Kỹ Thuật Lập Trình

Mục tiêu môn học?

- Học phần *Kỹ thuật lập trình* trang bị cho sinh viên những kỹ thuật cơ bản nhất mà một lập trình viên chuyên nghiệp cần phải nắm vững để viết mã nguồn hiệu quả. Các kiến thức giảng dạy góp phần quan trọng giúp sinh viên phát triển được các ứng dụng phần mềm chất lượng cao trong thực tế.
- Học phần này trang bị cho sinh viên các kỹ thuật lập trình quan trọng như quản lý bộ nhớ, hàm, kỹ thuật đệ quy, kỹ thuật sử dụng các cấu trúc dữ liệu để giải quyết vấn đề, kỹ thuật viết mã nguồn hiệu quả, kỹ thuật lập trình phòng ngừa, kỹ thuật gỡ rối, tinh chỉnh mã nguồn, phong cách lập trình. Học phần có các buổi thực hành nhằm rèn luyện và nâng cao kỹ năng lập trình của sinh viên.

Tài liệu học tập

1. Bài giảng trên lớp
2. Giáo trình Kỹ thuật lập trình khoa CNTT-ĐH Công nghiệp Hà Nội
3. Bài tập thực hành môn Kỹ thuật lập trình-ĐH Công nghiệp Hà Nội
4. Lập trình C++ cơ bản và nâng cao; Phạm văn Ất .
5. Efficient c programming. Mark Allen Weiss. Prentice Hall, 1998.

Tổng quan về lập trình

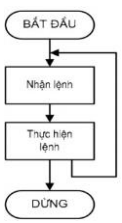
Hoạt động của chương trình máy tính và ngôn ngữ lập trình

Chương trình máy tính và ngôn ngữ lập trình

- Chương trình máy tính: Tập hợp các lệnh chỉ dẫn cho máy tính thực hiện nhiệm vụ
- Ngôn ngữ lập trình: Dùng để viết các lệnh, chỉ thị

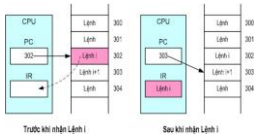
Hoạt động của chương trình máy tính

- Chương trình máy tính được nạp vào bộ nhớ chính (primary memory) như là một tập các lệnh viết bằng ngôn ngữ mà máy tính hiểu được, tức là một dãy tuần tự các số nhị phân (binary digits).
- Tại bất cứ một thời điểm nào, máy tính sẽ ở một trạng thái (state) nào đó. Đặc điểm cơ bản của trạng thái là con trỏ lệnh (instruction pointer) trỏ tới lệnh tiếp theo để thực hiện.
- Thứ tự thực hiện các nhóm lệnh được gọi là luồng điều khiển (flow of control).



Hoạt động của chương trình máy tính

- Bắt đầu mỗi chu trình lệnh, CPU nhận lệnh từ bộ nhớ chính.
  - PC (Program Counter): thanh ghi giữ địa chỉ của lệnh sẽ được nhận
  - Lệnh được nạp vào thanh ghi lệnh IR (Instruction Register)
- Sau khi lệnh được nhận vào, nội dung PC tự động tăng để trỏ sang lệnh kế tiếp



7

Ngôn ngữ lập trình

- Ngôn ngữ lập trình là một hệ thống các ký hiệu dùng để liên lạc, trao đổi với máy tính nhằm thực thi một nhiệm vụ tính toán.
- Có rất nhiều ngôn ngữ lập trình (khoảng hơn 1000), phần lớn là các ngôn ngữ hàn lâm, có mục đích riêng hay phạm vi.

8

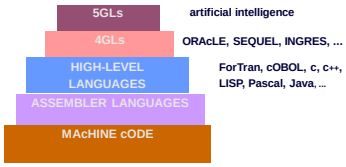
Ngôn ngữ lập trình

Có 3 thành phần căn bản của bất cứ 1 NNLT nào:

- *Mô thức lập trình* là những nguyên tắc chung cơ bản, dùng bởi LTV để xây dựng chương trình.
- *Cú pháp* của ngôn ngữ là cách để xác định cái gì là hợp lệ trong cấu trúc các câu của ngôn ngữ; Nắm được cú pháp là cách để đọc và tạo ra các câu trong các ngôn ngữ tự nhiên, như tiếng Việt, tiếng Anh. Tuy nhiên điều đó không có nghĩa là nó giúp chúng ta hiểu hết ý nghĩa của câu văn.
- *Ngữ nghĩa* của 1 program trong ngôn ngữ ấy. Rõ ràng, nếu không có semantics, 1 NNLT sẽ chỉ là 1 mớ các câu văn vô nghĩa; như vậy semantics là 1 thành phần không thể thiếu của 1 ngôn ngữ.

