Ngôn ngữ lập trình C++

Chương 2 – Các kiểu dữ liệu cơ bản
Các cấu trúc điều khiển

Tài liệu đọc thêm

- Tài liệu đọc thêm cho chương này:
  - Section 2.1. Complete C++ Language Tutorial (CCLT)
  - Day 7. Teach Yourself C++ in 21 Days (TY21)
  - Namespace (Sec.5-2.CCLT) (Không bắt buộc)

Chương 2 – Kiểu dữ liệu và phép toán cơ bản Cấu trúc điều khiển và cấu trúc chương trình

Đề mục

2.1 Các kiểu dữ liệu cơ bản

2.2 Các phép gán tắt, phép tăng, phép giảm

2.3 Các phép toán logic

2.4 Thuật toán, mã giả, điều khiển của chương trình, sơ đồ khối

2.5 Sơ lược về các cấu trúc điều khiển

2.6 Cấu trúc lựa chọn **`if`**, **`if/else`**

2.7 Phép toán lựa chọn 3 ngôi

2.8 Cấu trúc lặp **`while`**

2.9 Thiết lập thuật toán

2.10 Điều khiển lặp bằng con đếm và giá trị canh

Chương 2 – Kiểu dữ liệu và phép toán cơ bản Cấu trúc điều khiển và cấu trúc chương trình

Đề mục (tiếp theo)

2.11 Các cấu trúc lồng nhau

2.12 Vòng lặp **`for`**

2.13 Cấu trúc đa lựa chọn **`switch`**

2.14 Vòng lặp **`do/while`**

2.15 **`break`** và **`continue`**

2.16 Sơ lược về lập trình cấu trúc

2.1 Các kiểu dữ liệu cơ bản

char	ký tự hoặc số nguyên 8 bit
short	số nguyên 16 bit
long	số nguyên 32 bit
int	số nguyên độ dài bằng 1 word (16 bit hoặc 32 bit)
float	số chấm động 4 byte
double	số chấm động 8 byte
long double	số chấm động 10 byte
bool	giá trị Boolean, true hoặc false
wchar_t	ký tự 2 byte, lưu bảng chữ cái quốc tế

2.2 Các phép toán cơ bản

- phép gán – assignation (=)

```
x = 5;          //x: lvalue, 5: rvalue
```

  - là biểu thức có giá trị là giá trị được gán
- các phép toán số học - Arithmetic operators **(+, -, *, /, %)**
- các phép gán kép - Compound assignation operators **(+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=)**
- phép tăng và phép giảm (++, --)

2.2 Các phép toán cơ bản

- các phép quan hệ - relational operators
  (==, !=, >, <, >=, <=)
- các phép toán logic - Logic operators (**!, &&,** ||)
- phép điều kiện - Conditional operator (**?**).

  (7 == 5 ? 4 : 3) cho kết quả **3** do **7** khác **5**.
- các toán tử bit - Bitwise Operators
  (**&, |, ^, ~, <<, >>**).

2.2 Các phép gán tắt

- Các biểu thức gán tắt - Assignment expression abbreviations
  - Phép gán cộng

    `c = c + 3;` viết tắt thành `c += 3;`
- Các lệnh có dạng

```
variable = variable operator expression;
```

có thể được viết lại thành

```
variable operator= expression;
```

- Các phép gán khác

```
d -= 4      (d = d - 4)
e *= 5      (e = e * 5)
f /= 3      (f = f / 3)
g %= 9      (g = g % 9)
```

2.2 Các phép tăng và giảm

- Phép tăng - Increment operator (`++`)
  - có thể được dùng thay cho `c += 1`
- Phép giảm - Decrement operator (`--`)
  - có thể được dùng thay cho `c -= 1`
- Tăng/giảm trước – Preincrement/Predecrement
  - `++c` hoặc `--c`
  - Giá trị của biến bị thay đổi, sau đó biểu thức chứa nó được tính giá trị.
  - Biểu thức có giá trị là giá trị của biến sau khi tăng/giảm
- Tăng/giảm sau - Postincrement/Predecrement
  - `c++` hoặc `c--`
  - Biểu thức chứa biến được thực hiện, sau đó biến được thay đổi.
  - Biểu thức có giá trị là giá trị của biến trước khi tăng/giảm

2.2 Các phép tăng và giảm

- Ví dụ: nếu `c = 5`
  - `cout << ++c;`
    - `c` nhận giá trị `6`, rồi được in ra
  - `cout << c++;`
    - in giá trị `5` (`cout` được chạy trước phép tăng).
    - sau đó, `c` nhận giá trị `6`

- Khi biến không nằm trong biểu thức
  - Tăng trước và tăng sau có kết quả như nhau

```
++c;
cout << c;
```

và

```
c++;
cout << c;
```

là như nhau

```cpp
// Fig. 2.14: fig02_14.cpp
// Preincrementing and postincrementing.
#include <iostream>

using std::cout;
using std::endl;

// function main begins program execution
int main()
{
   int c;                    // declare variable

   // demonstrate postincrement
   c = 5;                    // assign 5 to c
   cout << c << endl;        // print 5
   cout << c++ << endl;      // print 5 then postincrement
   cout << c << endl << endl; // print 6

   // demonstrate preincrement
   c = 5;                    // assign 5 to c
   cout << c << endl;        // print 5
   cout << ++c << endl;      // preincrement then print 6
   cout << c << endl;        // print 6

   return 0;   // indicate successful termination

} // end function main
```

fig02_14.cpp
(1 of 2)

```
5
5
6

5
6
6
```

2.3 Các phép toán logic

- được dùng làm điều kiện trong các vòng lặp và lệnh if
- **&&** (logical **AND**)
  - **true** nếu cả hai điều kiện là **true**

```
if ( gender == 1 && age >= 65 )
   ++seniorFemales;
```

- **||** (logical **OR**)
  - **true** nếu ít nhất một trong hai điều kiện là **true**

```
if ( semesterAverage >= 90 || finalExam >= 90 )
   cout << "Student grade is A" << endl;
```

2.3 Các phép toán logic

- **!** (logical **NOT**, phủ định logic – logical negation)
  - trả về giá trị **true** khi điều kiện là **false**, và ngược lại

```
if ( !( grade == sentinelValue ) )
   cout << "The next grade is " << grade << endl;
```

tương đương với:

```
if ( grade != sentinelValue )
   cout << "The next grade is " << grade << endl;
```

Nhầm lẫn giữa phép so sánh bằng (==) và phép gán (=)

- Lỗi thường gặp
  - Thường không tạo lỗi cú pháp (syntax error)
- Các khía cạnh của vấn đề
  - biểu thức có giá trị có thể được dùng làm điều kiện
    - bằng không = false, khác không = true
  - Các lệnh gán cũng tạo giá trị (giá trị được gán)

Nhầm lẫn giữa phép so sánh bằng (==) và phép gán (=)

