Introduction to C++ Programming

Lecture 1

Translate by: Đỗ Thị Bích Ngọc

https://www.facebook.com/groups/CPP4PTIT/

Quy định về môn học

- Nghỉ quá 20% số giờ của môn học, HOẶC
- Thiếu một điểm thành phần (bài tập lớn, kiểm tra giữa kỳ)
  => KHÔNG ĐƯỢC THI

Hình thức kiểm tra *(Tham khảo ví dụ dưới đây)*	**Tỷ lệ đánh giá**	**Đặc điểm đánh giá**
- Đi học đầy đủ (trong lớp gây ảnh hưởng đến người khác, mỗi lần nhắc nhở trừ một điểm, mỗi buổi nghỉ học trừ một điểm)	5 %	Cá nhân
- Tích cực thảo luận (không phát biểu buổi nào sẽ được 0 điểm, phát biểu 1 buổi sẽ được 4 điểm, sau đó số buổi học có phát biểu tăng lên 1 thì điểm tăng lên 1)	5 %	Cá nhân
- Trung bình các điểm bài tập lớn	20%	Cá nhân/Nhóm
- Trung bình các bài kiểm tra trên lớp	20%	Cá nhân
- Kiểm tra cuối kỳ	50%	Cá nhân

Tài liệu tham khảo

- Sách
  - ***C++ How to Program***- 7th Edition, by H. M. Deitel, P. J. Deitel, Prentice Hall, New Jersey, 2010, ISBN: 0-13-038474.
  - Bài giảng ngôn ngữ lập trình C++ của thầy Hùng và thầy Sơn

- Trao đổi về C++ qua Facebook group
  https://www.facebook.com/groups/CPP4PTIT/

Ngôn ngữ máy

- Ngôn ngữ máy
  - máy tính có thể hiểu được
  - định nghĩa dựa trên thiết kế phần cứng
    - Phụ thuộc vào máy
  - Thường là chuỗi số gồm 0, 1
  - Điều khiển máy tính thực hiện từng thao tác một
  - Khó hiểu với con người
  - Ví dụ:

```
+1300042774
+1400593419
+ß1200274027
```

Ngôn ngữ máy

- Hợp ngữ - Assembly language
  - Dùng các từ tiếng Anh để biểu diễn các thao tác máy tính
  - Dễ hiểu hơn cho con người
  - Máy tính không hiểu được
    - Cần chương trình dịch (assemblers)
      - Chuyển hợp ngữ sang mã máy
  - Ví dụ:

```
LOAD  BASEPAY
ADD   OVERPAY
STORE GROSSPAY
```

Ngôn ngữ máy

- Ngôn ngữ lập trình cấp cao
  - Giống tiếng Anh, sử dụng các ký hiệu toán học
  - Một câu lệnh bao hàm một tập các thao tác
    - Hợp ngữ yêu cầu nhiều thao tác để hoàn thành một nhiệm vụ đơn giản
  - Chương trình dịch (Trình biên dịch - compilers)
    - Dịch sang ngôn ngữ máy
  - Trình thông dịch
    - trực tiếp thực hiện chương trình viết bằng ngôn ngữ lập trình cấp cao
  - Ví dụ:

```
grossPay = basePay + overTimePay
```

Giới thiệu về C

- Giới thiệu về C
  - Kết hợp của 2 ngôn ngữ lập trình
    - BCPL và B
      - Ngôn ngữ dùng ít kiểu hơn - “Typeless” languages
  - Dennis Ritchie (Bell Laboratories)
    - Thêm khả năng định nghĩa kiểu và chức năng khác
  - Dùng được với hệ điều hành UNIX
  - Độc lập với phần cứng
    - Có thể di chuyển được chương trình
  - 1989: chuẩn ANSI
  - 1990: phát hành chuẩn ANSI và ISO
    - ANSI/ISO 9899: 1990

Giới thiệu về C++ (ra đời 1980)

- Mở rộng từ C
- Có khả năng lập trình hướng đối tượng (object-oriented programming)
  - Đối tượng: tái sử dụng các thành phần phần mềm
    - Biểu diễn các sự vật trong thực tế
  - Lập trình hướng đối tượng
    - Dễ hiểu, dễ sửa lỗi, dễ thay đổi

Thư viện chuẩn C++

- Chương trình C++
  - Tạo bởi các lớp (class) và hàm (function)
- Thư viện chuẩn C++
  - Tập hợp phong phú các lớp và hàm sẵn có
- "Building block approach" để tạo chương trình
  - Tái sử dụng phần mềm ("Software reuse")

Hướng đối tượng: xu thế phát triển phần mềm phổ biến

- Đối tượng
  - Có thể tái sử dụng các thành phần phần mềm
  - Các thành phần/đơn vi phần mềm có nghĩa
    - Đối tượng Date, time, video, file, record…
    - Bất kỳ danh từ nào có thể được biểu diễn như là đối tượng
  - Dễ hiểu, dễ tổ chức, dễ bảo trì hơn ngôn ngữ lập trình hàm