9

Các lớp ngôn ngữ lập trình



10

Mã máy – Machine code

Máy tính chỉ nhận các tín hiệu điện tử - có, không có - tương ứng với các dòng bits.

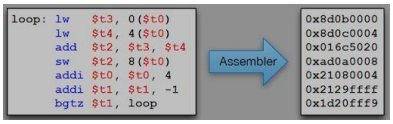
Một chương trình ở dạng đó gọi là mã máy (machine code).

11

Hợp ngữ - Assembly

Là bước đầu tiên của việc xây dựng cơ chế viết chương trình tiện lợi hơn thông qua các ký hiệu, từ khóa và cả mã máy.

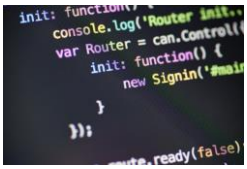
Tất nhiên, để chạy được các chương trình này thì phải chuyển thành machine code.



12

Ngôn ngữ lập trình bậc cao

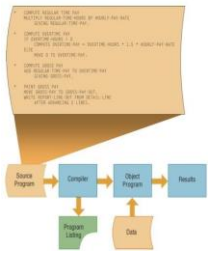
- Thay vì dựa trên phần cứng (machine-oriented) cần tìm cơ chế dựa trên vấn đề (problem-oriented) để tạo chương trình
- Gần gũi với ngôn ngữ tự nhiên hơn, thường sử dụng các từ khóa giống tiếng Anh



13

Trình dịch - compiler

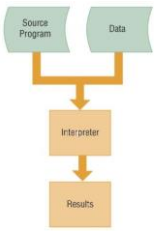
- Chương trình thực hiện biên dịch toàn bộ chương trình nguồn thành mã máy trước khi thực hiện



14

Thông dịch - interpreter

- Chương trình dịch và thực hiện từng dòng lệnh của chương trình cùng lúc
- Dịch từ ngôn ngữ này sang ngôn ngữ khác, không tạo ra chương trình dạng mã máy hay assembly



15

Các mô thức lập trình

Programming paradigm

Ngôn ngữ lập trình bậc cao

- Imperative paradigm
- Functional paradigm
- Logical paradigm
- Object-oriented paradigm
- Visual paradigm
- Parallel paradigm
- Concurrent paradigm
- Distributed paradigm
- Service-oriented paradigm

17

Imperative paradigm – hướng mệnh lệnh

first do this and next do that

Thành phần:

- **Declarative statements**, các lệnh khai báo: cung cấp các tên cho biến. Các biến này có thể thay đổi giá trị trong quá trình thực hiện Chương trình.
- **Assigment statements**, lệnh gán: gán giá trị mới cho biến
- **Program flow control statements**, các lệnh điều khiển cấu trúc chương trình: Xác định trình tự thực hiện các lệnh trong chương trình.
- **Module**: chia chương trình thành các chương trình con: Functions & Procedures

18

Functional paradigm – hướng chức năng

Thành phần
- Tập hợp các cấu trúc dữ liệu và các hàm liên quan
- Tập hợp các hàm cơ sở
- Tập hợp các toán tử

Đặc trưng cơ bản: ***module hóa chương trình***
- Chức năng là biểu diễn của một biểu thức
- Giải thuật thực hiện theo từng bước
- Giá trị trả về là không thể biến đổi
- Không thể thay đổi CTDL của giá trị nhưng có thể sao chép các thành phần tạo nên giá trị đó
- Tính toán bằng cách gọi các chức năng

19

Logic paradigm – hướng logic

answer a *question* via searching for a *solution*

- Ý tưởng: Tự động kiểm chứng trong trí tuệ nhân tạo
- Dựa trên các tiên đề - axioms, các quy luật suy diễn - inference rules, và các truy vấn - queries
- Chương trình thực hiện từ việc tìm kiếm có hệ thống trong 1 tập các sự kiện, sử dụng 1 tập các luật để đưa ra kết luận

