Ho Chi Minh City University of Technology
Faculty of Computer Science and Engineering

Chapter 4: Selection Statements

Introduction to Computer Programming
(C language)

Nguyễn Tiến Thịnh, Ph.D.

Email: ntthinh@hcmut.edu.vn

2022 – 2023, Semester 1

Course Content

C.1. Introduction to Computers and Programming

C.2. C Program Structure and its Components

C.3. Variables and Basic Data Types

C.4. Selection Statements

C.5. Repetition Statements

C.6. Functions

C.7. Arrays

C.8. Pointers

C.9. File Processing

References

[1] "*C: How to Program*", 7th Ed. – Paul Deitel and Harvey Deitel, Prentice Hall, 2012.

[2] "*The C Programming Language*", 2nd Ed. – Brian W. Kernighan and Dennis M. Ritchie, Prentice Hall, 1988

and others, especially those on the Internet

Content

Introduction

if.. statements

if..else.. statements

Nested if../if..else.. statements

switch..case.. statements

Summary

Introduction

Recall

- Statement

  ended with a semicolon (;)

  stretched on multiple lines with a backslash \ at the end

  able to be grouped in the brackets {}

  not consider spaces

- Block

  specified by {} with no semicolon after the right brace

  contains as many statements as required

  is a compound statement, syntactically equivalent to a single statement

Sequentially processed from the beginning to the end of a function

Introduction

- Given a set of n positive numbers, find the smallest one.

(Chapter 1 – Real code in C)

```
void main() {
    double positiveNumber[10] = {2, 1, 3, 10, 8, 3, 4, 5, 9, 12};
    int n = 10;
    double minNumber = positiveNumber[0];
    int iteration = 1;
    while (iteration < n) {
        if (minNumber <= positiveNumber[iteration])
            iteration = iteration + 1;
        else {
            minNumber = positiveNumber[iteration];
            iteration = iteration + 1;
        }
    }
}
```

Single statement

Block

Introduction

Control statements in C

- Sequence
  - Assignment
  - Function calling
  - …
- Selection
  - if
  - if..else..
  - switch..case..
- Repetition
  - for..
  - while..
  - do..while..

Introduction

Given a set of n positive numbers, find the smallest one.

(Chapter 1 – Real code in C)

```
void main() {
    double positiveNumber[10] = {2, 1, 3, 10, 8, 3, 4, 5, 9, 12};
    int n = 10;
    double minNumber = positiveNumber[0];
    int iteration = 1;
    while (iteration < n) {
        if (minNumber <= positiveNumber[iteration])
            iteration = iteration + 1;
        else {
            minNumber = positiveNumber[iteration];
            iteration = iteration + 1;
        }
    }
}
```

Control Statements for Selection

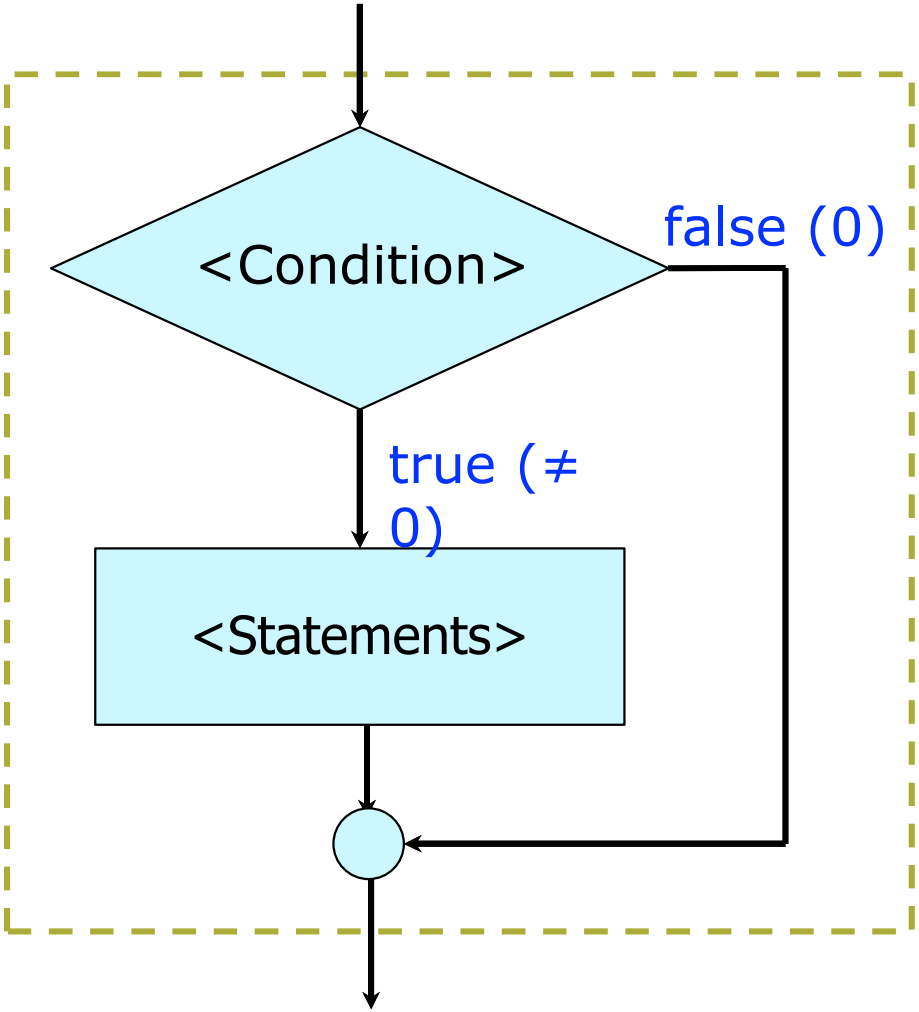
if.. statements

if (<Condition>**)** <Statement>

```
if (1==1) printf("print: 1=1");

if (1==0) printf("never print!");
```

if (<Condition> **)**
{
<Statements>
}

```
if (1==1)
{
    printf("block\n");
    printf("print: 1=1");
}

if (1==0)
{
    printf("block\n");
    printf("never print!");
}
```



<Statements> is performed (selected) if <Condition> is true.
Otherwise, ignored.