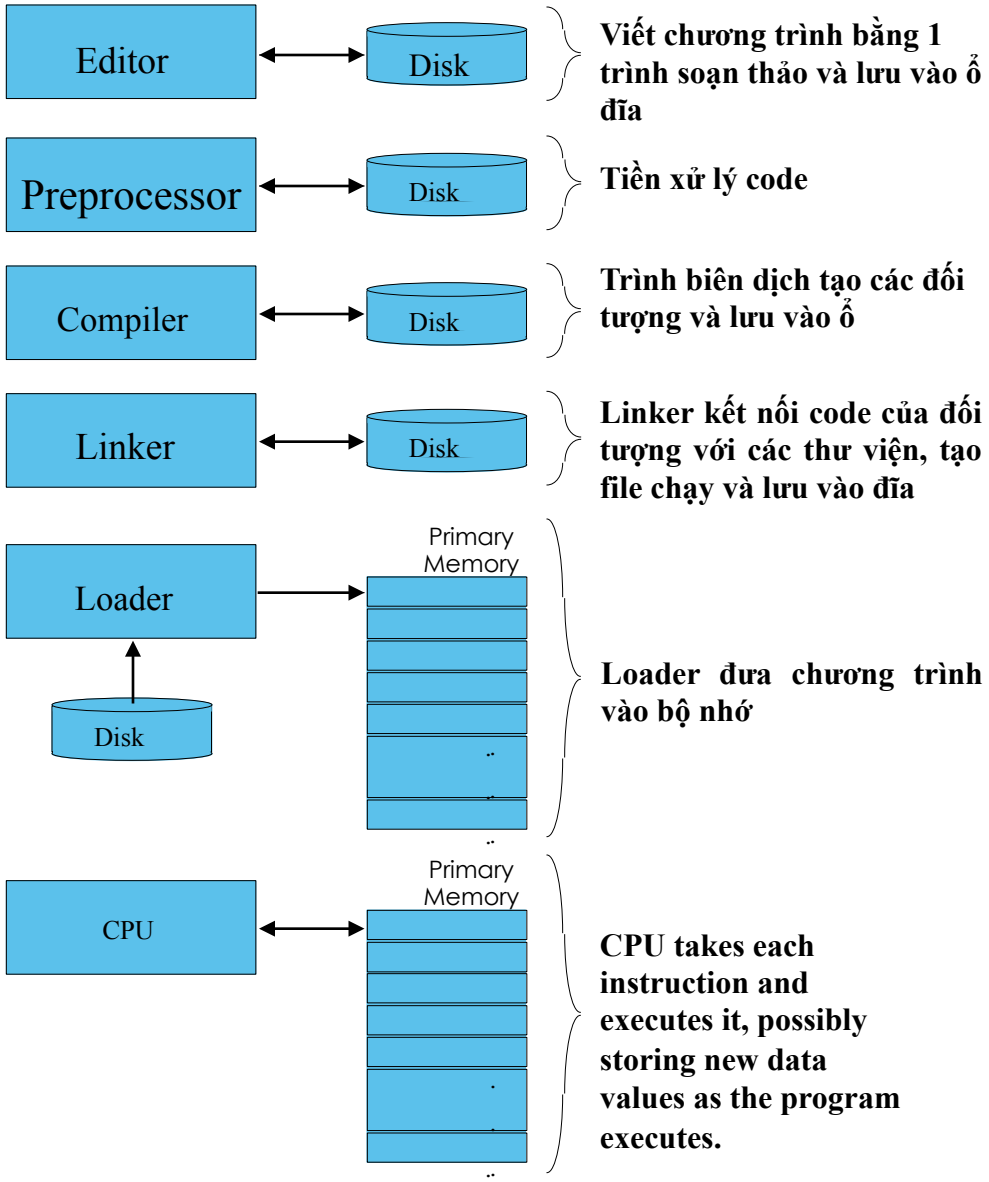
Môi trường điển hình của C++

- Hệ thống C++
  - Môi trường lập trình
  - Ngôn ngữ
  - Thư viện chuẩn
- Phần mở rộng tên file C++
  - .cpp
  - .cxx
  - .cc
  - .c

Môi trường điển hình của C++

Các pha của chương trình :

1. Viết code
2. Tiền xử lý
3. Dịch
4. Liên kết
5. Nạp
6. Thực thi

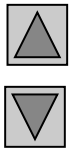


Môi trường điển hình của C++

- Hàm vào ra cơ bản
  - **`cin`**
    - Luồng vào chuẩn (nhập dữ liệu)
    - Thường là keyboard
  - **`cout`**
    - Luồng ra chuẩn (hiển thị dữ liệu)
    - Thường là màn hình
  - **`cerr`**
    - Luồng lỗi chuẩn
    - Hiển thị thông báo lỗi

chương trình: hiển thị 1 dòng chữ

- Trước khi viết chương trình
  - Comments (chú giải)
    - Tài liệu cho chương trình
    - Cải tiến việc đọc và hiểu chương trình
    - Trình biên dịch sẽ bỏ qua comments
    - Cú pháp
      - Để trong /* .. */ hoặc bắt đầu với **//**
  - Chỉ thị tiền xử lý
    - Thực hiện bởi bộ tiền xử lý trước khi dịch
    - Bắt đầu bằng **#**



Outline

fig01_02.cpp

fig01_02.cpp
output (1 of 1)

```
1  // Fig. 1.2: fig01_02.cpp
2  // A first progra
3  #include <iostrea
4
5  // function main b
6  int main()
7  {
8     std::cout << "
9
10    return 0;   //
11
12 } // end function main
```

```
Welcome to C++!
```

Single-line comments.