20

Object-oriented paradigm – hướng đối tượng

send messages between *objects* to simulate a temporal evolution of a set of *real world phenomena*

- Ý tưởng: Các khái niệm và mô hình tương tác trong thế giới thực
- Dữ liệu cũng như các thao tác trên dữ liệu được bao gói trong các đối tượng
- Cơ chế che giấu thông tin nội bộ được sử dụng để tránh những tác động từ bên ngoài

21

Object-oriented paradigm – hướng đối tượng

- Các đối tượng tương tác với nhau qua việc truyền thông điệp, đó là phép ẩn dụ cho việc thực hiện các thao tác trên 1 đối tượng
- Trong phần lớn các NNLT HĐT, đối tượng phân loại thành các lớp
  - *Đối tượng trong các lớp có chung các thuộc tính, cho phép lập trình trên lớp, thay vì lập trình trên từng đối tượng riêng lẻ*
  - *Lớp đại diện cho các khái niệm còn đối tượng đại diện cho thể hiện*
  - *Lớp có tính kế thừa, cho phép mở rộng hay chuyên biệt hóa*

22

Giới thiệu về ngôn ngữ C++

23

Các yếu tố cơ bản

1. Bảng ký tự

- Các chữ cái la tinh (viết thường và viết hoa): a .. z và A .. Z. Cùng một chữ cái nhưng viết thường phân biệt với viết hoa. Ví dụ chữ cái 'a' là khác với 'A'.
- Dấu gạch dưới: _
- Các chữ số thập phân: 0, 1, . .., 9.
- Các ký hiệu toán học: +, -, *, /, % , &, ||, !, >, <, = ...
- Các ký hiệu đặc biệt khác: , ;: [ ], {}, #, dấu cách, ...

Các yếu tố cơ bản

2. Từ khóa

Một từ khoá là một từ được qui định trước trong NNLT với một ý nghĩa cố định, thường dùng để chỉ các loại dữ liệu hoặc kết hợp thành câu lệnh

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

25

Các yếu tố cơ bản

3. Tên gọi

Để phân biệt các đối tượng với nhau chúng cần có một tên gọi, qui tắc đặt tên:

- Là dãy ký tự liên tiếp (không chứa dấu cách) và phải bắt đầu bằng chữ cái hoặc gạch dưới.
- Phân biệt kí tự in hoa và thường.
- Không được trùng với từ khóa.
- Số lượng chữ cái dùng để phân biệt tên gọi có thể được đặt tuỳ ý.
- Chú ý: các tên gọi có sẵn của C++ cũng tuân thủ theo đúng qui tắc trên.

26

Các yếu tố cơ bản

4. Ghi chú trong chương trình

Có 2 cách báo cho chương trình biết một đoạn chú thích:

- Nếu chú thích là một đoạn kí tự bất kỳ liên tiếp nhau (trong 1 dòng hoặc trên nhiều dòng) ta đặt đoạn chú thích đó giữa cặp dấu đóng mở chú thích /* (mở) và */ (đóng).
- Nếu chú thích bắt đầu từ một vị trí nào đó cho đến hết dòng, thì ta đặt dấu // ở vị trí đó. Như vậy // sử dụng cho các chú thích chỉ trên 1 dòng.

•*Chú ý: Cặp dấu chú thích /* ... */ không được phép viết lồng nhau*

27

Môi trường làm việc

- Tương tự như ngôn ngữ lập trình và các phần mềm ứng dụng khác – sinh viên đọc trong tài liệu (sẽ làm quen trong buổi thực hành đầu tiên)

28

Chương trình c++ đầu tiên

```
1. #include <iostream.h>
2.
3. int main()
4. {
5.     cout<<"Hello\n";
6.     return 0;
7. }
```

29

Chương trình C++

- #include <iostream>//#include <iostream.h>
  - khai báo sử dụng thư viện xuất/nhập chuẩn (standard I/O library). các thư viện khác: string, time, math…
- int main()
  - khai báo hàm main(). chương trình C++ phải khai báo (duy nhất) một hàm main(). Khi chạy, chương trình sẽ bắt đầu thực thi ở câu lệnh đầu tiên trong hàm main().
- { … } mở và đóng một khối mã.
- cout Lệnh cout gửi kết xuất ra thiết bị xuất chuẩn .
- return 0; ngừng chương trình. Mã lỗi 0 (error code 0) – không có lỗi khi chạy chương trình.

