i Euros\n", amount);
      break;

    case 3:
      /* View password */
      printf("\nThe password is: %i\n", number);
      break;
```

Outline

1. Introduction
2. Conditional instructions
   a. `if else`
   b. `switch`
3. **Loop instructions**
   a. **`while`**
   b. `do while`
   c. `for`

Loop instructions

`while` and `do-while`

A block of instructions is executed while the result of a logical expression (the condition) is true

`for`

A block of instructions is executed several times

3. Loop instructions

while

Continually executes a block of statements while a particular condition is true

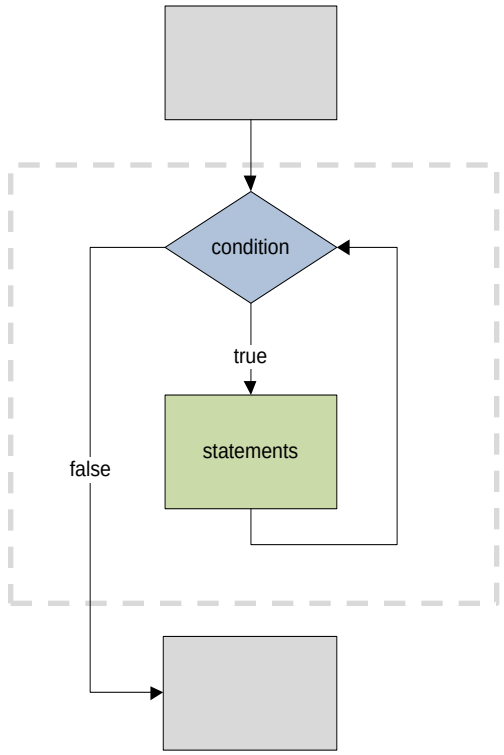
First, if the condition is true, the block of code associated to the `while` is executed

After finishing the **`while`** block, the condition is tested again

If the condition is false, the execution continues

Used if the number of iterations is not fixed

If the condition is never false > **infinite loop**




```
while(<boolean expression>) {
  <statement(s)>
}
```

Example

Ask for a password until it is correct

```
14  /* Read password */
15  int number = 0;
16
17  while(number != PASSWORD) {
18    /* Ask user for password */
19    printf("Enter password: ");
20    scanf("%i", &number);
21  }
22
23  /* Ask user for choice */
24  printf("Please enter your option: \n");
25  printf("1: Withdraw money\n");
26  printf("2: Consult credit\n");
27  printf("3: View password\n");
28  printf("4: Exit\n");
```

Initialization of the variable
The initial value must be different from the PASSWORD to enter the while for the first time and read the user input.

Example

Ask for a password while until it is correct