- Ví dụ

```
if ( 4 == payCode )
    cout << "You get a bonus!" << endl;
```

  – Nếu mã tiền lương (paycode) là 4 thì thưởng

- Nếu **==** bị thay bởi **=**

```
if ( payCode = 4 )
   cout << "You get a bonus!" << endl;
```

  – Paycode được gán giá trị 4 (không cần biết giá trị của paycode trước đó)
  – lệnh gán cho giá trị true (vì 4 khác 0)
  – trường hợp nào cũng được thưởng

Nhầm lẫn giữa phép so sánh bằng (==) và phép gán (=)

- Lvalue
  - là biểu thức có thể xuất hiện tại vế trái của phép gán
  - xác định một vùng nhớ có thể được gán trị (i.e, các biến)
    - `x = 4;`
- Rvalue
  - chỉ xuất hiện bên phải phép gán
  - hằng, các giá trị (literal)
    - không thể viết `4 = x;`
- Lvalue có thể được dùng như các rvalue, nhưng chiều ngược lại là không thể

Viết chương trình

- Trước khi viết chương trình
  - Hiểu kỹ bài toán
  - Lập kế hoạch giải quyết bài toán
- Trong khi viết chương trình
  - Biết lời giải có sẵn cho các bài toán con
  - Sử dụng các nguyên lý lập trình tốt

Thuật toán - Algorithm

- Các bài toán tin học
  - được giải bằng cách thực hiện một chuỗi hành động theo một thứ tự cụ thể
- Thuật toán: một quy trình quyết định
  - Các hành động cần thực hiện
  - Thứ tự thực hiện
  - Ví dụ: cách nấu một món ăn
- Điều khiển của chương trình – Program Control
  - Chỉ ra thứ tự thực hiện các lệnh

Mã giả - Pseudocode

- Mã giả: ngôn ngữ không chính thức được dùng để mô tả thuật toán
  - tương tự với ngôn ngữ hàng ngày
- Không chạy được trên máy tính
  - dùng để mô tả chương trình trước khi viết chương trình
    - dễ chuyển thành chương trình C++
  - chỉ gồm các lệnh chạy
    - không cần khai báo biến

```
Ví dụ:
    tìm số nhỏ hơn trong hai số


1.  nhập 2 số x,y
2.  nếu x>y thì in y ra màn hình
3.  nếu không, in x ra màn hình
```

Các cấu trúc điều khiển - Control Structures
Khái niệm

- Thực thi tuần tự - Sequential execution
  - Các lệnh được thực hiện theo thứ tự tuần tự
- Chuyển điều khiển - Transfer of control
  - Lệnh tiếp theo được thực thi *không phải* lệnh tiếp theo trong chuỗi lệnh.
- 3 cấu trúc điều khiển
  - Cấu trúc tuần tự - Sequence structure
    - theo mặc định, chương trình chạy tuần tự từng lệnh
  - Các cấu trúc chọn lựa - Selection structures
    - **`if`**, **`if/else`**, **`switch`**
  - Các cấu trúc lặp - Repetition structures
    - **`while`**, **`do/while`**, **`for`**

Các cấu trúc điều khiển

- Các từ khóa của C++
  - Không thể dùng làm tên biến hoặc tên hàm

C++ Keywords

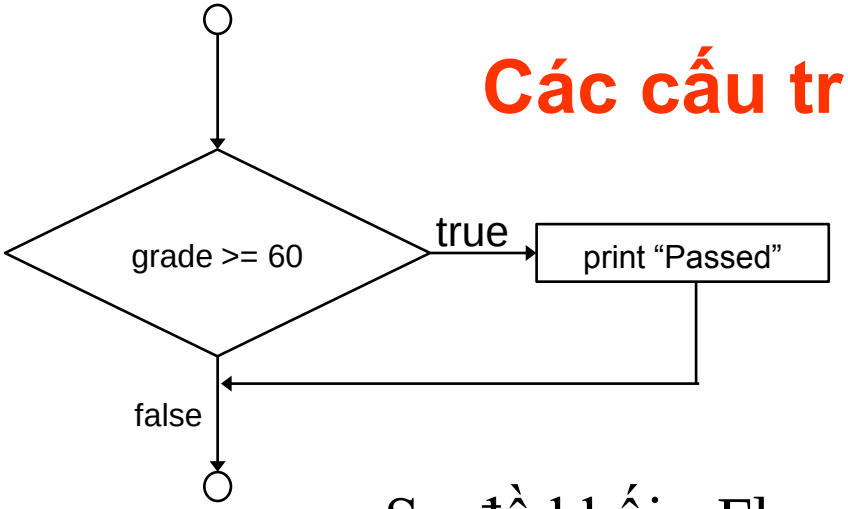
Keywords common to the C and C++ programming languages

auto	break	case	char	const
continue	default	do	double	else
enum	extern	float	for	goto
if	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	while			

C++ only keywords

asm	bool	catch	class	const_cast
delete	dynamic_cast	explicit	false	friend
inline	mutable	namespace	new	operator
private	protected	public	reinterpret_cast	
static_cast	template	this	throw	true
try	typeid	typename	using	virtual
wchar_t				

Các cấu trúc điều khiển



- Sơ đồ khối - Flowchart
  - mô tả thuật toán bằng hình vẽ
  - gồm các ký hiệu đặc biệt được nối bằng các mũi tên (flowlines)
  - Hình chữ nhật (ký hiệu hành động)
    - kiểu hành động bất kỳ
  - ký hiệu oval
    - Bắt đầu hoặc kết thúc một chương trình, hoặc một đoạn mã (hình tròn)
- Các cấu trúc điều khiển có đúng 1 đầu vào, 1 đầu ra
  - Kết nối đầu ra của một cấu trúc điều khiển với đầu vào của cấu trúc tiếp theo
  - xếp chồng các cấu trúc điều khiển

Cấu trúc lựa chọn if

- Cấu trúc lựa chọn - Selection structure
  - chọn giữa các tuyến hành động khác nhau
  - ví dụ bằng mã giả:

    If student's grade is greater than or equal to 60

    Print "Passed"
  - Nếu điều kiện thỏa mãn (có giá trị **`true`**)
    - lệnh Print được thực hiện, chương trình chạy tiếp lệnh tiếp theo
  - Nếu điều kiện không thỏa mãn (có giá trị **`false`**)
    - lệnh Print bị bỏ qua, chương trình chạy tiếp
  - Cách viết thụt đầu dòng làm chương trình dễ đọc hơn
    - C++ bỏ qua các ký tự trắng (tab, space, etc.)