30

Mở rộng 1

```
1.  #include <iostream>
2.  using namespace  std;
3.  int main()
4.  {
5.     int a, b, c;
6.     a = 5;
7.     b = 7;
8.     c = a + b;
9.     cout<<a<< b<<c;
10.    return 0;
11. }
```

31

Các bước tạo và thực hiện một chương trình

1. Qui trình viết và thực hiện chương trình
2. Soan thảo chương trình nguồn
3. Dịch chương trình

Sau khi đã soạn thảo xong chương trình nguồn, bước tiếp theo thường là dịch (ấn tổ hợp phím Alt-F9) để tìm và sửa các lỗi gọi là lỗi cú pháp. Trong khi dịch C++ sẽ đặt con trỏ vào nơi gây lỗi (viết sai cú pháp) trong văn bản. *(dùng Alt-F8 để chuyển con trỏ đến lỗi tiếp, để chuyển con trỏ về ngược lại lỗi trước đó có thể dùng Alt-F7*)

4. Chạy chương trình

Ấn Ctrl-F9 để chạy chương trình

32

Vào / Ra trong C++

1. Vào dữ kiệu từ bàn phím

Để nhập dữ liệu vào cho các biến có tên **biến_1**, **biến_2**, **biến_3** chúng ta sử dụng câu lệnh:

cin >> biến_1 ;
cin >> biến_2 ;
cin >> biến_3 ;

hoặc:

cin >> biến_1 >> biến_2 >> biến_3 ;

- Khi chạy chương trình nếu gặp các câu lệnh trên chương trình sẽ "tạm dừng" để chờ NSD nhập dữ liệu vào cho các biến.
- Sau khi NSD nhập xong dữ liệu, chương trình sẽ tiếp tục chạy từ câu lệnh tiếp theo sau của các câu lệnh trên.

33

Vào / Ra trong C++

2. In dữ liệu ra màn hình

Để in giá trị của các bt_1, bt_2, bt_3 ra màn hình ta dùng câu lệnh sau:

cout << bt_1 ;
cout << bt_2 ;
cout << bt_3 ;

hoặc:

cout << bt_1 << bt_2 << bt_3 ;

Chú ý: để sử dụng các câu lệnh nhập và in trong phần này, đầu chương trình phải có dòng khai báo #include <iostream.h>.

34

Vào / Ra trong C++

- Định dạng dữ liệu trước khi xuất:

- **Cách 1: Sử dụng toán tử định dạng**
  - ***cout.width(int n)***
  - ***cout.fill(char ch)***
  - ***cout.precision(int n)***

- ***VD: cout.width(9); cout.fill('0'); cout.precision(2); cout<<a;***

Vào / Ra trong C++

- Định dạng thông tin cần đưa ra màn hình

Cách 2: Để sử dụng các định dạng này cần khai báo file nguyên mẫu ***<iomanip.h>***

- **endl**: Tương đương với kí tự xuống dòng '\n'.
- **setw(int n):** n là số cột màn hình để hiển thị đối tượng
- **setfill (char ch)**
- **setprecision(int n):** Chỉ định số chữ số của phần thập phân in ra là n. Số sẽ được làm tròn trước khi in ra.
- **setiosflags(ios::showpoint):** Phương thức setprecision chỉ có tác dụng trên một dòng in. Để cố định các giá trị đã đặt cho mọi dòng in (cho đến khi đặt lại giá trị mới) ta sử dụng phương thức setiosflags(ios::showpoint).
- **VD:** ***cout<<setw(9)<<setfill('0')<<setprecision(2)<<a;***

36

Vào / Ra trong C++

- Chú ý: Toán tử nhập ">>" chủ yếu làm việc với dữ liệu kiểu số. Để nhập kí tự hoặc xâu kí tự, C++ cung cấp các phương thức (hàm) sau:
- cin.get(c): cho phép nhập một kí tự vào biến kí tự c,
- cin.getline(s,n): cho phép nhập tối đa n-1 kí tự vào xâu s.