```
14  /* Read password */
15  int number = 0;
16  int tries  = 0;
17
18  while(number != PASSWORD && tries < MAX_TRIES) {
19    /* Ask user for password */
20    printf("Enter password: ");
21    scanf("%i", &number);
22    tries++;
23  }
24
25  if(number != PASSWORD) {
26    printf("Exceeded number of tries!\n");
27
28  } else {
29    /* Ask user for choice */
30    printf("Please enter your option: \n");
31    printf("1: Withdraw money\n");
32    printf("2: Consult credit\n");
33    printf("3: View password\n");
34    printf("4: Exit\n");
35  }
```

Complex condition
The *while* block is executed while the PASSWORD has not been found **AND** the number of tries is less than the MAX_TRIES value

Post-check
The *while* block ends when the password is found or when the tries limit is reached. It is necessary to test after the while which one has been the exit situation

Outline

1. Introduction
2. Conditional instructions
   a. `if else`
   b. `switch`
3. **Loop instructions**
   a. `while`
   b. **`do while`**
   c. `for`

3. Loop instructions

do while

Executes a block of statements; repeat the execution if the condition is true

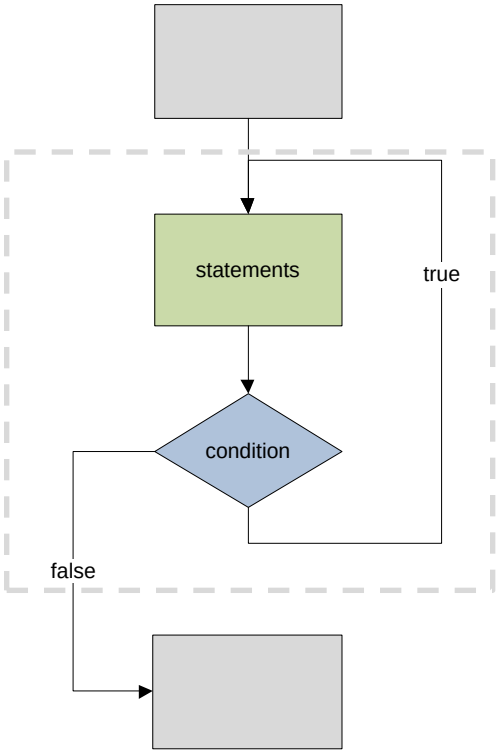
First, the block of code is executed

If the condition is true, the block of code associated to the **do while** is executed again

After finishing the **do while** block, the condition is tested again

If the condition is false, the execution continues

The block is executed at least once!




```
do{
   <statement(s)>
} while(<boolean expression>);
```

Example

Ask for a password until it is correct

Initialization of the variable
It is not necessary to initialize the number variable.

```
14  /* Read password */
15  int number;
16
17  do {
18    /* Ask user for password */
19    printf("Enter password: ");
20    scanf("%i", &number);
21  } while(number != PASSWORD);
22
23  /* Ask user for choice */
24  printf("Please enter your option: \n");
25  printf("1: Withdraw money\n");
26  printf("2: Consult credit\n");
27  printf("3: View password\n");
28  printf("4: Exit\n");
```

```
14  /* Read password */
15  int number = 0;
16  int tries  = 0;
17
18  do {
19    /* Ask user for password */
20    printf("Enter password: ");
21    scanf("%i", &number);
22    tries++;
23  } while(number != PASSWORD && tries < MAX_TRIES);
24
25  if(number != PASSWORD) {
26    printf("Exceeded number of tries!\n");
27
28  } else {
29    /* Ask user for choice */
30    printf("Please enter your option: \n");
31    printf("1: Withdraw money\n");
32    printf("2: Consult credit\n");
33    printf("3: View password\n");
34    printf("4: Exit\n");
```

Print on the screen the numbers in {1, …, 50} in decreasing order

```
/* Print numbers in [1, 50] in decreasing order */
int n = 50;
while(n >= 1) {
  printf("%i\n", n);
  n--;
}
```

```
/* Print numbers in [1, 50] in decreasing order */
int n = 50;
do {
  printf("%i\n", n);
  n--;
} while(n >= 1);
```

What is the value of n after finishing the block?

Print on the screen the numbers in {1, …, 100} divisible by 5 and 7

```
/* Print numbers in [1, 100] divisible by 5 and 7 */
int a = 1;
while(a <= 100) {
  if( a%5==0 && a%7==0)
    printf("%i\n", a);
  a++;
}
```

Add numbers while the number read is larger than 0

```
/* Add numbers while the number read is larger than 0 */
int a;
int sum = 0;