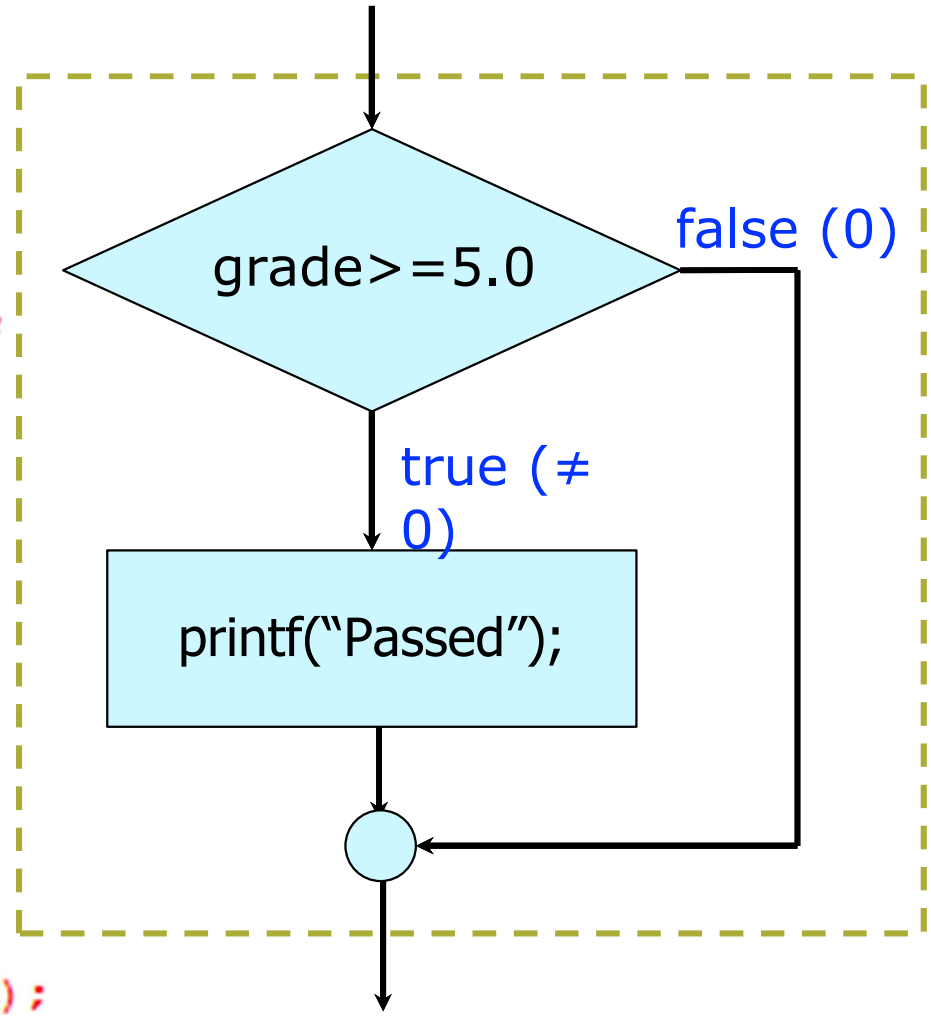
if.. statements

```
grade = 7.5;

if (grade >= 5.0) printf("Passed");

printf("\n\nafter the if.. statement\n");
```

```
Passed

after the if.. statement
```

```
grade = 4.5;

if (grade >= 5.0) printf("Passed");

printf("\n\nafter the if.. statement\n");
```

```
after the if.. statement
```



Print "Passed" if grade >= 5.0.

Otherwise, ignored.

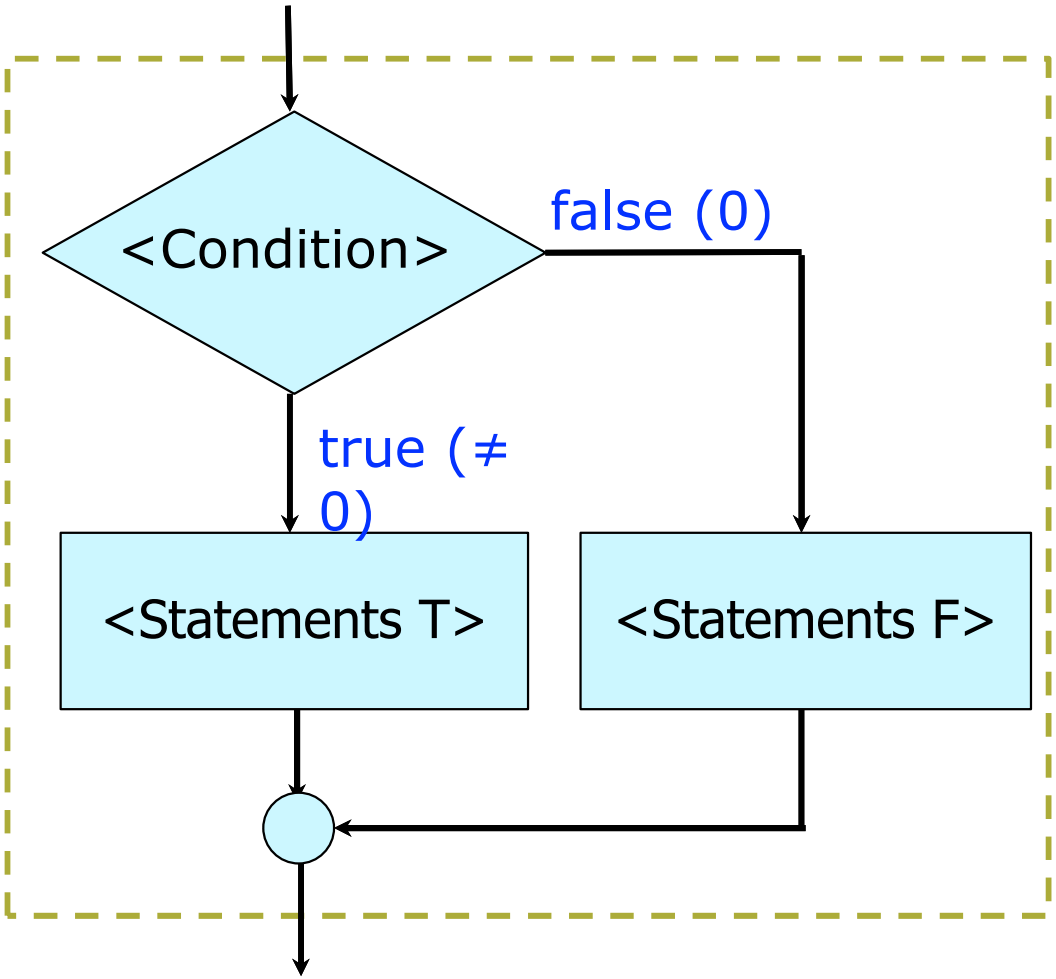
if..else.. statements

```
if (<Condition>) <Statement T>
else <Statement F>
```

```
if (<Condition>) <Statement T>
else
{
        <Statements F>
}
```

```
if (<Condition> )
{
        <Statements T>
}
else  <Statement F>
```

```
if (<Condition> )
{
        <Statements T>
}
else
{
        <Statements F>
}
```