37

Một chương trình đơn giản

```
1.  #include <iostreamh>
2.  using namespace std;
3.  int main()
4.  {
5.     int a, b, c;
6.     cout<<"Nhap so thu nhat: ";
7.     cin>>a;
8.     cout<<"Nhap so thu hai: ";
9.     cin>>b;
10.    c = a + b;
11.    cout<<a<<"+"<<b<<"="<<c;
12.    return 0;
13. }
```

12	c
7	b
5	a

```
c:\> tong.exe
   Nhap so thu nhat:  5
   Nhap so thu hai:   7
   5 + 7 = 12
c:\>_
```

38

KIỂU DỮ LIỆU, BIỂU THỨC VÀ CÂU LỆNH

các kiểu dữ liệu cơ bản

Loại dữ liệu	Tên kiểu	Số ô nhớ	Miền giá trị
Kí tự	char	1 byte	– 128 .. 127
	unsigned char	1 byte	0 .. 255
Số nguyên	int	2 byte	– 32768 .. 32767
	unsigned int	2 byte	0 .. 65535
	short	2 byte	– 32768 .. 32767
	long	4 byte	$-2^{15} .. 2^{15} - 1$
Số thực	float	4 byte	$\pm 10^{-37} .. \pm 10^{+38}$
	double	8 byte	$\pm 10^{-307} .. \pm 10^{+308}$

39

Kiểu ký tự

- Theo bảng trên ta thấy có 2 loại kí tự là char với miền giá trị từ -128 đến 127 và unsigned char (kí tự không dấu) với miền giá trị từ 0 đến 255.
- Trường hợp một biến được gán giá trị vượt ra ngoài miền giá trị của kiểu thì giá trị của biến sẽ được tính theo mã bù – (256 - c).
- Ví dụ nếu gán cho char c giá trị 179 (vượt khỏi miền giá trị đã được qui định của char) thì giá trị thực sự được lưu trong máy sẽ là - (256 - 179) = -77

40

Kiểu ký tự

- *Ví dụ 1* :
- char c, d ; // c, d được phép gán giá trị từ -128 đến 127
- unsigned e,f ; // e được phép gán giá trị từ 0 đến 255
- c = 65 ; d = 179 ; // d có giá trị ngoài miền cho phép
- e = 179; f = 330 ; // f có giá trị ngoài miền cho phép
- cout << c << int(c) ; // in ra chữ cái 'A' và giá trị số 65
- cout << d << int(d) ; // in ra là kí tự '|' và giá trị số -77
- cout << e << int(e) ; // in ra là kí tự '|' và giá trị số 179
- cout << f << int(f) ; // in ra là kí tự 'J' và giá trị số 74

- *Chú ý: Qua ví dụ trên ta thấy một biến nếu được gán giá trị ngoài miền cho phép sẽ dẫn đến kết quả không theo suy nghĩ thông thường. Do vậy nên tuân thủ qui tắc chỉ gán giá trị cho biến thuộc miền giá trị mà kiểu của biến đó qui định*

41

Kiểu số nguyên

- Các số nguyên được phân chia thành 4 loại kiểu khác nhau với các miền giá trị tương ứng được cho trong bảng 1.
- Đó là kiểu số nguyên ngắn (**short**) tương đương với kiểu số nguyên (**int**) sử dụng 2 byte
- Số nguyên dài (**long int**) sử dụng 4 byte.
- Kiểu số nguyên thường được chia làm 2 loại có dấu (**int**) và không dấu (**unsigned int** hoặc có thể viết gọn hơn là **unsigned**)

42

Kiểu số thực

Để sử dụng số thực ta cần khai báo kiểu float hoặc double mà miền giá trị của chúng được cho trong bảng 1

Ví dụ 2 : Chương trình sau đây sẽ in diện tích và chu vi của một hình tròn có bán kính 2cm với 3 số lẻ.

```
#include <iostream>
#include <iomanip.h>
using namespace std;
int main()
{
   float r = 2 ; // r là tên biến dùng để chứa bán kính
   cout << "Diện tích = " << setiosflags(ios::showpoint) ;
   cout << setprecision(3) << r * r * 3.1416 ;
   return 0;
}
```

43

HẰNG - KHAI BÁO VÀ SỬ DỤNG HẰNG

Hằng là một giá trị cố định nào đó ví dụ 3 (hằng nguyên), 'A' (hằng kí tự), 5.0 (hằng thực), "Ha noi" (hằng xâu kí tự). Một giá trị có thể được hiểu dưới nhiều kiểu khác nhau, do vậy khi viết hằng ta cũng cần có dạng viết thích hợp.