do {
  scanf("%i", &a);
  if(a > 0)
    sum += a;
} while(a > 0);
printf("Sum: %i\n", sum);
```

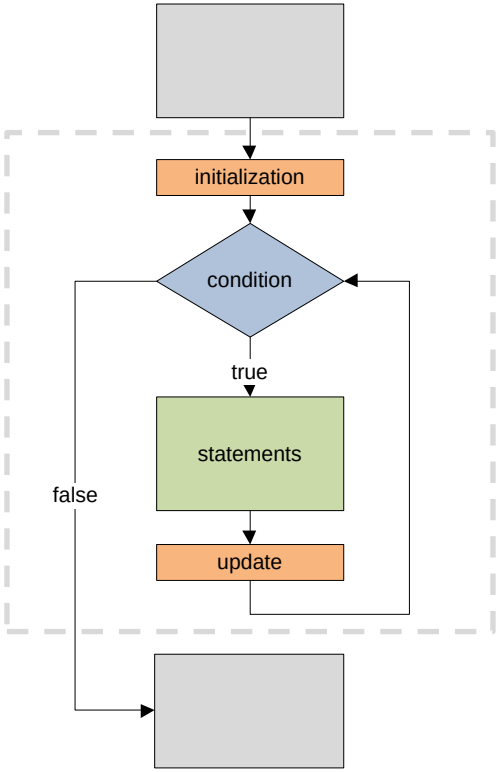
Outline

1. Introduction
2. Conditional instructions
   a. `if else`
   b. `switch`
3. **Loop instructions**
   a. `while`
   b. `do while`
   c. **`for`**

3. Loop instructions

`for`

Executes a block of statements; repeat the execution if the condition is true (the same as **`while`**)



Additionally, performs more operations

- **Initialization statement**
- **Update statement**

The first time, the pre-block statement is executed

If the condition is true, the associated block of code is executed

After finishing the block, the after-block statement is executed

If the condition is true, the block is executed again

If the condition is false, the execution continues

```
for ([initialization]; <bool. expr.>; [update]){
  <statement(s)>
}
```

Print on the screen the numbers in [1, 50]

Increasing order

```
/* Print numbers in [1, 50] (increasing) */
int i;
printf("Increasing order\n");

for(i=1; i<=50; i++)
  printf("%i\n", i);
```

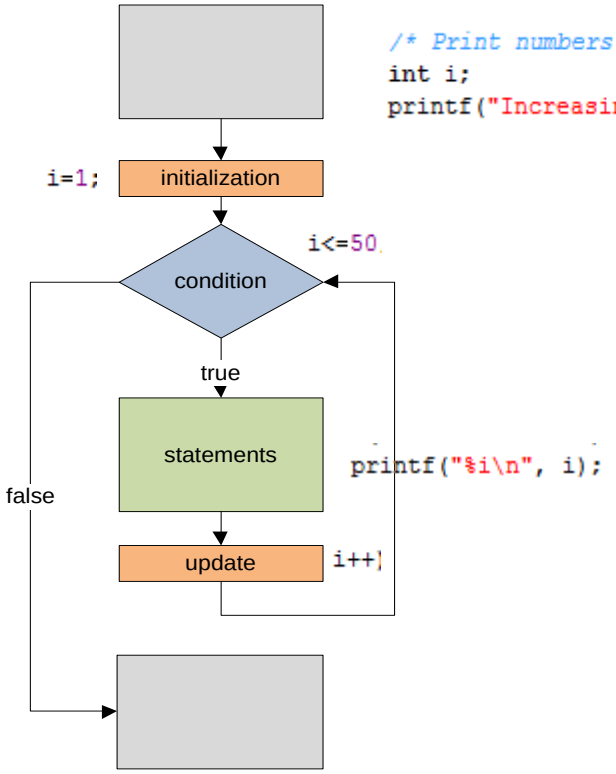
Decreasing order

```
/* Print numbers in [1, 50] (decreasing) */
int i;
printf("Decreasing order\n");

for(i=50; i>=1; i--)
  printf("%i\n", i);
```

What is the value of `i` after finishing the block?

3. Loop instructions

for




```
/* Print numbers in [1, 50] (increasing) */
int i;
printf("Increasing order\n");
```

Add the numbers in [1, 50]

```
/* Add numbers in [1, 50] */
int i;
int sum = 0;
for(i=1; i<=50; i++)
  sum += i;
printf("Sum: %i\n", sum);
```

Loop and control flow instructions can be nested with the following observations:

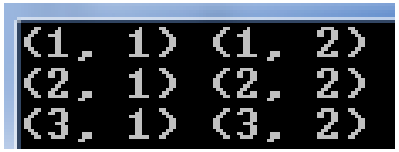
- The inner instruction must be included **inside** the outer instruction
- With control flow instructions
  - For each value of the counter of the outer instruction, the counter of the inner instruction takes all its values

Read an integer value (larger than 0) and print all the positive integers lesser than this value. The operation is repeated until the user enters the value 0.