<Statements T> is performed (selected) if <Condition> is true.
Otherwise, <Statements F> is performed.

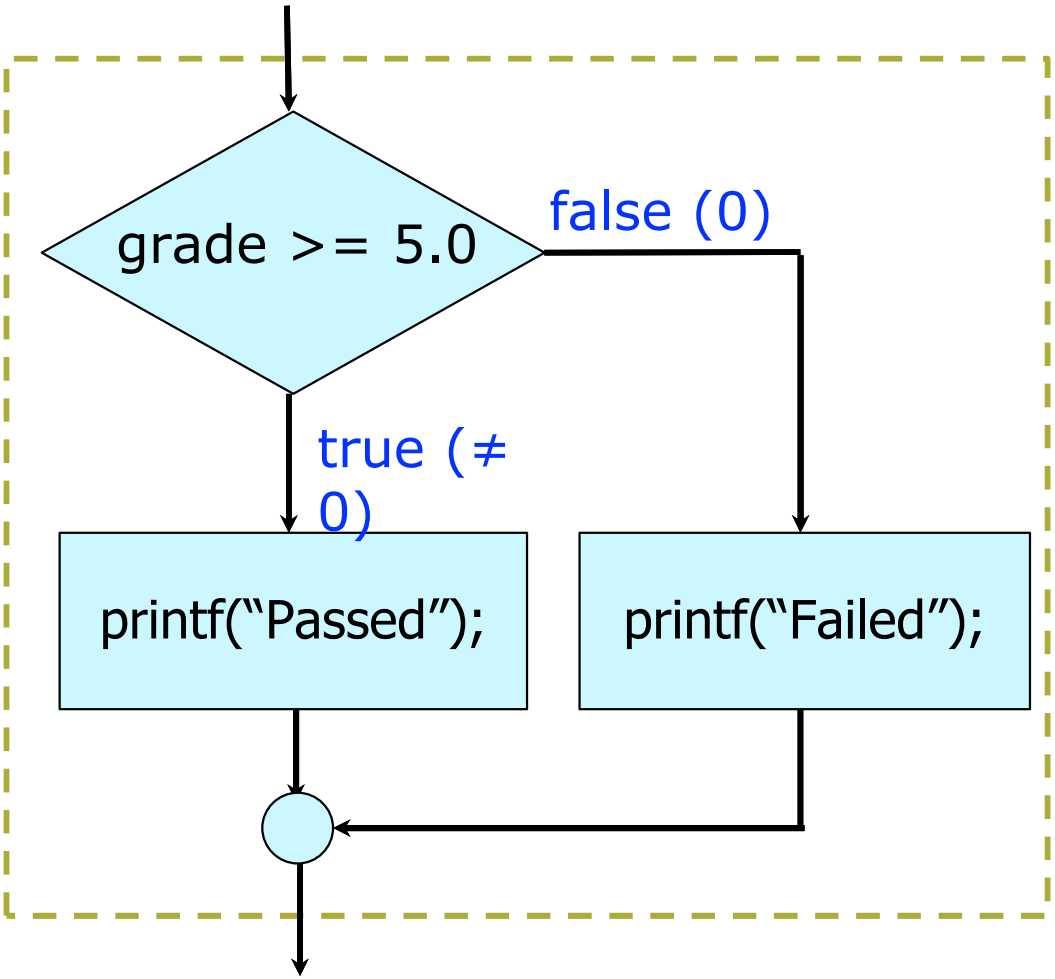
if..else.. statements

```
grade = 7.5;

if (grade >= 5.0) printf("Passed");
else printf("Failed");

printf("\n\nafter the if..else.. statement\n");
```

```
Passed

after the if..else.. statement
```

```
grade = 4.5;

if (grade >= 5.0) printf("Passed");
else printf("Failed");

printf("\n\nafter the if..else.. statement\n");
```

```
Failed

after the if..else.. statement
```



Print "Passed" if grade >= 5.0.
Otherwise, print "Failed".

if..else.. statements

- Conditional expression:

<condition>**?**<expression T>**:**<expression F>

```
result = grade>=5.0?'P':'F';
```

can be regarded as:

if (<condition>**)** <expression T>;

else <expression F>;

```
if (grade>=5.0) result = 'P';
else result = 'F';
```

if..else.. statements

```
float grade = 5.5;

grade>=5.0?printf("Passed"):printf("Failed");

if (grade>=5.0) printf("Passed");
else printf("Failed");
```

Which one do you prefer: conditional expressions or if..else.. statements?

```
float grade = 5.5;

grade>=5.0?(printf("Passed"), printf("\ngrade=%4.1f", grade)):\
    (printf("Failed"), printf("\ngrade=%4.1f", grade));

if (grade>=5.0) {
    printf("Passed");
    printf("\ngrade=%4.1f", grade);
}
else {
    printf("Failed");
    printf("\ngrade=%4.1f", grade);
}
```