Function **main** returns an

Left brace **{** begins function body.

r directive to

ars exactly once in every C++ program..

Statements end with a semicolon **;**.

Corresponding right brace **}** ends function body

Name c

Stream insertion operator.

namespace **std**.

Keyword **return** is one of several means to exit function; value **0** indicates program terminated successfully.

chương trình: hiển thị 1 dòng chữ

- Đối tượng luồng (stream) ra chuẩn
  - `std::cout`
  - "Kết nối" với màn hình
  - `<<`
    - Stream insertion operator (phép chèn)
    - Giá trị để bên phải sẽ được chèn vào stream ra
- Namespace
  - `std::` xác định sẽ sử dụng tên đi phía sau "namespace" `std`
  - `std::` có thể bỏ nếu đã dùng câu lệnh `using`
- Ký tự đặc biệt
  - \
  - Dùng cho ký tự output đặc biệt

chương trình: hiển thị 1 dòng chữ

Ký tự	Mô tả
`\n`	Dòng mới
`\t`	Thêm tab
`\r`	Chuyển tới đầu dòng
`\a`	Cảnh báo. Phát ra âm thanh system bell.
`\\`	Hiển thị ký tự gạch chéo
`\"`	Hiển thị nháy kép. Dùng khi muốn hiển thị dấu nháy kép.

Chương trình: tổng 2 số

- Biến
  - Vị trí trong ô nhớ mà lưu giá trị
  - Các kiểu cơ
    - `int` - integer numbers
    - `char` - characters
    - `double` - floating point numbers
  - Khai báo biến và kiểu trước khi dùng

```
int integer1;
int integer2;
int sum;
```

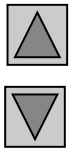
  - Có thể khai báo nhiều biến trong cùng 1 câu lệnh
    - Ngăn cách bởi dấu phẩy

```
int integer1, integer2, sum;
```

Chương trình: tổng 2 số

- Đối tượng luồng vào
  - **>>** (stream extraction operator – phép xuất)
    - Sử dụng cùng **`std::cin`**
    - Đợi người dùng nhập giá trị input, và sau đó ấn Enter
    - Lưu giá trị vào biến ở bên phải
      - Chuyển giá trị thành kiểu dữ liệu của biến
- **=** (phép gán)
  - Gán giá trị cho biến
  - Binary operator (phép toán nhị phân, 2 toán hạng)
  - Ví dụ:

```
sum = variable1 + variable2;
```



Outline

fig01_06.cpp
(1 of 1)

```
1  // Fig. 1.6: fig01_06.cpp
2  // Addition program.
3  #include <iostream>
4
5  // function main begins program execution
6  int main()
7  {
8     int integer1;  // first number to be input by user
9     int integer2;  // second nu
10    int sum;       // variable
11
12    std::cout << "Enter first i
13    std::cin >> integer1;                    // read an integer
14
15    std::cout << "Enter second integer\n"; // prompt
16    std::cin >> integer2;
17
18    sum = integer1 + integer2;
19
20    std::cout << "Sum is " << sum << std::endl; // print sum
21
22    return 0;   // indicate that program ended successfully
23
24 } // end function main
```

Declare integer variables.

Use stream extraction operator with standard input stream to obtain user input.

Calculations can be performed in output statements: alternative for lines 18 and 20:

```
std::cout << "Sum is " << integer1 + integer2 << std::endl;
```

Concatenating, chaining or cascading stream insertion operations.

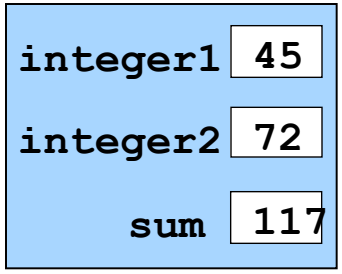
Khái niệm về bộ nhớ

- Tên biến
  - Tương ứng với vị trí thực trong bộ nhớ máy tính
  - Biến gồm tên, kiểu, kích thước và giá trị
  - Khi giá trị mới được gán vào biến, giá trị cũ sẽ bị ghi đè (mất đi)

  - std::cin >> integer1;
  - Giả sử người dùng nhập 45

  - std::cin >> integer2;
  - Giả sử người dùng nhập 72

  - sum = integer1 + integer2;

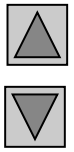


Arithmetic – Phép toán số học

- Phép toán
  - `*` : Nhân
  - `/` : Chia
    - Chia số nguyên sẽ được làm tròn xuống
      - `7 / 5` bằng 1
  - `%` : chia lấy dư
      - `7 % 5` bằng 2

Phép so sánh

Standard algebraic equality operator or relational operator	C++ equality or relational operator	Example of C++ condition	Meaning of C++ condition
Relational operators			
>	`>`	`x > y`	`x` is greater than `y`
<	`<`	`x < y`	`x` is less than `y`
≥	`>=`	`x >= y`	`x` is greater than or equal to `y`
≤	`<=`	`x <= y`	`x` is less than or equal to `y`
Equality operators			
=	`==`	`x == y`	`x` is equal to `y`
≠	`!=`	`x != y`	`x` is not equal to `y`



Outline

fig01_14.cpp
(1 of 2)

```
1  // Fig. 1.14: fig01_14.cpp
2  // Using if statements, relational
3  // operators, and equality operators.
4  #include <iostream>
5
6  using std::cout;  // program uses cout
7  using std::cin;   // program uses cin
8  using std::endl;  // program uses endl
9
10 // function main begins progra
11 int main()
12 {
13    int num1;  // first n
14    int num2;  // second
15
16    cout << "Enter two integers
17         << "the relationships
18    cin >> num1 >> num2;    // r
19
20    if ( num1 == num2 )
21       cout << num1 << " is equ
22
23    if ( num1 != num2 )
24       cout << num1 << " is not equal to " << num2 << endl;
25
```

using statements eliminate need for **std::** prefix.