- **Hằng nguyên**

- kiểu short, int: 3, -7, ...

- kiểu unsigned: 3, 123456, ...

- kiểu long, long int: 3L, -7L, 123456L, ... (viết L vào cuối mỗi giá trị)

44

HẰNG - KHAI BÁO VÀ SỬ DỤNG HẰNG

- **Hằng thực**

Một số thực có thể được khai báo dưới dạng kiểu float hoặc double và các giá trị của nó có thể được viết dưới một trong hai dạng.

- *Dạng dấu phảy tĩnh*

Theo cách viết thông thường. Ví dụ: 3.0, -7.0, 3.1416, ...

- *Dạng dấu phảy động*

Tổng quát, một số thực x có thể được viết dưới dạng: men hoặc mEn, trong đó m được gọi là phần định trị, n gọi là phần bậc (hay mũ). Số men biểu thị giá trị x = m x 10n

45

HẰNG - KHAI BÁO VÀ SỬ DỤNG HẰNG

- **Hằng kí tự**

- Cách viết hằng

Có 2 cách để viết một hằng kí tự. Đối với các kí tự có mặt chữ thể hiện ta thường sử dụng cách viết thông dụng đó là đặt mặt chữ đó giữa 2 dấu nháy đơn như: 'A', '3',

Với một số kí tự không có mặt chữ ta buộc phải dùng giá trị (số) của chúng, như viết 27 thay cho kí tự được nhấn bởi phím Escape

46

HẰNG - KHAI BÁO VÀ SỬ DỤNG HẰNG

- **Hằng kí tự**

- Một số hằng thông dụng

'\n' (endl)	:	biểu thị kí tự xuống dòng (cũng tương đương với
'\t'	:	kí tự tab
'\a' (một tiếng 'bíp')	:	kí tự chuông (tức thay vì in kí tự, loa sẽ phát ra
'\r'	:	xuống dòng
'\f'	:	kéo trang
'\\'	:	dấu \
'\?'	:	dấu chấm hỏi ?
'\''	:	dấu nháy đơn '
'\"'	:	dấu nháy kép "

47

HẰNG - KHAI BÁO VÀ SỬ DỤNG HẰNG

- **Hằng xâu kí tự**

Là dãy kí tự bất kỳ đặt giữa cặp dấu nháy kép.

Ví dụ: "Lớp K43*", "12A4", "A", "<>", "" là các hằng xâu kí tự, trong đó "" là xâu không chứa kí tự nào, các xâu "<>", "A" chứa 1 kí tự ...

Số các kí tự giữa 2 dấu nháy kép được gọi là độ dài của xâu.

Ví dụ xâu "" có độ dài 0, xâu "<>" hoặc "A" có độ dài 1 còn xâu "Lớp K43*" có độ dài 8.

- ***Chú ý***: phân biệt giữa 2 cách viết 'A' và "A",

- Tương tự ta không được viết '' (2 dấu nháy đơn liền nhau) vì không có khái niệm kí tự "rỗng". Để chỉ xâu rỗng (không có kí tự nào) ta phải viết "" (2 dấu nháy kép liền nhau)

48

HẰNG - KHAI BÁO VÀ SỬ DỤNG HẰNG

- Để khai báo hằng ta dùng các câu khai báo sau:

#define <tên_hằng> <giá_trị_hằng> ;

hoặc:

const <tên_hằng> = <giá_trị_hằng> ;

Ví dụ:

```
#define SOSV 50 ;
#define MAX 100 ;
const  PI = 3.15 ;
```

49

Khai báo và sử dụng biến

- Biến số (variable) được dùng để giữ các giá trị và có thể thay đổi các giá trị mà biến đang giữ

Khai báo: ***<typename> varname***;