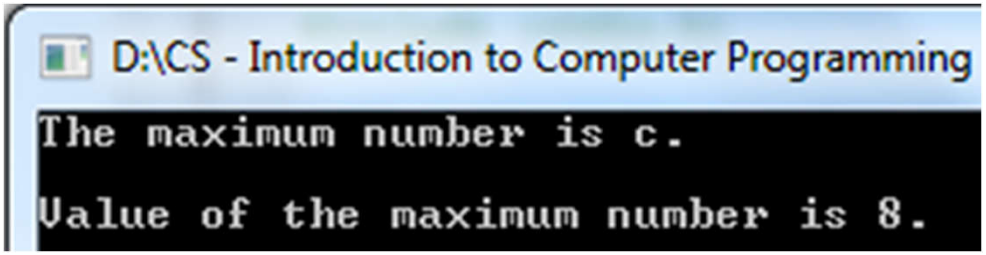
Nested if../if..else.. statements

```
if (<condition 1>)
{
        …
        if (<condition 2>) …
        …
}
```

```
if (<condition 1>)
{
        …
        if (<condition 2>) …
        …
}
else
{
        …
        if (<condition 3>) …
        …
}
```

Nested if../if..else.. statements

```c
#include <stdio.h>

//Find the maximum number among three numbers
void main () {

    int a = 3, b = 2, c = 8;

    if (a<b) {
        if (b<c) {
            printf("The maximum number is c.\n\n");
            printf("Value of the maximum number is %d.\n", c);
        }
        else { //b>=c
            printf("The maximum number is b.\n\n");
            printf("Value of the maximum number is %d.\n", b);
        }
    }
    else { //a>=b
        if (a<c) {
            printf("The maximum number is c.\n\n");
            printf("Value of the maximum number is %d.\n", c);
        }
        else { //a>=c
            printf("The maximum number is a.\n\n");
            printf("Value of the maximum number is %d.\n", a);
        }
    }
}
```



Nested if../if..else.. statements

```c
#include <stdio.h>

//selection statements for checking if user's inputs are valid
void main() {

    int yourAge;
    char yourAnswer;

    printf("Enter your age: ");
    scanf("%d", &yourAge);

    if (yourAge > 0) {
        if (yourAge >= 18) {
            printf("\nHave you ever been alone in Ha Noi? (Y/N) ");
            scanf("%*c%c", &yourAnswer);
            if (yourAnswer == 'Y' || yourAnswer == 'y') printf("\nWOW!");
            else printf("\n...");
        }
        else {
            printf("\nHave you ever travelled with your family in Da Nang? (Y/N)");
            scanf("%*c%c", &yourAnswer);
            if (yourAnswer == 'Y' || yourAnswer == 'y') printf("\nWOW!");
            else printf("\n...");
        }
    }
}
```

```
Enter your age: 20

Have you ever been alone in Ha Noi? (Y/N) y

WOW!
```

Nested if../if..else.. statements

A multi-way decision

if (<condition 1>**)** <statements T1>

else if (<condition 2>**)** <statements T2>

else if (<condition 3>**)** <statements T3>

…

else if (<condition k>**)** <statements Tk>

else <statements Fk>

```
#include <stdio.h>

void main() {

    float grade;

    printf("Enter your grade (>=0): ");
    scanf("%f", &grade);
    printf("\nYour grade is %4.1f\n", grade);

    if (grade < 0 || grade > 10) return;

    if (grade < 5.0) printf("\nunsatisfactory");
    else if (grade >= 5.0 && grade < 7.0) printf("\nminimally satisfactory");
    else if (grade >= 7.0 && grade < 8.5) printf("\nsatisfactory");
    else printf("\nexemplary");
}
```

```
Enter your grade (>=0): 7.75

Your grade is  7.8

satisfactory
```

Nested if../if..else.. statements

Be careful with specifying "else" for "which if":

if (<condition 1>**)**

    if (<condition 2>**)** <statements T2>

else <statements F>

<u>should be</u>:

if (<condition 1>**)** {

    if (<condition 2>**)** <statements T2>

}

else <statements *F1*>

<u>or</u>:

if (<condition 1>**)** {

    if (<condition 2>**)** <statements T2>

    else <statements *F2*>

}

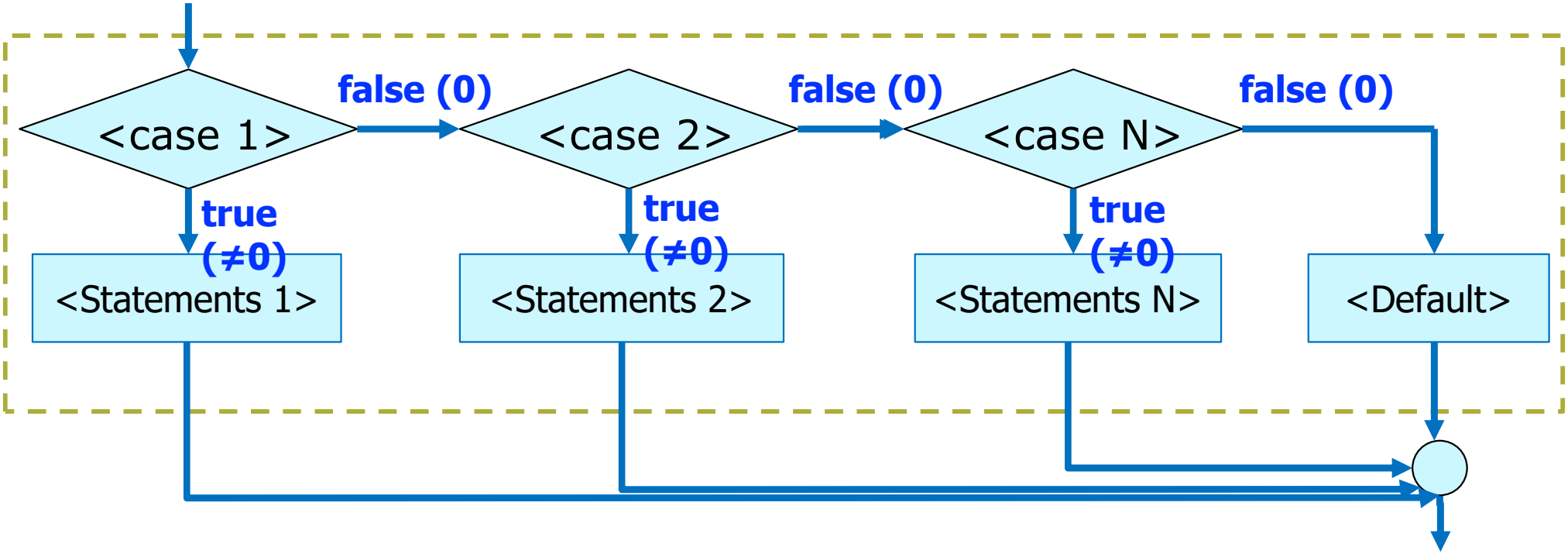
```c
#include <stdio.h>

main () {
    int a = 1, b = 3, c = 2, d = 5;

    if (a>b)
        if (b>c)
            d = 10;
    else
        d = 20;

    printf("d = %d\n", d);
}
```