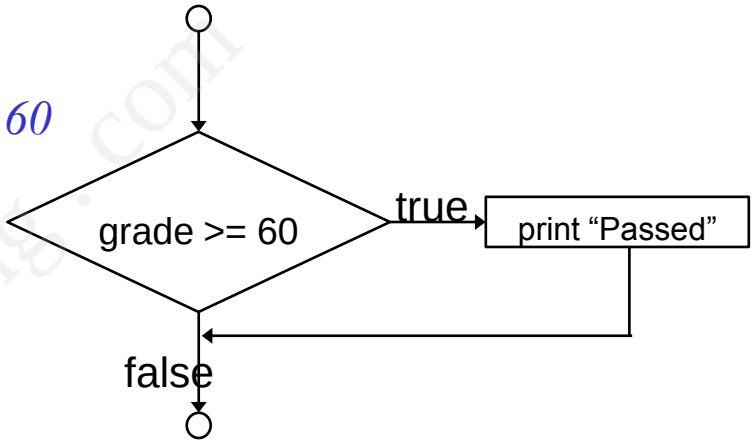
Cấu trúc lựa chọn if

- Dịch sang C++

  If student's grade is greater than or equal to 60

  Print "Passed"

```
if ( grade >= 60 )
   cout << "Passed";
```



- ký hiệu hình thoi (ký hiệu quyết định)
  - đánh đấu chọn lựa cần thực hiện
  - chứa một biểu thức có giá trị true hoặc false
    - kiểm tra điều kiện, đi theo đường thích hợp
- cấu trúc **`if`**
  - Single-entry/single-exit

Một biểu thức bất kỳ đều có thể được sử dụng làm điều kiện cho lựa chọn.

bằng 0 - **`false`**

khác 0 - **`true`**

Ví dụ:

`3 - 4` có giá trị **`true`**

Cấu trúc chọn lựa `if/else`

- **`if`**
  - Thực hiện hành động nếu điều kiện thỏa mãn
- **`if/else`**
  - thực hiện những hành động khác nhau tùy theo điều kiện được thỏa mãn hay không
- mã giả

```
if student's grade is greater than or equal to 60
   print "Passed"
else
   print "Failed"
```

- mã C++

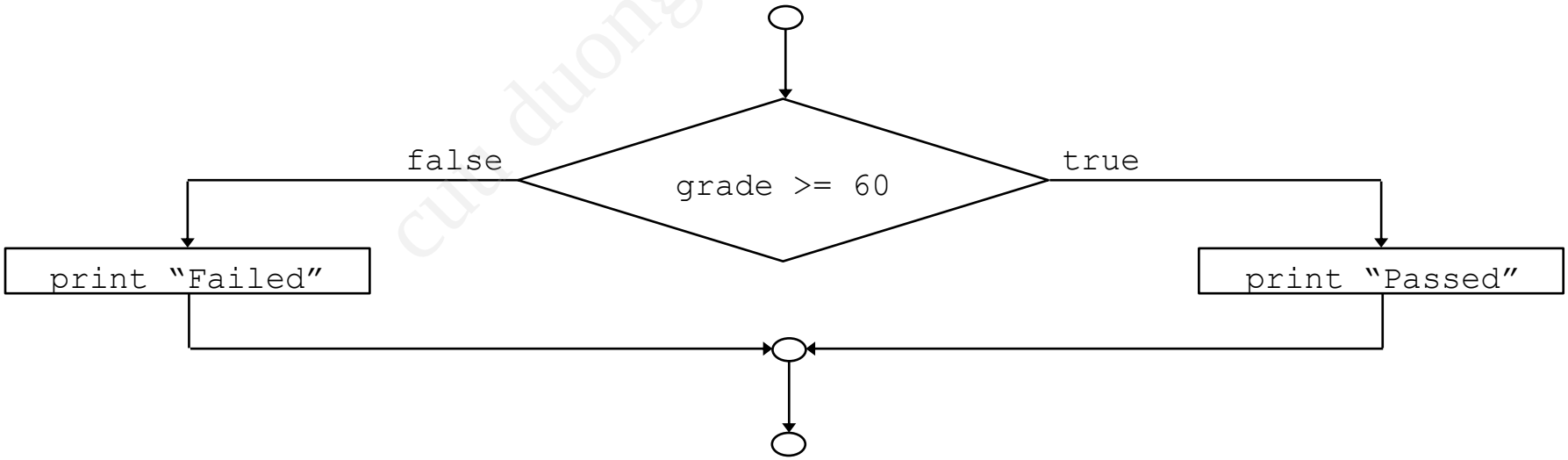
```
if ( grade >= 60 )
   cout << "Passed";
else
   cout << "Failed";
```

Cấu trúc chọn lựa `if/else`

- phép toán điều kiện 3 ngôi (**`?:`**)
  - ba tham số (điều kiện, giá trị nếu **`true`**, giá trị nếu **`false`**)
- mã có thể được viết:

```
cout << ( grade >= 60 ? "Passed" : "Failed" );
```

Condition **Value if true** **Value if false**



Cấu trúc chọn lựa `if/else`

- Các cấu trúc **`if/else`** lồng nhau
  - lệnh này nằm trong lệnh kia, kiểm tra nhiều trường hợp
  - Một khi điều kiện thỏa mãn, các lệnh khác bị bỏ qua

```
if student's grade is greater than or equal to 90
      Print "A"
else
      if student's grade is greater than or equal to 80
        Print "B"
      else
        if student's grade is greater than or equal to 70
          Print "C"
        else
          if student's grade is greater than or equal to 60
            Print "D"
          else
                Print "F"
```

Cấu trúc chọn lựa `if/else`

- Ví dụ

```
if ( grade >= 90 )          // 90 and above
   cout << "A";
else if ( grade >= 80 )     // 80-89
   cout << "B";
else if ( grade >= 70 )     // 70-79
   cout << "C";
else if ( grade >= 60 )     // 60-69
   cout << "D";
else                        // less than 60
   cout << "F";
```

Cấu trúc chọn lựa `if/else`

- lệnh phức – compound statement
  - tập lệnh bên trong một cặp ngoặc

```
if ( grade >= 60 )
   cout << "Passed.\n";
else {
   cout << "Failed.\n";
   cout << "You must take this course again.\n";
}
```

  - nếu không có ngoặc,

```
cout << "You must take this course again.\n";
```

  sẽ luôn được thực hiện
- Khối chương trình - Block
  - tập lệnh bên trong một cặp ngoặc

Cấu trúc lặp while

- Cấu trúc lặp - Repetition structure
  - hành động được lặp đi lặp lại trong khi một điều kiện nào đó còn được thỏa mãn
  - mã giả

    Trong khi vẫn còn tên hàng trong danh sách đi chợ của tôi

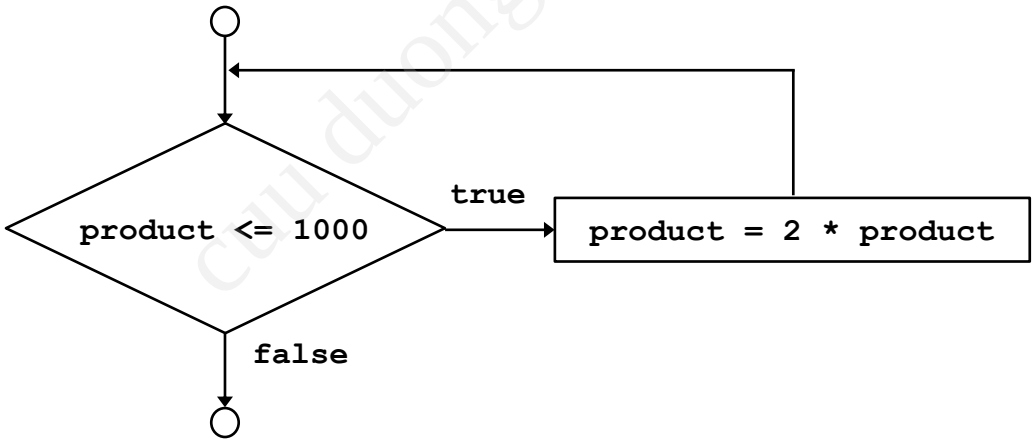
    Mua mặt hàng tiếp theo và gạch tên nó ra khỏi danh sách
  - vòng **`while`** lặp đi lặp lại cho đến khi điều kiện không thỏa mãn