Declare variables.

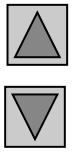
Can write **cout** and **cin** without **std::** prefix.

if structure compares values of **num1** and

If condition is true (i.e., , execute this

if structure compares values of **num1** and inequality.

If condition is true (i.e., values are not equal), execute this statement.



Outline

```
26     if ( num1 < num2 )
27        cout << num1 << " is less than " << num2 << endl;
28
29     if ( num1 > num2 )
30        cout << num1 << " is greater than " << num2 << endl;
31
32     if ( num1 <= num2 )
33        cout << num1 << " is less than or equal to "
34             << num2 << endl;
35
36     if ( num1 >= num2 )
37        cout << num1 << " is greater than or equal to "
38             << num2 << endl;
39
40     return 0;   // indicate that program ended successfully
41
42  } // end function main
```

```
Enter two integers, and I will tell you
the relationships they satisfy: 22 12
22 is not equal to 12
22 is greater than 12
22 is greater than or equal to 12
```

Statements may be split over several lines.

fig01_14.cpp
fig01_14.cpp
output (1 of 2)

Cấu trúc điều khiển

- 3 cấu trúc điều khiển để xây dựng bất kỳ chương trình nào
  - Cấu trúc tuần tự
    - Chương trình mặc định là thực hiện tuần tự
  - Cấu trúc chọn
    - `if`, `if/else`, `switch`
  - Cấu trúc lặp
    - `while`, `do/while`, `for`

Từ khoá

- C++ keywords
  - Không được dùng để đặt tên biến, class…

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				

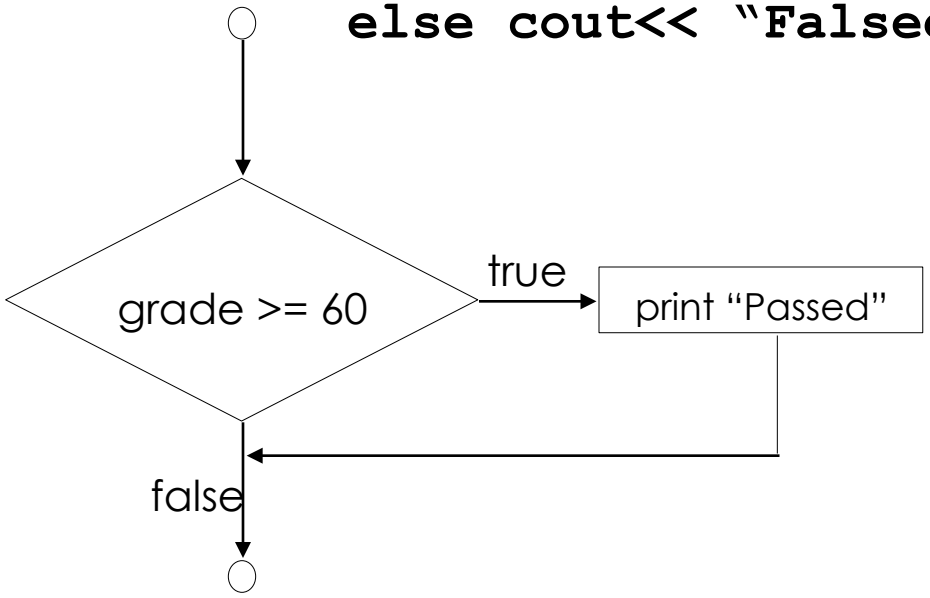
Cấu trúc if

- Ví dụ

If điểm sinh viên lớn hơn hoặc bằng 60

Print "Passed"

```
if ( grade >= 60 ){
    cout << "Passed";}
 else cout<< "Falsed";
```



Có thể dùng cả biểu thức số.

zero - **false**

nonzero - **true**

Ví dụ:

3 - 4 là **true**

Cấu trúc `if/else`

- **`if`**
  - Thực hiện hành động nếu điều kiện if là true
- **`if/else`**
  - Hành động khác nhau tương ứng với điều kiện if là true hay false
- Pseudocode

  if điểm sinh viên lớn hơn hoặc bằng 60
  print "Passed"
  else
  print "Failed"