Vd:

```
int i;
float x, y, z;
char c;
```

Gán giá trị cho biến: ***<varname> = <value>;***

vd:

```
i = 4;
x = 5.4;
y = z = 1.2;
```

50

Khai báo và sử dụng biến

- Khai báo nhiều biến cùng kiểu:

  <typename> varname1,varname2,….;

- Vừa khai báo vừa khởi tạo:

  <typename> varname1=var1, varname2=var2, . …;

51

PHÉP TOÁN, BIỂU THỨC VÀ CÂU LỆNH

- **Phép toán**

*- Các phép toán số học: +, -, *, /, %*

Các phép toán + (cộng), - (trừ), * (nhân) được hiểu theo nghĩa thông thường trong số học.

Phép toán a / b (chia) được thực hiện theo kiểu của các toán hạng, tức nếu cả hai toán hạng là số nguyên thì kết quả chỉ lấy phần nguyên, nếu 1 trong 2 toán hạng là thực thì kết quả là số thực.

Ví dụ:

```
13/5 = 2                          // do 13 và 5 là 2 số nguyên
13.0/5 = 13/5.0 = 13.0/5.0 = 2.6 // do có ít nhất 1 toán hạng là thực
```

Phép toán a % b (lấy phần dư) trả lại phần dư của phép chia a/b, trong đó a và b là 2 số nguyên. Ví dụ:

```
13%5 = 3      // phần dư của 13/5
5%13 = 5
```

52

PHÉP TOÁN, BIỂU THỨC VÀ CÂU LỆNH

- ***Các phép toán tự tăng, giảm: i++, ++i, i--, --i***

Phép toán ++i và i++ sẽ cùng tăng i lên 1 đơn vị tức tương đương với câu lệnh i = i+1

Phép i--, --i cũng có ý nghĩa tương tự

- Giả sử i = 3, j = 15

(các phép toán dưới đây là độc lập)

Phép toán	Tương đương	Kết quả
i = ++j	j = j + 1 ; i = j ;	i = 16 , j = 16
i = j++	i = j ; j = j + 1	i = 15 , j = 16
j = ++i + 5 ;	i = i + 1 ; j = i + 5	i = 4, j = 9
j = i++ + 5	j = i + 5; i = i + 1	i = 4, j = 8

53

PHÉP TOÁN, BIỂU THỨC VÀ CÂU LỆNH

- **Các phép toán so sánh và logic**

Các phép toán so sánh

== (bằng nhau), **!=** (khác nhau), **>** (lớn hơn), **<** (nhỏ hơn), **>=** (lớn hơn hoặc bằng), **<=** (nhỏ hơn hoặc bằng).

Hai toán hạng của các phép toán này phải cùng kiểu. Ví dụ:

```
3 == 3 hoặc 3 == (4 -1)     // nhận giá trị 1 vì đúng
3 == 5                      // = 0 vì sai
3 != 5                      // = 1
3 + (5 < 2)                 // = 3 vì 5<2 bằng 0
3 + (5 >= 2)                // = 4 vì 5>=2 bằng 1
```

54

PHÉP TOÁN, BIỂU THỨC VÀ CÂU LỆNH

- **Các phép toán logic:**
  && (và), **||** (hoặc), **!** (không, phủ định)

- Ví dụ

```
3 && (4 > 5)          // = 0 vì có hạng thức (4>5) sai
(3 >= 1) && (7)       // = 1 vì cả hai hạng thức cùng đúng
!1                    // = 0
! (4 + 3 < 7)         // = 1 vì (4+3<7) bằng 0
5 || (4 >= 6)         // = 1 vì có một hạng thức (5) đúng
(5 < !0) || (4 >= 6)  // = 0 vì cả hai hạng thức đều sai
```

55

PHÉP TOÁN, BIỂU THỨC VÀ CÂU LỆNH

- **Các phép gán**

Phép gán thông thường: Đây là phép gán đã được trình bày trong mục trước.

Phép gán có điều kiện:

biến = (điều_kiện) ? <giá trị đúng>: <giá trị sai> ;

- Ví dụ:

```
x = (3 + 4 < 7) ? 10: 20      // x = 20 vì 3+4<7 là sai
x = (3 + 4) ? 10: 20          // x = 10 vì 3+4 khác 0, tức điều kiện đúng
x = (a > b) ? a: b            // x = số lớn nhất trong 2 số a, b.
```

56

PHÉP TOÁN, BIỂU THỨC VÀ CÂU LỆNH

Cách viết gọn của phép gán:

Một phép gán dạng x = x @ a ; có thể được viết gọn dưới dạng x @= a trong đó @ là các phép toán số học, xử lý bit,...