Cấu trúc lặp while

- Ví dụ

```
int product = 2;
while ( product <= 1000 )
   product = 2 * product;
```

- Sơ đồ khối của vòng **while**



Thiết lập thuật toán
(Điều khiển lặp bằng con đếm)

- Vòng lặp được điều khiển bằng con đếm (counter)
  - Lặp đến khi con đếm đạt đến giá trị nào đó
- Lặp hữu hạn - Definite repetition
  - số lần lặp biết trước
- Ví dụ

  Một lớp gồm 10 sinh viên làm một bài thi. Cho biết các điểm thi (số nguyên trong khoảng từ 0 đến 100). Tính trung bình điểm thi của lớp.

Thiết lập thuật toán
(Điều khiển lặp bằng con đếm)

- Mã giả cho ví dụ:

Đặt tổng bằng 0

Đặt con đếm bằng 1

Trong khi con đếm nhỏ hơn hoặc bằng 10

Nhập điểm tiếp theo

Cộng điểm đó vào tổng

Thêm 1 vào con đến

Đặt trung bình lớp bằng tổng chia cho 10

In trung bình lớp

- Tiếp theo: Mã C++ cho ví dụ trên

```
1  // Fig. 2.7: fig02_07.cpp
2  // Class average program with counter-controlled repetition.
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  // function main begins program execution
10 int main()
11 {
12    int total;        // sum of grades input by user
13    int gradeCounter; // number of grade to be entered next
14    int grade;        // grade value
15    int average;      // average of grades
16
17    // initialization phase
18    total = 0;          // initialize total
19    gradeCounter = 1;   // initialize loop counter
20
```

fig02_07.cpp
(1 of 2)

```
21     // processing phase
22     while ( gradeCounter <= 10 ) {        // loop 10 times
23        cout << "Enter grade: ";           // prompt for input
24        cin >> grade;                      // read grade from user
25        total = total + grade;             // add grade to total
26        gradeCounter = gradeCounter + 1;   // increment counter
27     }
28
29     // termination phase
30     average = total / 10;
31
32     // display result
33     cout << "Class average is " << average << endl;
34
35     return 0;   // indicate program ended successfully
36
37  } // end function main
```

fig02_07.cpp (2 of 2)

fig02_07.cpp output (1 of 1)

Con đếm được tăng thêm 1 mỗi lần vòng lặp chạy. Cuối cùng, con đếm làm vòng lặp kết thúc.

```
Enter grade: 98
Enter grade: 76
Enter grade: 71
Enter grade: 87
Enter grade: 83
Enter grade: 90
Enter grade: 57
Enter grade: 79
Enter grade: 82
Enter grade: 94
Class average is 81
```

Thiết lập thuật toán
(Điều khiển lặp bằng lính canh)

- Giả sử bài toán trở thành:

  Viết một chương trình tính điểm trung bình của lớp, chương trình sẽ xử lý một số lượng điểm tùy ý mỗi khi chạy chương trình.

  - Số sinh viên chưa biết
  - Chương trình sẽ làm thế nào để biết khi nào thì kết thúc?
- Giá trị canh
  - Ký hiệu "Kết thúc của dữ liệu vào"
  - Vòng lặp kết thúc khi nhập canh
  - Canh được chọn để không bị lẫn với dữ liệu vào thông thường
    - trong trường hợp này là -1

Thiết lập thuật toán
(Điều khiển lặp bằng lính canh)

- Thiết kế từ trên xuống, làm mịn từng bước
  - Bắt đầu bằng mã giả cho mức cao nhất

    Tính trung bình điểm thi của lớp
  - Chia thành các nhiệm vụ nhỏ hơn, liệt kê theo thứ tự

    Khởi tạo các biến

    Nhập, tính tổng, và đếm các điểm thi

    Tính và in trung bình điểm thi

Thiết lập thuật toán
(Điều khiển lặp bằng lính canh)

- Nhiều chương trình có 3 pha
  - Khởi tạo - Initialization
    - Khởi tạo các biến chương trình
  - Xử lý - Processing
    - Nhập dữ liệu, điều chỉnh các biến trong chương trình
  - Kết thúc - Termination
    - Tính và in kết quả cuối cùng
  - Giúp việc chia nhỏ chương trình để làm mịn từ trên xuống

Thiết lập thuật toán
(Điều khiển lặp bằng lính canh)

- Làm mịn pha khởi tạo

  Khởi tạo các biến

  thành

  Khởi tạo tổng bằng 0

  Khởi tạo biến đếm bằng 0

- Xử lý

  Nhập, tính tổng, và đếm các điểm thi

  thành

  Nhập điểm đầu tiên (có thể là canh)

  Trong khi người dùng còn chưa nhập canh

  Cộng điểm vừa nhập vào tổng

  Cộng thêm 1 vào biến đếm điểm

  Nhập điểm tiếp theo (có thể là canh)

Thiết lập thuật toán (Điều khiển lặp bằng lính canh)

- Kết thúc

Tính và in trung bình điểm thi

thành

Nếu con đếm khác 0

Đặt trung bình bằng tổng chia cho con đếm

In giá trị trung bình

Nếu không

In "Không nhập điểm nào"

- Tiếp theo: chương trình C++

```
1  // Fig. 2.9: fig02_09.cpp
2  // Class average program with sentinel-controlled repetition.
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8  using std::fixed;
9
10 #include <iomanip>          // parameterized stream manipulators
11
12 using std::setprecision;  // sets numeric output precision
13
14 // function main begins program execution
15 int main()
16 {
17    int total;           // sum of
18    int gradeCounter;    // number
19    int grade;           // grade value
20
21    double average;      // number with decimal point for average
22
23    // initialization phase
24    total = 0;           // initialize total
25    gradeCounter = 0;    // initialize loop counter
```

Dữ liệu kiểu **double** dùng để biểu diễn số thập phân.