- C++ code

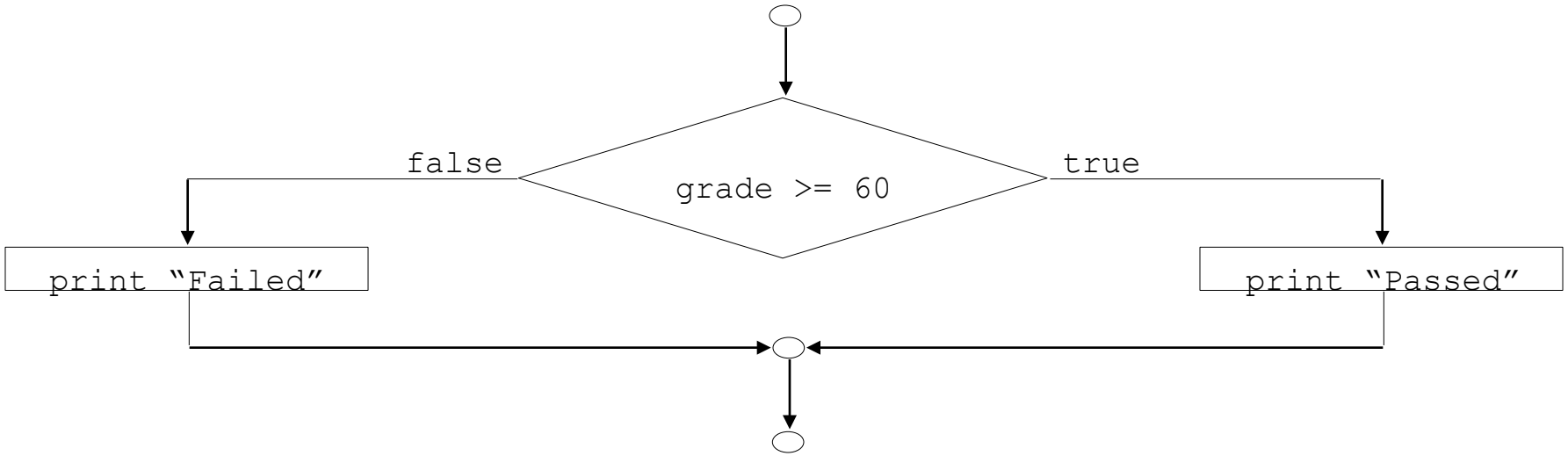
```
if ( grade >= 60 )
   cout << "Passed";
else
   cout << "Failed";
```

Cấu trúc `if/else`

- Ternary conditional operator (`?:`)
  - 3 tham số (điều kiện, giá trị nếu **`true`**, giá trị nếu **`false`**)
- Ví dụ:

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

Điều kiện **Giá trị nếu true** **Giá trị nếu false**



Cấu trúc `if/else` lồng nhau

- Cấu trúc **`if/else`** lồng nhau
  - Dùng khi có nhiều trường hợp/điều kiện

```
If điểm sinh viên lớn hơn hoặc bằng 90
      Print "A"
else
      if  nếu điểm sinh viên lớn hơn hoặc bằng 80
       Print "B"
     else
        if điểm sinh viên lớn hơn hoặc bằng 70
         Print "C"
       else
         if điểm sinh viên lớn hơn hoặc bằng 60
           Print "D"
          else
                  Print "F"
```

Cấu trúc if/else

- Ví dụ

```
if ( grade >= 90 )          // 90 hoặc hơn
   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                        // nhỏ hơn 60
   cout << "F";
```

Cấu trúc `if/else`

- Nhóm câu lệnh
  - Nhóm câu lệnh vào dấu {}

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

  - Nếu không có {},

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

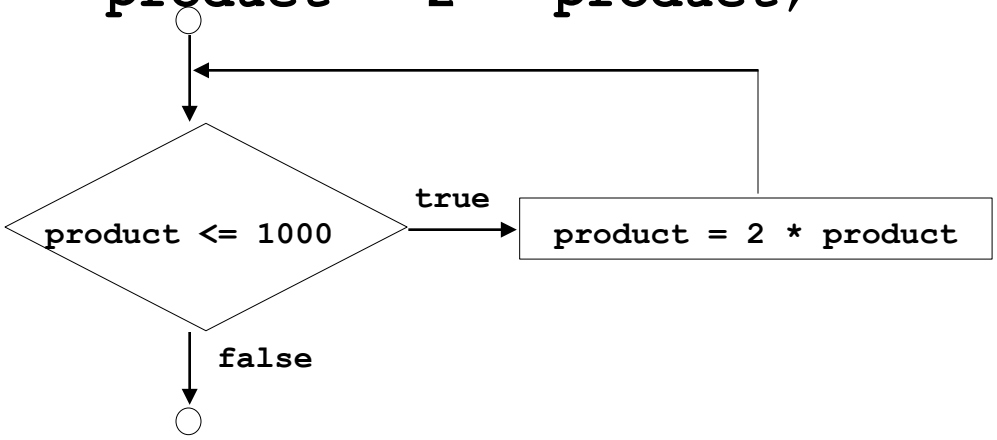
  Nằm ngoài câu lệnh if và luôn được thực thi

- Block
  - Tập các câu lệnh trong dấu {}

Cấu trúc lặp while

- Cấu trúc lặp
  - Hành động được thực hiện lặp lại khi điều kiện là true
  - **while** loop lặp cho tới khi điều kiện trở thành false
- Ví dụ

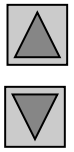
```
int product = 2;
while ( product <= 1000 )
   product = 2 * product;
```



Bộ đếm lặp

- Counter-controlled repetition
  - Loop lặp cho tới khi counter đạt giá trị cho trước
- Định nghĩa số lần lặp
  - Số lần lặp biết trước
- Ví dụ

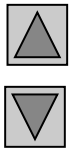
  Lớp có 10 sinh viên làm bài kiểm tra. Điểm là số nguyên từ 0-100. Tính điểm trung bình của lớp.