```
while(!stop) {
  printf("\nEnter a number (0 to end): ");
  scanf("%i", &num);

  if(num > 0) {
    int i;
    for(i=1; i<=num; i++)
      printf("%i ", i);

  } else if (num == 0) {
    stop = 1;

  } else {
    printf("Negative values are not allowed\n");
  }
}
```

Show the index positions of a 2x3 matrix

```
/* Matrix positions */
int nRows = 3, nCols = 2;
int i, j;
for(i=1; i<=nRows; i++) {

  for(j=1; j<=nCols; j++)
    printf("(%i, %i) ", i, j);

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



```c
/* Day of the week */
int cont;
int n;

do {
  printf("\nEnter an integer in [1, 7]: ");
  scanf("%i", &n);

  switch(n) {
    case 1:
      printf("Monday");
      break;
    case 2:
      printf("Tuesday");
      break;
    case 3:
      printf("Wednesday");
      break;
    case 4:
      printf("Thursday");
      break;
    case 5:
      printf("Friday");
      break;
    case 6:
      printf("Saturday");
      break;
    case 7:
      printf("Sunday");
      break;
    default:
      printf("Wrong value");
  }

  printf("\n\nContinue (yes:1/no:0)? ");
  scanf("%i", &cont);

} while(cont == 1);
```

Read an integer value representing a day of the week and print the corresponding name. Ask the user to repeat the procedure.

- Example
  - Print on the screen the seconds, minutes and hours in a clock
    - Basic: Only hours and minutes

```
00:01
00:02
…
00:59
01:00
01:01
…
```

    - Advanced: Hours, minutes and seconds

Outline

1. Introduction
2. Conditional instructions
   a. `if else`
   b. `switch`
3. Loop instructions
   a. `while`
   b. `do while`
   c. `for`

Bibliography

Recommended lectures

Basic

- Ivor Horton. *Beginning C: From Novice to Professional*. Apress, 2006 (4th Edition) – Chapters 3, 4

Additional information

- Stephen G. Kochan. *Programming in C.* Sams, 2004 (3rd Edition), Programming in C – Chapters 5, 6
- Stephen Prata. *C Primer Plus.* Sams, 2004 (5th Edition) – Chapters 5, 6, 7

Program structure

```
#include <stdio.h>
…
#define MAX 100
…
int main(void){
    <statements>
    return 0;
}
```

Declaration statements

```
<type> <variable name> [= <expression>];
const <type> <variable name> [= <expression>];
```

Assignment statements

```
<variable> = <expression> ;
```

Input and output

```
printf("%i", a);
scanf("%i", &a)
```

if-else

```
if(<condition>) {
   <statements>
} else {
   <statements>
}
```

if-else-if

```
if(<condition>) {
   <statements>
} else if(<condition>) {
   <statements>
} else if(<condition>) {
   <statements>
} else {
   <statements>
}
```

switch

```
switch(<selector>) {
   case <value>:
        <statements>
        break;
   …
   case <value>:
        <statements>
        break;
   default:
        <statements>
}
```

for

```
for(<variable>=<initial value>; <condition>; <update variable> ) {
   <statements>
}
```

while

```
while(<condition>) {
   <statements>
}
```

do while

```
do {
   <statements>
} while(<condition>);
```

Outline

1. Introduction
2. Conditional instructions
   a. `if else`
   b. `switch`
3. Loop instructions
   a. `while`
   b. `do while`
   c. `for`