fig02_09.cpp (1 of 3)

```
26
27    // processing phase
28    // get first grade from user
29    cout << "Enter grade, -1 to end: ";  // prompt for input
30    cin >> grade;                        // read grade from user
31
32    // loop until sentinel value read from user
33    while ( grade != -1 ) {
34       total = total + grade;            // add grade to total
35       gradeCounter = gradeCounter + 1;  // increment counter
36
37       cout << "Enter grade, -1 to end: ";  // prompt for input
38       cin >> grade;                       // read next grade
39
40    } // end while
41
42    // termination phase
43    // if user entered at least one grade ...
44    if ( gradeCounter != 0 ) {
45
46       // calculate average of all grades entered
47       average = static_cast< double >( total ) / gradeCounter;
48
```

fig02_09.cpp
(2 of 3)

`static_cast<double>()` coi **`total`** như một **`double`** tạm thời (casting).
Cần thiết vì phép chia số nguyên bỏ qua phần dư.
`gradeCounter` là một biến **`int`**, nhưng nó được nâng lên kiểu **`double`**.

```
49          // display average with two digits of precision
50          cout << "Class average is " << setprecision( 2 )
51               << fixed << average << endl;
52
53       } // end if part of if/else
54
55       else // if no grades were entered, output appropriate message
56          cout << "No grades were entered" << endl;
57
58       return 0;   // indicate program ended successfully
59
60    } // end function main
```

fig02_09.cpp (3 of 3)

fig02_09.cpp output (1 of 1)

```
Enter grade, -1 to end: 75
Enter grade, -1 to end: 94
Enter grade, -1 to end: 97
Enter grade, -1 to end: 88
Enter grade, -1 to end: 70
Enter grade, -1 to end: 64
Enter grade, -1 to end: 83
Enter grade, -1 to end: 89
Enter grade, -1 to end: -1
Class average is 82.50
```

`fixed` làm số liệu ra được in theo dạng thông thường (không phải dạng ký hiệu khoa học); qui định in cả các chữ số 0 ở sau và in dấu chấm thập phân.

Include **`<iostream>`**

`setprecision(2)` in hai chữ số sau dấu phảy (làm tròn theo độ chính xác quy định).

Các chương trình dùng hàm này phải include **`<iomanip>`**

Các cấu trúc điều khiển lồng nhau

- Phát biểu bài toán

  Một trường có danh sách kết quả thi (1 = đỗ, 2 = trượt) của 10 sinh viên. Viết một chương trình phân tích kết quả thi. Nếu có nhiều hơn 8 sinh viên đỗ thì in ra màn hình dòng chữ *"Tăng tiền học phí".*

- Lưu ý
  - Chương trình xử lý 10 kết quả thi
    - số lần lặp cố định, sử dụng vòng lặp điều khiển bằng biến đếm
  - Có thể sử dụng hai con đếm
    - Một con đếm để đếm số lượng đỗ
    - Một con đếm khác đếm số lương trượt
  - Mỗi kết quả thi chỉ là 1 hoặc 2
    - Nếu không phải 1 thì coi là 2

Các cấu trúc điều khiển lồng nhau

- Phác thảo mức cao nhất - Top level outline

  Analyze exam results and decide if tuition should be raised

- Làm mịn lần một - First refinement

  Initialize variables

  Input the ten quiz grades and count passes and failures

  Print a summary of the exam results and decide if tuition should be raised

- Làm mịn - Refine

  Initialize variables

  to

  Initialize passes to zero

  Initialize failures to zero

  Initialize student counter to one

Các cấu trúc điều khiển lồng nhau

- Refine

Input the ten quiz grades and count passes and failures

to

While student counter is less than or equal to ten
Input the next exam result
If the student passed
Add one to passes
Else
Add one to failures
Add one to student counter

Các cấu trúc điều khiển lồng nhau

- tiếp tục làm mịn

  Print a summary of the exam results and decide if tuition should be raised

  to

  Print the number of passes

  Print the number of failures

  If more than eight students passed

  Print "Raise tuition"

- Program next

```cpp
// Fig. 2.11: fig02_11.cpp
// Analysis of examination results.
#include <iostream>

using std::cout;
using std::cin;
using std::endl;

// function main begins program execution
int main()
{
   // initialize variables in declarations
   int passes = 0;           // number of passes
   int failures = 0;         // number of failures
   int studentCounter = 1;   // student counter
   int result;               // one exam result

   // process 10 students using counter-controlled loop
   while ( studentCounter <= 10 ) {

      // prompt user for input and obtain value from user
      cout << "Enter result (1 = pass, 2 = fail): ";
      cin >> result;

```

fig02_11.cpp
(1 of 2)

```
25       // if result 1, increment passes; if/else nested in while
26       if ( result == 1 )        // if/else nested in while
27          passes = passes + 1;
28
29       else  // if result not 1, increment failures
30          failures = failures + 1;
31
32       // increment studentCounter so loop eventually terminates
33       studentCounter = studentCounter + 1;
34
35    } // end while
36
37    // termination phase; display number of passes and failures
38    cout << "Passed " << passes << endl;
39    cout << "Failed " << failures << endl;
40
41    // if more than eight students passed, print "raise tuition"
42    if ( passes > 8 )
43       cout << "Raise tuition " << endl;
44
45    return 0;   // successful termination
46
47 } // end function main
```

fig02_11.cpp
(2 of 2)

```
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Passed 6
Failed 4

Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Passed 9
Failed 1
Raise tuition
```

fig02_11.cpp
output (1 of 1)

Những điểm quan trọng về vòng lặp điều khiển bằng con đếm

- vòng lặp điều khiển bằng con đếm đòi hỏi
  - Tên của biến điều khiển(control variable) hay biến đếm (loop counter)
  - Giá trị khởi tạo của biến điều khiển
  - Điều kiện kiểm tra giá trị cuối cùng
  - Tăng/giảm biến đếm khi thực hiện vòng lặp

```
int counter = 1;                    // initialization

while ( counter <= 10 ) {           // repetition condition
   cout << counter << endl;         // display counter
   ++counter;                       // increment
}
```

```
1  // Fig. 2.16: fig02_16.cpp
2  // Counter-controlled repetition.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  // function main begins program execution
9  int main()
10 {
11    int counter = 1;              // initialization
12
13    while ( counter <= 10 ) {     // repetition condition
14       cout << counter << endl;   // display counter
15       ++counter;                 // increment
16
17    } // end while
18
19    return 0;   // indicate successful termination
20
21 } // end function main
```

fig02_16.cpp
(1 of 1)

```
1
2
3
4
5
6
7
8
9
10
```

Cấu trúc vòng lặp for

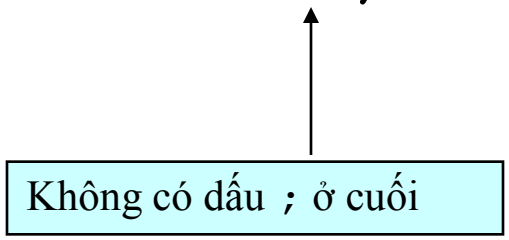
- Dạng tổng quát của vòng **for**

```
for ( khởi_tạo; điều_kiện_lặp; tăng/giảm )
    lệnh
```

- Ví dụ

```
for( int counter = 1; counter <= 10; counter++ )
    cout << counter << endl;
```