Outline

fig02_07.cpp
(1 of 2)

```
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
```



Outline

fig02_07.cpp
(2 of 2)

fig02_07.cpp
output (1 of 1)

```
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;                 // integer division
31
32    // display result
33    cout << "Class average is
34
35    return 0;   // indicate p
36
37 } // end function main
```

The counter gets incremented each time the loop executes. Eventually, the counter causes the loop to end.

```
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
```

Sentinel-Controlled Repetition (lính canh)

- Giả sử bài toán trở thành:

  Phát triển 1 chương trình Develop a class-averaging program that will process an arbitrary number of grades each time the program is run

  – Không biết số sinh viên

  – Làm sao chương trình biết khi nào thì dừng?

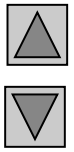
- Sentinel value (giá trị dừng)

  – Xác định “end of data entry” (kết thúc nhập dữ liệu)

  – Loop kết thúc khi nhập giá trị dừng

  – Chọn giá trị dừng sao cho không bị nhầm với giá trị hợp lệ

    - -1 trong trường hợp này

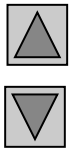


Outline

fig02_09.cpp
(1 of 3)

```
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 grades
18    int gradeCounter;   // number of grades entered
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
```

Data type **double** used to represent decimal numbers.



Outline

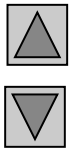
fig02_09.cpp
(2 of 3)

```
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 + g
35       gradeCounter = gr
36
37       cout << "Enter gr
38       cin >> 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
```

`static_cast<double>()` treats **`total`** as a **`double`** temporarily (casting).

Required because dividing two integers truncates the remainder.

`gradeCounter` is an **`int`**, but it gets *promoted* to **`double`**.



Outline

```
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
```

```
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
```

fig02_09.cpp
(3 of 3)

fig02_09.cpp
output (1 of 1)

`setprecision(2)` prints two digits past decimal point (rounded to fit precision).

Programs that use this must include **`<iomanip>`**

`fixed` forces output to print in fixed point format (not scientific notation). Also, forces trailing zeros and decimal point to print.

Include **`<iostream>`**

cấu trúc chọn switch

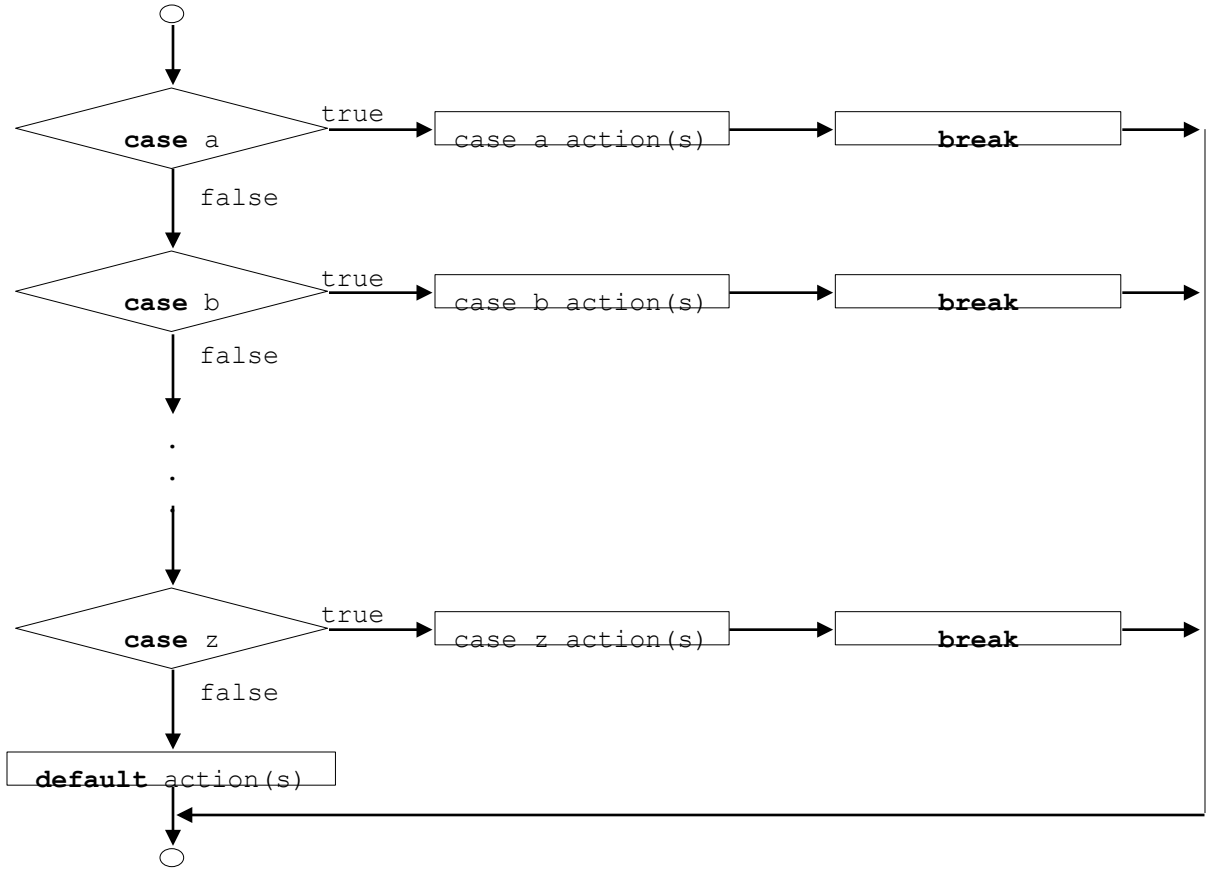
- **switch**
  - Kiểm tra biến với nhiều giá trị

```
switch ( variable ) {
    case value1:          // kiểm tra nếu variable == value1
    statements
    break;          // để thoát switch

case value2:
case value3:          // kiểm tra nếu variable == value2 hoặc == value3
    statements
    break;

default:              // thực hiện nếu không case nào thoả mãn
    statements
        break;
}
```