d = ? <u>5</u>? 10? 20?

```c
#include <stdio.h>

main () {
    int a = 1, b = 3, c = 2, d = 5;

    if (a>b) {
        if (b>c)
            d = 10;
    }
    else
        d = 20;

    printf("d = %d\n", d);
}
```

d = ? 5? 10? <u>20</u>?

switch..case.. statements



switch (<expression>) **{**

case <case 1>**:** <Statements 1>; ***break;***

case <case 2>**:** <Statements 2>; ***break;***

…

case <case N>**:** <Statements N>; ***break;***

[**default:** <Default>]

}

a multi-way decision that tests whether an expression matches one of a number of *constant* integer values, and branches accordingly

switch..case.. statements

switch (<expression>**) {**
case <case 1>**:** <Statements 1>; ***break*;**
case <case 2>**:** <Statements 2>; ***break*;**
…
case <case N>**:** <Statements N>; ***break*;**
[**default:** <Default>]
}

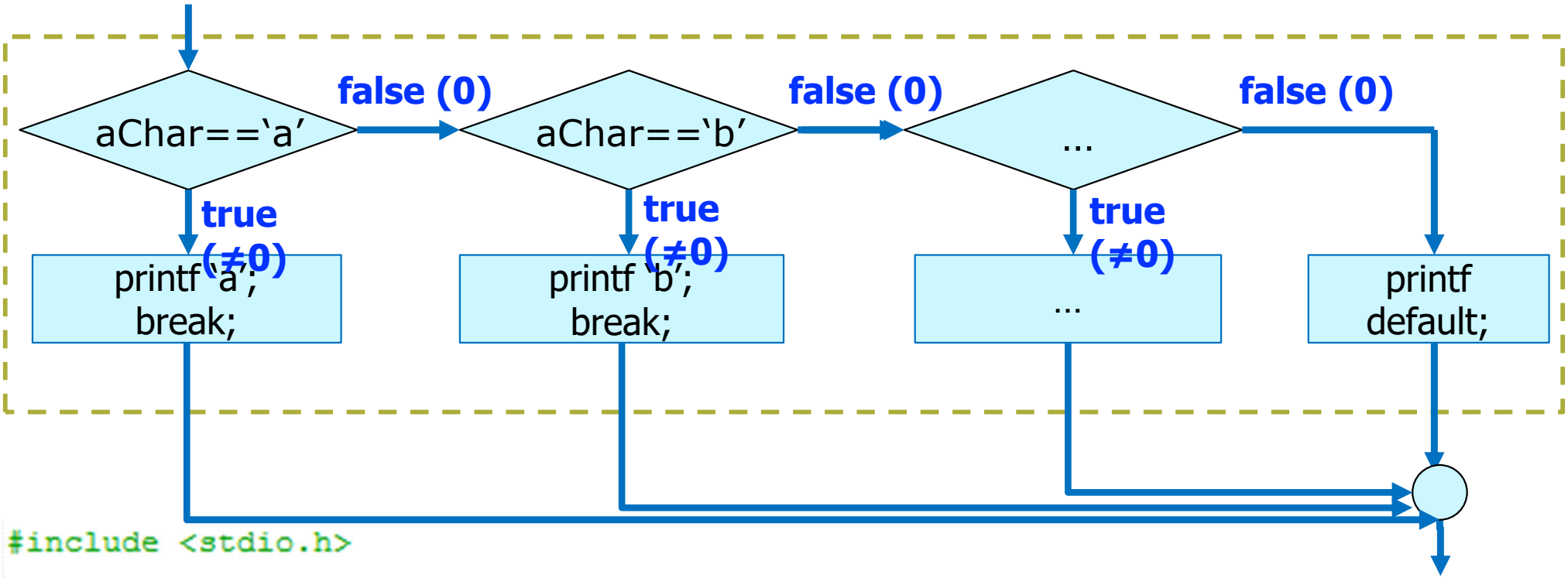
can be regarded as:

if (<expression> == <case 1>**)** <Statements 1>
else if (<expression> == <case 2>**)** <Statements 2>
…
else if (<expression> == <case N>**)** <Statements N>
[**else** <Default>]

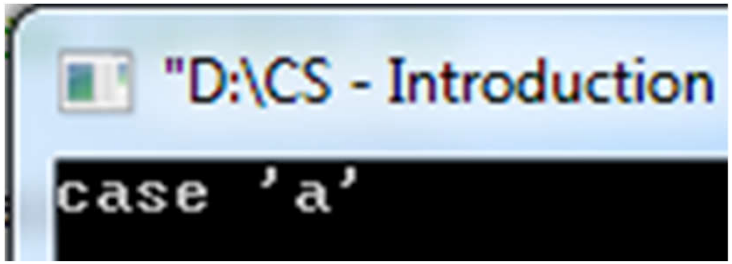
switch..case.. statements

- <expression> has a type of integer numbers, enumerated data, characters.
- <case 1>, …, <case N> are constants of one of the aforementioned types.
  - Cases serve as labels.
- [default: <Default>] is optional.
- "*fall-through*" property of switch..case..
  - After the code for one case is done, execution *falls through* to the next unless an explicit action is taken to escape.
    - break (return) statement