  – In các số nguyên từ 1 đến 10



```
1  // Fig. 2.17: fig02_17.cpp
2  // Counter-controlled repetition with the for structure.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  // function main begins program execution
9  int main()
10 {
11    // Initialization, repetition condition and incrementing
12    // are all included in the for structure header.
13
14    for ( int counter = 1; counter <= 10; counter++ )
15       cout << counter << endl;
16
17    return 0;   // indicate successful termination
18
19 } // end function main
```

fig02_17.cpp
(1 of 1)

```
1
2
3
4
5
6
7
8
9
10
```

Cấu trúc vòng lặp for

- vòng **for** thường có thể viết được thành vòng **while** tương đương

```
khởi_tạo;
while ( điều_kiện_lặp){
   lệnh
   tăng/giảm biến đếm;
}
```

- Khởi tạo và tăng biến đếm
  - nếu sử dụng nhiều biến đếm, sử dụng dấu phảy để tách

```
for (int i = 0, j = 0;  j + i <= 10; j++, i++)
   cout << j + i << endl;
```

```
1  // Fig. 2.20: fig02_20.cpp
2  // Summation with for.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  // function main begins program execution
9  int main()
10 {
11    int sum = 0;                        // initialize sum
12
13    // sum even integers from 2 through 100
14    for ( int number = 2; number <= 100; number += 2 )
15       sum += number;                   // add number to sum
16
17    cout << "Sum is " << sum << endl;   // output sum
18    return 0;                           // successful termination
19
20 } // end function main
```

```
Sum is 2550
```

2_20.cpp
1)

2_20.cpp
ut (1 of 1)

Ví dụ sử dụng vòng for

- Chương trình tính lãi kép (compound interest)
- *Một người đầu tư $1000.00 vào một tài khoản tiết kiệm với lãi suất 5%. Giả sử tiền lãi được gộp với vốn trong tài khoảng, tính và in ra số tiền trong tài khoản vào cuối mỗi năm trong vòng 10 năm. Sử dụng công thức sau để tính các khoản tiền đó:*

$$a = p(1+r)^n$$

- p : khoản đầu tư ban đầu (i.e., the principal),
  r : lãi suất hàng năm, (interest rate)
  n : số năm, và
  a : lượng tiền có trong tài khoản (amount on deposit)
  vào cuối năm thứ n

```
1  // Fig. 2.21: fig02_21.cpp
2  // Calculating compound interest.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7  using std::ios;
8  using std::fixed;
9
10 #include <iomanip>
11
12 using std::setw;
13 using std::setprecision;
14
15 #include <cmath>  // enables program to use function pow
16
17 // function main begins program execution
18 int main()
19 {
20    double amount;               // amount on deposit
21    double principal = 1000.0;   // starting principal
22    double rate = .05;           // interest rate
23
```

<cmath> header cần cho hàm **pow** (chương trình sẽ không dịch nếu không có khai báo này).

fig02_21.cpp
(1 of 2)

```
24     // output table column heads
25     cout << "Year" << setw( 21 ) << "Amount on deposit" << endl;
26
27     // set floating-point number format
28     cout << fixed << setprecision( 2 );
29
30     // calculate amount on deposit for each of ten years
31     for ( int year = 1; year <= 10; year++ ) {
32
33        // calculate new amount for specified year
34        amount = principal * pow( 1.0 + rate, year );
35
36        // output one table row
37        cout << setw( 4 ) << year
38             << setw( 21 ) << amount << endl;
39
40     } // end for
41
42     return 0;   // indicate successful termination
43
44  } // end function main
```

Đặt độ rộng của output ít nhất 21 ký tự. Nếu output ít hơn 21 ký tự thì căn phải.

pow(x,y) = x mũ y

```
Year    Amount on deposit
   1              1050.00
   2              1102.50
   3              1157.63
   4              1215.51
   5              1276.28
   6              1340.10
   7              1407.10
   8              1477.46
   9              1551.33
  10              1628.89
```

Các số được căn phải do các lệnh setw (với tham số có giá trị 4 và 21).

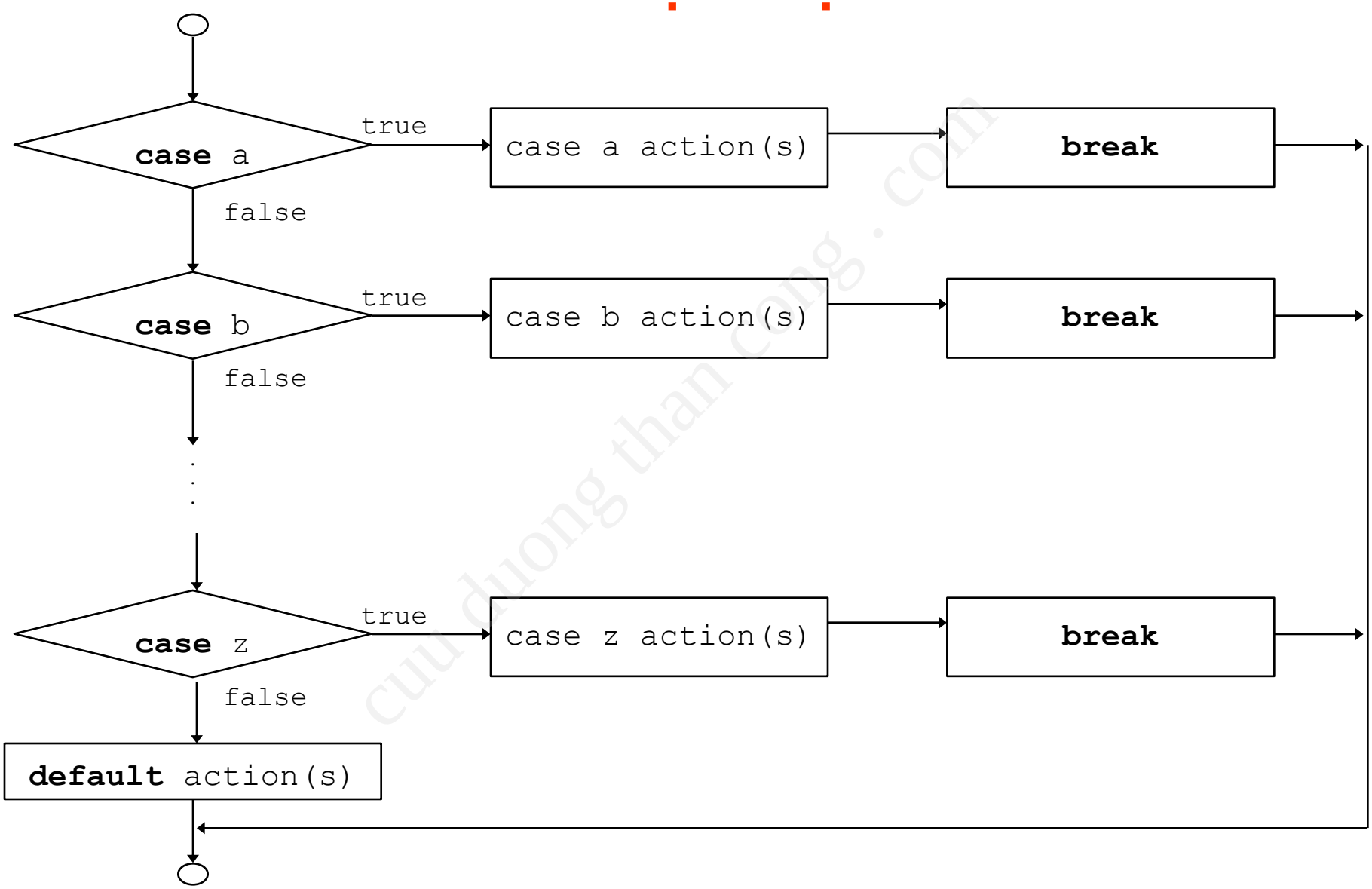
fig02_21.cpp
output (1 of 1)