cấu trúc chọn switch



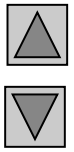
cấu trúc chọn switch

- Ví dụ
  - Chương trình đọc điểm (A-F)
  - Hiển thị số tương ứng của điểm nhập vào
- Về characters (kí tự)
  - Kí tự đơn thường được lưu trong kiểu dữ liệu **char**
    - **char** là integer 1 byte, vì thế nhiều **char**s có thể được lưu bằng nhiều **int**s
  - Có thể coi kí tự là **int** hoặc **char**
    - 97 là biểu diễn số của 'a' (ASCII)
    - Sử dụng nháy đơn để biểu diễn char

```
cout << "Ki tu (" << 'a' << ") co gia tri la "
       << static_cast< int > ( 'a' ) << endl;
```

sẽ in ra màn hình:

```
Ki tu (a) co gia tri la 97
```



Outline

fig02_22.cpp
(1 of 4)

```
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
```

Lập trình cấu trúc (Structured programming)

- Tất cả các chương trình được chia nhỏ thành
  - Tuần tự
  - Chọn
    - **`if`**, **`if/else`**, hoặc **`switch`**
    - Bất kỳ cấu trúc chọn nào cũng có thể viết bằng câu lệnh **`if`**
  - Lặp
    - **`while`**, **`do/while`** hoặc **`for`**
    - Bất kỳ cấu trúc lặp nào cũng có thể viết lại bằng câu lệnh **`while`**

Outline

22.cpp

```
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 structu
27
28          case 'A':        // grade was uppercase A
29          case 'a':        // or lowercase a
30             ++aCount;     // increment aCount
31             break;        //
32
33          case 'B':        //
34          case 'b':        //
35             ++bCount;     //
36                           //
37
38                           //
39                           //
40             ++cCount;     //
41             break;        //
42
```

`break` causes **`switch`** to end and the program continues with the first statement after the **`switch`** structure.

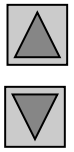
`cin.get()` uses dot notation (explained chapter 6). This function gets 1 character from the keyboard (after *Enter* pressed), and it is assigned to **`grade`**.

`cin.get()` returns EOF (end-of-file) after the EOF character is input, to indicate the end of data. EOF may be ctrl-d or ctrl-z, depending on your OS.

Assignment statements have a value, which is the same as the variable on the left of the **`=`**. The value of this statement is the same as the value returned by **`cin.get()`**.

This can also be used to initialize multiple variables:
`a = b = c = 0;`

Compares **`grade`** (an **`int`**) to the numerical representations of **`A`** and **`a`**.



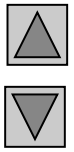
Outline

fig02_22.cpp
(3 of 4)

```
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
49          case 'f':          // or lowerc
50             ++fCount;       // increment
51             break;          // exit swit
52
53          case '\n':         // ignore ne
54          case '\t':         // tabs,
55          case ' ':          // and spac
56             break;          // exit swi
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
```

This test is necessary because *Enter* is pressed after each letter grade is input. This adds a newline character that must be removed. Likewise, we want to ignore any whitespace.

Notice the **`default`** statement, which catches all other cases.



Outline

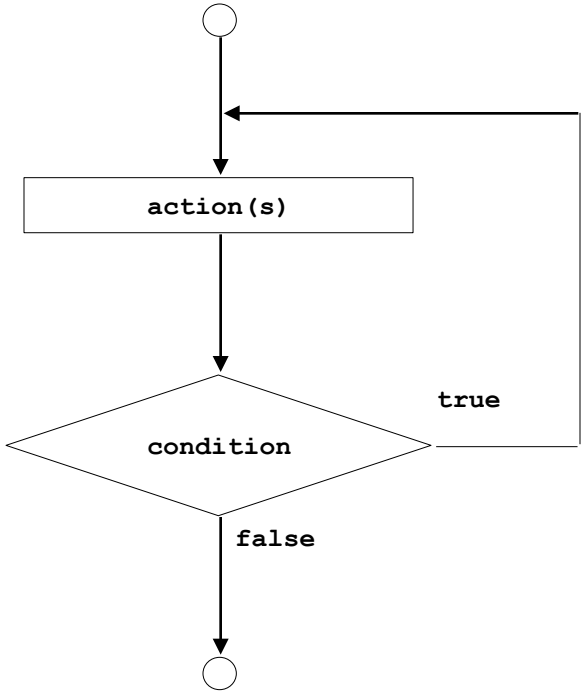
fig02_22.cpp
(4 of 4)