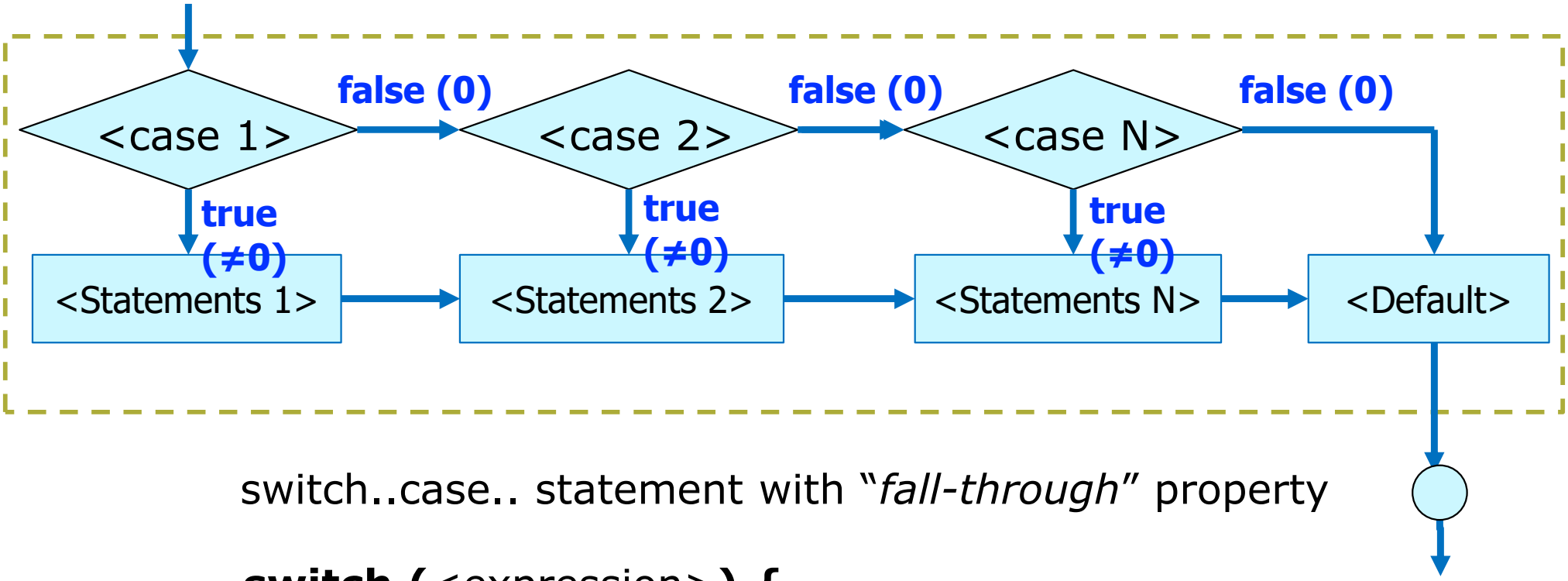
switch..case.. statements




```c
#include <stdio.h>

void main() {

    char aChar = 'a';

    switch (aChar) {
        case 'a': printf("case \'a\'\n"); break;
        case 'b': printf("case \'b\'\n"); break;
        default: printf("case default\n");
    }
}
```



switch..case.. statements



switch..case.. statement with "*fall-through*" property

switch (<expression>**) {**

case <case 1>**:** <Statements 1>

case <case 2>**:** <Statements 2>

…

case <case N>**:** <Statements N>

[**default:** <Default>]

}

switch..case.. statements

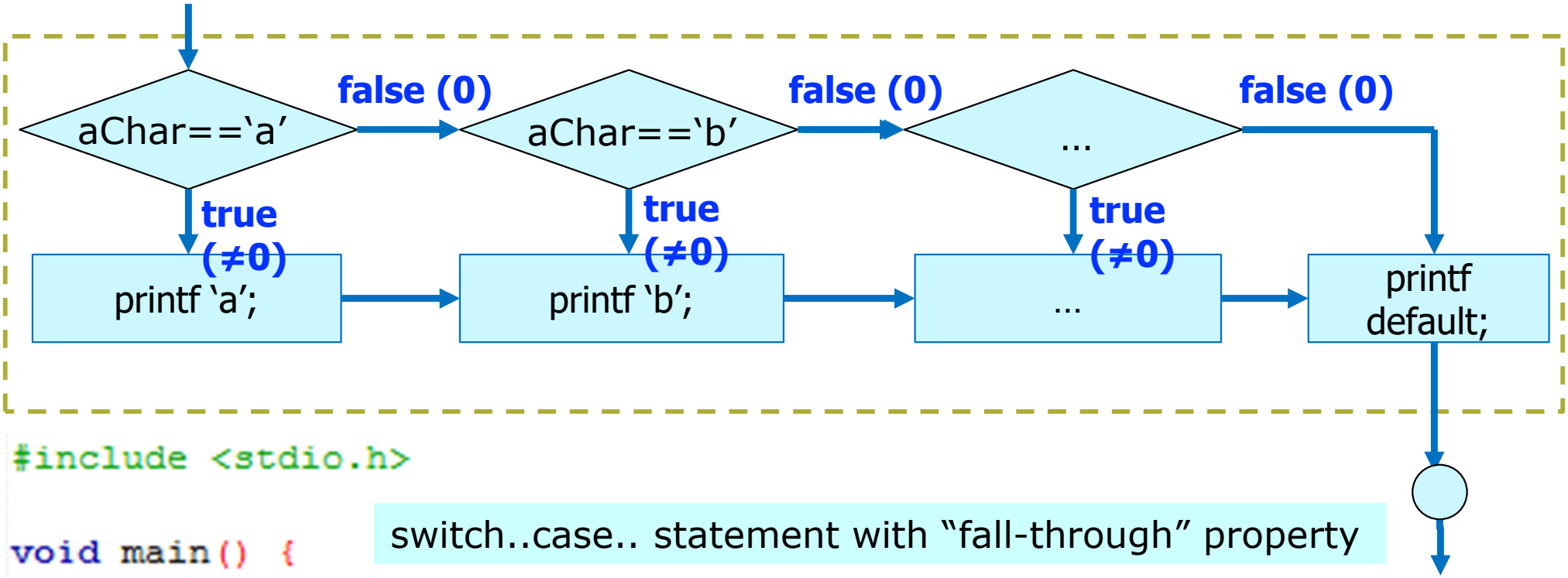
switch..case.. statement with “fall-through” property

```
switch (<expression>) {
        case <case 1>: <Statements 1>
        case <case 2>: <Statements 2>
        …
        case <case N>: <Statements N>
        [default: <Default>]
}
```

can be regarded as:

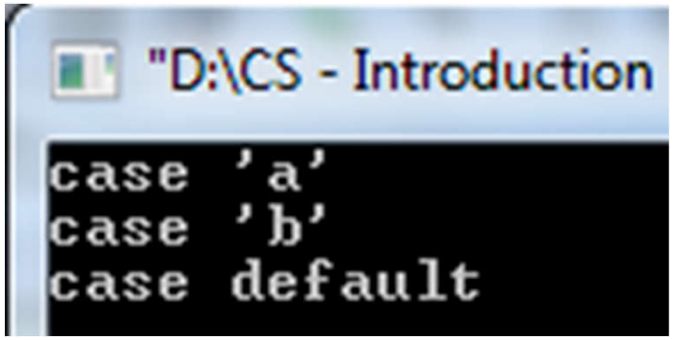
```
if (<expression> == <case 1>) {
        <Statements 1>
        <Statements 2>
        …
        <Default>
}
else if (<expression> == <case 2>) {
        <Statements 2>
        …
        <Default>
}
…
else if (<expression> == <case N>) {
        <Statements N>
        <Default>
}
[else <Default>]
```

switch..case.. statements

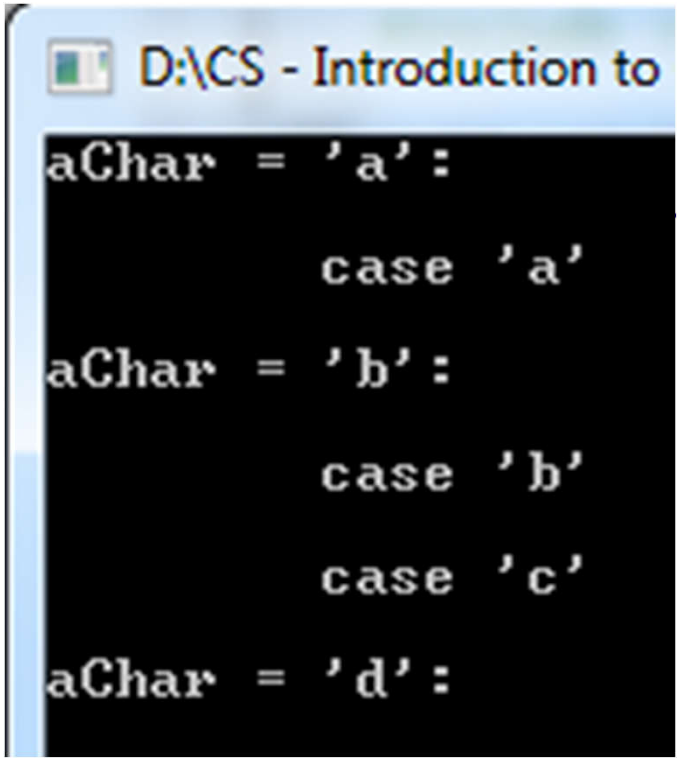



```c
#include <stdio.h>

void main() {

    char aChar = 'a';

    switch (aChar) {
        case 'a': printf("case \'a\'\n");
        case 'b': printf("case \'b\'\n");
        default: printf("case default\n");
    }
}
```

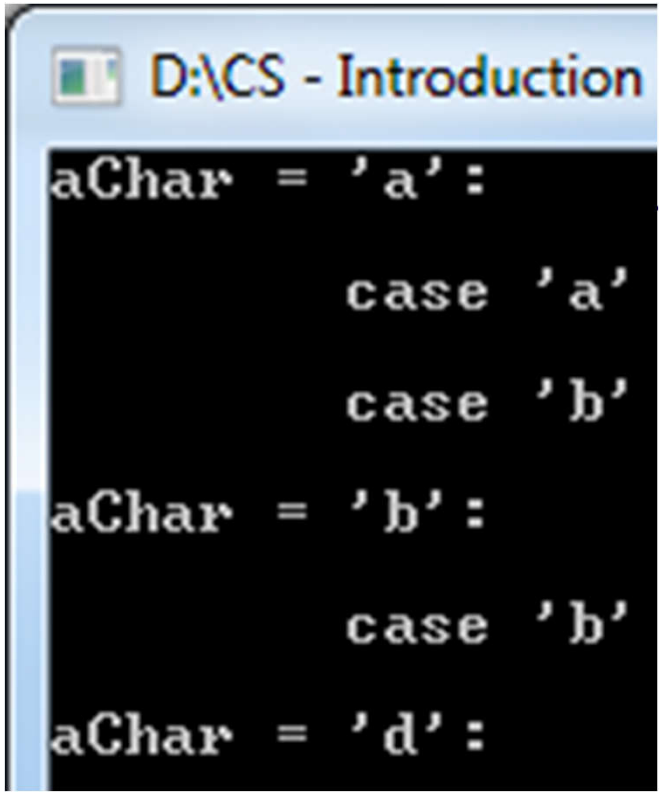
switch..case.. statement with "fall-through" property



```c
#include <stdio.h>

void main() {

    char aChar;

    aChar = 'a';

    printf("aChar = \'%c\':\n\n", aChar);

    switch (aChar) {
        case 'a': printf("\tcase \'a\'\n\n"); break;
        case 'b': printf("\tcase \'b\'\n\n");
        case 'c': printf("\tcase \'c\'\n\n");
    }

    aChar = 'b';

    printf("aChar = \'%c\':\n\n", aChar);

    switch (aChar) {
        case 'a': printf("\tcase \'a\'\n\n"); break;
        case 'b': printf("\tcase \'b\'\n\n");
        case 'c': printf("\tcase \'c\'\n\n");
    }

    aChar = 'd';

    printf("aChar = \'%c\':\n\n", aChar);

    switch (aChar) {
        case 'a': printf("\tcase \'a\'\n\n"); break;
        case 'b': printf("\tcase \'b\'\n\n");
        case 'c': printf("\tcase \'c\'\n\n");
    }
}
```