Cấu trúc đa lựa chọn switch

- **`switch`**
  - Test biến với nhiều giá trị
  - chuỗi các nhãn **`case`**
  - trường hợp **`default`** không bắt buộc

```
switch ( variable ) {
     case value1:         // taken if variable == value1
     statements
     break;              // necessary to exit switch

     case value2:
     case value3:    // taken if variable == value2 or == value3
     statements
     break;

     default:         // taken if variable matches no other cases
     statements
     break;
}
```

Cấu trúc đa lựa chọn switch



Cấu trúc đa lựa chọn switch

- Ví dụ sắp tới
  - Chương trình đọc xếp loại điểm (A-F)
  - Hiện số lượng mỗi xếp loại được nhập
- Chi tiết về các ký tự
  - Các ký tự đơn thường được lưu bằng kiểu dữ liệu **`char`**
    - **`char:`** số nguyên 1-byte, →có thể được lưu dưới dạng các giá trị **`int`**
  - Có thể coi ký tự là **`int`** hoặc **`char`**
    - 97 là biểu diễn dạng số của chữ 'a' thường (ASCII)
    - dùng cặp nháy đơn để lấy biểu diễn chữ của ký tự

```
cout << "The character (" << 'a' << ") has the value "
       << static_cast< int > ( 'a' ) << endl;
```

In ra dòng:

```
The character (a) has the value 97
```

```
1  // Fig. 2.22: fig02_22.cpp
2  // Counting letter grades.
3  #include <iostream>
4
5  using std::cout;
6  using std::cin;
7  using std::endl;
8
9  // function main begins program execution
10 int main()
11 {
12    int grade;        // one grade
13    int aCount = 0;   // number of As
14    int bCount = 0;   // number of Bs
15    int cCount = 0;   // number of Cs
16    int dCount = 0;   // number of Ds
17    int fCount = 0;   // number of Fs
18
19    cout << "Enter the letter grades." << endl
20         << "Enter the EOF character to end input." << endl;
21
```

fig02_22.cpp
(1 of 4)

```
22     // loop until user types end-of-file key sequence
23     while ( ( grade = cin.get() ) != EOF )
24
25        // determine which grade was input
26        switch ( grade ) {  // switch structure nested in while
27
28           case 'A':        // grade was uppercase A
29           case 'a':        // or lowercase a
30              ++aCount;     // increment aCount
31              break;        // necessary to exit swit
32
33           case 'B':                                    B
34           case 'b':
35              ++bCount;
36              break;
37
38           case 'C':                                    C
39           case 'c':
40              ++cCount;
41              break;
42
```

break kết thúc lệnh **switch** và chương trình chạy tiếp tại lệnh đầu tiên sau cấu trúc **switch**.

fig02_22.cpp
(2 of 4)

cin.get() sử dụng dot notation (ký hiệu kiểu dấu chấm). Hàm này đọc một ký tự từ bàn phím (sau khi nhấn *Enter*), và gán giá trị đó cho biến **grade**.

cin.get() trả về EOF (end-of-file), sau khi ký tự EOF được nhập, để đánh dấu kết thúc của dữ liệu vào. EOF có thể là ctrl-d hoặc ctrl-z, tùy theo hệ điều hành. (MS-Windows: ctrl-z, Unix/Linux: ctrl-d)

Các lệnh gán là biểu thức có giá trị bằng biến bên trái dấu gán =. Giá trị của lệnh này bằng giá trị trả về bởi hàm **cin.get()**.

Đặc điểm này còn được sử dụng để khởi tạo nhiều biến một lúc:
a = b = c = 0;

So sánh **grade** (một biến **int**) với biểu diễn số của **A** và **a**.

```
43          case 'D':          // grade was uppercase D
44          case 'd':          // or lowercase d
45             ++dCount;       // increment dCount
46             break;          // exit switch
47
48          case 'F':          // grade was up
49          case 'f':          // or lowercase
50             ++fCount;       // increment fC
51             break;          // exit switch
52
53          case '\n':         // ignore newlines,
54          case '\t':         // tabs,
55          case ' ':          // and spaces in input
56             break;          // exit switch
57
58          default:           // catch all other characters
59             cout << "Incorrect letter grade entered."
60                  << " Enter a new grade." << endl;
61             break;          // optional; will exit switch anyway
62
63       } // end switch
64
65    } // end while
66
```

fig02_22.cpp

Kiểm tra này là cần thiết vì *Enter* được nhấn sau mỗi chữ cái xếp loại được nhập. Việc nhấn *Enter* tạo một ký tự xuống dòng cần được loại bỏ. Cũng như vậy, ta muốn bỏ qua các ký tự trắng.

Lưu ý trường hợp **`default`** bao gồm tất cả các trường hợp còn lại (chưa xét đến).

```
67     // output summary of results
68     cout << "\n\nTotals for each letter grade are:"
69          << "\nA: " << aCount   // display number of A grades
70          << "\nB: " << bCount   // display number of B grades
71          << "\nC: " << cCount   // display number of C grades
72          << "\nD: " << dCount   // display number of D grades
73          << "\nF: " << fCount   // display number of F grades
74          << endl;
75
76     return 0;  // indicate successful termination
77
78  } // end function main
```

fig02_22.cpp
(4 of 4)