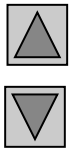
```
67    // output summary of results
68    cout << "\n\nTotals for each letter grade are:"
69         << "\nA: " << aCount   // display number of A grades
70         << "\nB: " << bCount   // display number of B grades
71         << "\nC: " << cCount   // display number of C grades
72         << "\nD: " << 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
```

Cấu trúc lặp do/while

- Tương tự **while**
  - Thực hiện vòng lặp rồi mới kiểm tra điều kiện ở cuối (while kiểm tra điều kiện ở đầu)
  - Thân vòng lặp thực hiện ít nhất 1 lần
- Cấu trúc

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





Outline

fig02_24.cpp
(1 of 1)

fig02_24.cpp
output (1 of 1)

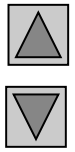
```
1  // Fig. 2.24: fig02_24.cpp
2  // Using the do/while repetition structure.
3  #include <iostream>
4
5  using std::cout;
6  using std::endl;
7
8  // function main begins program execution
9  int main()
10 {
11    int counter = 1;
12
13    do {
14       cout << counter << "  ";    // display counter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

Notice the preincrement in loop-continuation test.

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

Câu lệnh break và continue

- **`break`**
  - thoát luôn khỏi **`while`**, **`for`**, **`do/while`**, **`switch`**
  - chương trình tiếp tục với câu lệnh đầu tiên sau khối lệnh chứa break
- Thường dùng để
  - Thoát sớm khỏi vòng lặp
  - Bỏ qua phần còn lại của **`switch`**



Outline

fig02_26.cpp
(1 of 2)

```
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;
```

Exits **for** structure when **break** executed.

Phép toán logic

- Sử dụng trong điều kiện của câu lệnh lặp và if
- **&&** (phép **AND**)
  - **true** nếu cả 2 điều kiện là **true**

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

- || (phép **OR**)
  - **true** nếu một trong 2 điều kiện là **true**

```
if ( semesterAverage >= 90 || finalExam >= 90 )
   cout << "Diem sinh vien la A" << endl;
```

Phép toán logic

- **!** (phép **NOT**, phép phủ định)
  - Trả về **true** nếu điều kiện là **false**, & ngược lại

```
if ( !( grade == sentinelValue ) )
   cout << "Diem tiep theo la " << grade << endl;
```

Thay bằng:

```
if ( grade != sentinelValue )
   cout << "Diem tiep theo la " << grade << endl;
```

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

- Lỗi thông dụng
  - Không gây ra lỗi cú pháp
- Ảnh hưởng tới chương trình
  - Giá trị của các biểu thức sẽ được dùng làm giá trị của điều kiện
    - Zero = false, khác zero = true
  - Phép gán sẽ thay đổi giá trị cho biến được gán

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

- Ví dụ

```
if ( payCode == 4 )
    cout << "Ban duoc thuong!" << endl;
```

  - Nếu paycode là 4, bạn sẽ được thưởng

- Nếu == được thay bởi =

```
if ( payCode = 4 )
   cout << "Ban duoc thuong!" << endl;
```

  - Paycode được gán bằng 4 (bất kể trước đó nó bằng gì)
  - Câu lệnh là true (vì 4 khác 0)
  - Phần thưởng sẽ được phát cho bất kì trường hợp nào

Lập trình cấu trúc (Structured programming)

- Structured programming
  - Chương trình dễ hiểu, dễ kiểm thử, debug và chỉnh sửa
- Quy tắc cho structured programming
  - Chỉ dùng các cấu trúc điều khiển có 1 đầu vào/1 đầu ra
  - Quy tắc
    1) Bắt đầu bằng flowchart đơn giản nhất
    2) Bất kì hình chữ nhật nào (action) cũng có thể thay bằng 2 hình chữ nhật tuần tự
    3) Bất kì hình chữ nhật nào (action) cũng có thể thay bằng bất kì cấu trúc điều khiển nào (sequence, if, if/else, switch, while, do/while hoặc for)
    4) Quy tắc 2 và 3 có thể được áp dụng theo thứ tự bất kỳ và áp dụng nhiều lần

Lập trình cấu trúc (Structured programming)

- Structured programming
  - Chương trình dễ hiểu, dễ kiểm thử, debug và chỉnh sửa
- Quy tắc cho structured programming
  - Chỉ dùng các cấu trúc điều khiển có 1 đầu vào/1 đầu ra
  - Quy tắc
    1) Bắt đầu bằng flowchart đơn giản nhất
    2) Bất kì hình chữ nhật nào (action) cũng có thể thay bằng 2 hình chữ nhật tuần tự
    3) Bất kì hình chữ nhật nào (action) cũng có thể thay bằng bất kì cấu trúc điều khiển nào (sequence, if, if/else, switch, while, do/while hoặc for)
    4) Quy tắc 2 và 3 có thể được áp dụng theo thứ tự bất kỳ và áp dụng nhiều lần

Lập trình cấu trúc (Structured programming)

Ví dụ về áp dụng quy tắc 3 (thay thế hình chữ nhật bằng 1 cấu trúc điều khiển)