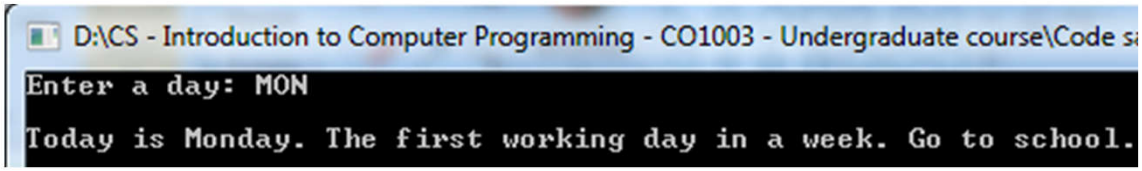
```c
#include <stdio.h>

void main() {

    char aChar;

    aChar = 'a';

    printf("aChar = \'%c\':\n\n", aChar);

    switch (aChar) {
        case 'a': printf("\tcase \'a\'\n\n");
        case 'b': printf("\tcase \'b\'\n\n"); break;
        case 'c': printf("\tcase \'c\'\n\n");
    }

    aChar = 'b';

    printf("aChar = \'%c\':\n\n", aChar);

    switch (aChar) {
        case 'a': printf("\tcase \'a\'\n\n");
        case 'b': printf("\tcase \'b\'\n\n"); break;
        case 'c': printf("\tcase \'c\'\n\n");
    }

    aChar = 'd';

    printf("aChar = \'%c\':\n\n", aChar);

    switch (aChar) {
        case 'a': printf("\tcase \'a\'\n\n");
        case 'b': printf("\tcase \'b\'\n\n"); break;
        case 'c': printf("\tcase \'c\'\n\n");
    }
}
```

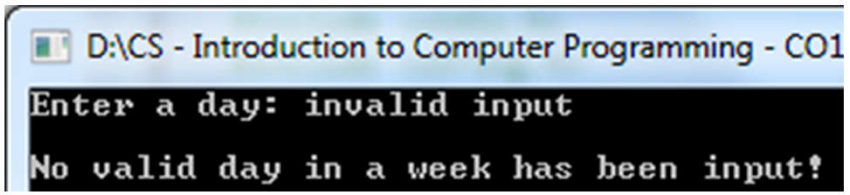


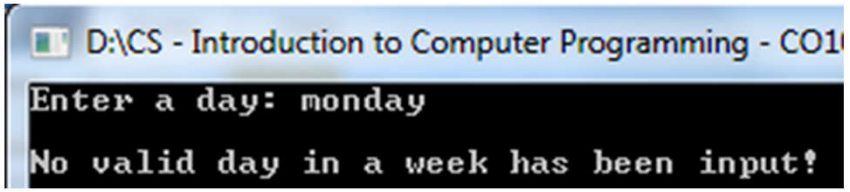
Put them all together

- Given a problem: build your timetable in a week. Input a day in a week and output its corresponding activities.
- string.h: a standard library file for strings
  - Compare two strings
    - int strcmp(const char *str1, const char *str2)
      - < 0 if str1 < str2 (*less than*)
      - > 0 if str1 > str2 (*greater than*)
      - = 0 if str1 = str2 (*equal*)
  - Copy a string to another one
    - char *strcpy(char *destination, const char *source)

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

typedef enum {MON, TUE, WED, THU, FRI, SAT, SUN} DAY;

void main() {

    DAY aDay;
    char sDay[3];

    printf("Enter a day: ");
    scanf("%s", sDay);

    if (strcmp(sDay, "MON")==0) {
        aDay = MON;
        strcpy(sDay, "Monday");
    }
    else if (strcmp(sDay, "TUE")==0) {
        aDay = TUE;
        strcpy(sDay, "Tuesday");
    }
    else if (strcmp(sDay, "WED")==0) {
        aDay = WED;
        strcpy(sDay, "Wednesday");
    }
    else if (strcmp(sDay, "THU")==0) {
        aDay = THU;
        strcpy(sDay, "Thursday");
    }
    else if (strcmp(sDay, "FRI")==0) {
        aDay = FRI;
        strcpy(sDay, "Friday");
    }
    else if (strcmp(sDay, "SAT")==0) {
        aDay = SAT;
        strcpy(sDay, "Saturday");
    }
    else if (strcmp(sDay, "SUN")==0) {
        aDay = SUN;
        strcpy(sDay, "Sunday");
    }
    else aDay = -1;

    switch(aDay) {
        case MON: printf("\nToday is %s. The first working day in a week. Go to school.\n", sDay); break;
        case TUE: printf("\nToday is %s. The second working day in a week. Do homework.\n", sDay); break;
        case WED: printf("\nToday is %s. The third working day in a week. Check the due date.\n", sDay); break;
        case THU: printf("\nToday is %s. The fourth working day in a week. Submit the work.\n", sDay); break;
        case FRI: printf("\nToday is %s. The fifth working day in a week. Read a book.\n", sDay); break;
        case SAT: printf("\nToday is %s. The first break in a week. Running.\n", sDay); break;
        case SUN: printf("\nToday is %s. Take another break. Go shopping.\n", sDay); break;
        default: printf("\nNo valid day in a week has been input!");
    }

}
```







Add more codes to make such an input valid (?!)

Summary

- Control statements for selection
  - if.. statements
  - if..else.. statements
  - switch..case.. statements
- Statements can be selected for execution according to a “TRUE” (≠ 0) value of a condition (expression).
- Selection statements play an important role in programming.

Chapter 4: Selection Statements