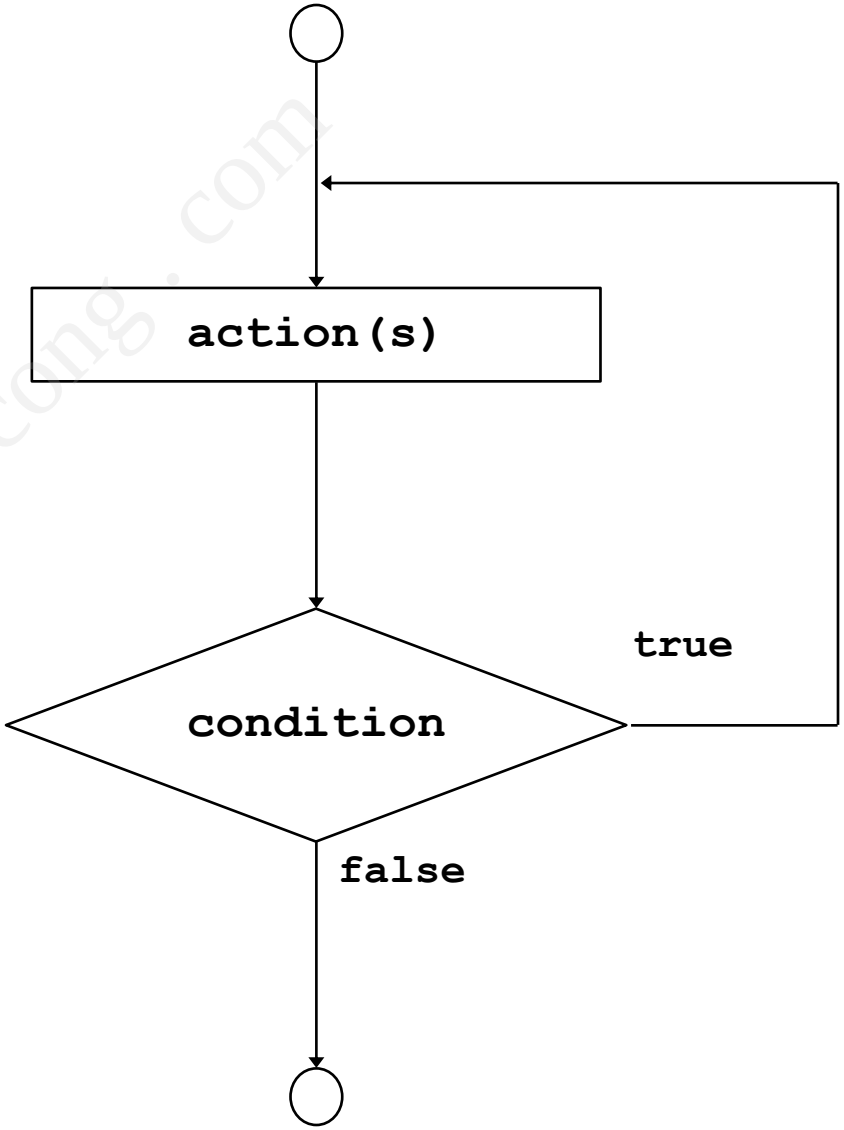
```
Enter the letter grades.
Enter the EOF character to end input.
a
B
c
C
A
d
f
C
E
Incorrect letter grade entered. Enter a new grade.
D
A
b
^Z


Totals for each letter grade are:
A: 3
B: 2
C: 3
D: 2
F: 1
```

fig02_22.cpp
output (1 of 1)

Cấu trúc lặp do/while

- Tương tự cấu trúc **`while`**
  - Kiểm tra điều kiện tiếp tục lặp ở cuối, không kiểm tra ở đầu
  - Thân vòng lặp chạy ít nhất một lần
- Công thức

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



```
1  // Fig. 2.24: fig02_24.cpp
2  // Using the do/while repetition structure.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  // function main begins program
9  int main()
10 {
11    int counter = 1;              // initialize counter
12
13    do {
14       cout << counter << "  ";   // display counter
15    } while ( counter++ <= 10 ); // end do/while
16
17    cout << endl;
18
19    return 0;   // indicate successful termination
20
21 } // end function main
```

Chú ý phép tăng trước (preincrement) trong phần kiểm tra điều kiện lặp.

```
1  2  3  4  5  6  7  8  9  10
```

fig02_24.cpp (1 of 1)

fig02_24.cpp output (1 of 1)

Các lệnh break và continue

- **`break`**
  - Thoát ngay ra khỏi các cấu trúc **`while`**, **`for`**, **`do/while`**, **`switch`**
  - Chương trình tiếp tục chạy tại lệnh đầu tiên ngay sau cấu trúc
- thường được sử dụng để
  - Thoát khỏi vòng lặp sớm hơn bình thường
  - bỏ qua phần còn lại của **`switch`**

```
1  // Fig. 2.26: fig02_26.cpp
2  // Using the break statement in a for structure.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  // function main begins program execution
9  int main()
10 {
11
12    int x;  // x declared here so it can be used after the loop
13
14    // loop 10 times
15    for ( x = 1; x <= 10; x++ ) {
16
17       // if x is 5, terminate loop
18       if ( x == 5 )
19          break;             // break loop only if x is 5
20
21       cout << x << " ";    // display value of x
22
23    } // end for
24
25    cout << "\nBroke out of loop when x became " << x << endl;
26
27    return 0;   // indicate successful termination
28
29 } // end function main
```

```
1 2 3 4
Broke out of loop when x became 5
```

Thoát khỏi vòng **for** khi **break** được thực thi.

Các lệnh break và continue

- **`continue`**
  - được dùng trong **`while`**, **`for`**, **`do/while`**
  - bỏ qua phần còn lại của thân vòng lặp
  - chạy tiếp lần lặp tiếp theo
- với các vòng **`while`** và **`do/while`**
  - thực hiện kiểm tra điều kiện lặp ngay sau lệnh **`continue`**
- với vòng **`for`**
  - biểu thức tăng/giảm biến đếm được thực hiện
  - sau đó, điều kiện lặp được kiểm tra

```
1  // Fig. 2.27: fig02_27.cpp
2  // Using the continue statement in a for structure.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  // function main begins program execution
9  int main()
10 {
11    // loop 10 times
12    for ( int x = 1; x <= 10; x++ ) {
13
14       // if x is 5, continue with next iteration of loop
15       if ( x == 5 )
16          continue;        // skip remaining code in loop body
17
18       cout << x << " ";   // display value of x
19
20    } // end for structure
21
22    cout << "\nUsed continue to skip printing the value 5"
23         << endl;
24
25    return 0;              // indicate successful termination
26
27 } // end function main
```

```
1 2 3 4 6 7 8 9 10
Used continue to skip printing the value 5
```

Bỏ qua phần còn lại của thân vòng **for**, nhảy đến lần lặp tiếp theo.

Lập trình cấu trúc
Structured-Programming

- Lập trình cấu trúc – Structured programming
  - Chương trình dễ hiểu, test, tìm lỗi (debug) và dễ sửa đổi hơn
- Các quy tắc lập trình cấu trúc
  - Chỉ sử dụng các cấu trúc điều khiển một đầu vào một đầu ra
  - Quy tắc
    1) Bắt đầu bằng một sơ đồ khối đơn giản nhất
    2) Mỗi hình chữ nhật (hành động) có thể được thay bằng một chuỗi gồm 2 hình chữ nhật khác
    3) Mỗi hình chữ nhật (hành động) có thể được thay bằng một cấu trúc điều khiển tùy ý (tuần tự, if, if/else, switch, while, do/while hoặc for)
    4) Các quy tắc 2 và 3 có thể được áp dụng nhiều lần và theo thứ tự tùy ý

Lập trình cấu trúc
Structured-Programming

Mô tả quy tắc 3 (thay một hình chữ nhật tùy ý bằng một cấu trúc điều khiển)