Ví dụ:

thay cho viết x = x + 2 có thể viết x += 2;

hoặc x = x/2 ; x = x*2 có thể được viết lại như x /= 2; x *= 2;

57

PHÉP TOÁN, BIỂU THỨC VÀ CÂU LỆNH

- **Biểu thức**

Biểu thức là dãy kí hiệu kết hợp giữa các toán hạng, phép toán và cặp dấu () theo một qui tắc nhất định. Các toán hạng là hằng, biến, hàm

58

Các Toán Tử -thứ tự ưu tiên

Priority	category	Example	Associativity
0	Primary expression	*identifiers constants*	None
1	Postfix	Function() () [] ->	left to right
2	Prefix and unary	! ~ + - ++ -- & sizeof	right to left
2	Type cast	(typeName)	right to left
3	Multiplicative	* / %	left to right
4	Additive	+ -	left to right
5	Shift	<< >>	left to right
6	Relational	< <= > >=	left to right
7	Equality	== !=	left to right
8	Boolean AND	&	left to right
9	Boolean XOR	^	left to right
10	Boolean OR	|	left to right
11	Logical AND	&&	left to right
12	Logical OR	||	left to right
13	conditional operator	?	Right to left
14	Assignment	= *= /= %= += -= &= ^= <<= >>=	right to left

59

Các toán tử so sánh và toán tử logic

		Relational and Quality Operators						Possible Mistakes		
X	Y	X < Y	X <= Y	X > Y	X >= Y	X != Y	X == Y	X=Y	X<<Y	X>>Y
3	3	0	1	0	1	0	1	3	24	0
3	4	1	1	0	0	1	0	4	48	0
4	3	0	0	1	1	1	0	3	32	0

		Logical Operators				Possible Mistakes	
X	Y	X && Y	X || Y	!X	!Y	X & Y	X | Y
0	0	0	0	1	1	0	0
0	7	0	1	1	0	0	7
5	0	0	1	0	1	0	5
5	7	1	1	0	0	5	7
8	7	1	1	0	0	0	15

60

Chuyển đổi kiểu

• ***Chuyển kiểu tự động***: về mặt nguyên tắc, khi cần thiết các kiểu có giá trị thấp sẽ được chương trình tự động chuyển lên kiểu cao hơn cho phù hợp với phép toán. Cụ thể phép chuyển kiểu có thể được thực hiện theo sơ đồ như sau:
char ⟶ int ⟶ long int ⟶ float ⟶ double

•***Ép kiểu***

(tên_kiểu)biểu_thức // cú pháp cũ trong C
hoặc:
tên_kiểu(biểu_thức) // cú pháp mới trong C++
- **Ví dụ**
int i;
float f = 3 ;
i = int(f + 2) ;//khi đó f+2 (bằng 5.0) được chuyển thành 5 và gán cho i

61

THƯ VIỆN CÁC HÀM TOÁN HỌC

Một số hàm này đều được khai báo trong file nguyên mẫu <**math.h**>
- Các hàm số học
+ abs(x), labs(x), fabs(x) : trả lại giá trị tuyệt đối của một số nguyên, số nguyên dài và số thực.
+ pow(x, y) : hàm mũ, trả lại giá trị x lũy thừa y (xy).
+ exp(x) : hàm mũ, trả lại giá trị e mũ x (ex).
+ log(x), log10(x) : trả lại lôgarit cơ số e và lôgarit thập phân của x (lnx, logx) .
+ sqrt(x) : trả lại căn bậc 2 của x.
+ atof(s_number) : trả lại số thực ứng với số viết dưới dạng xâu kí tự s_number.
- Các hàm lượng giác
+ sin(x), cos(x), tan(x) : trả lại các giá trị sinx, cosx, tgx.

62

Lịch sử ngôn ngữ C

- Ra đời trong những năm 1970, gắn liền với sự phát triển của HĐH Unix. Tác giả: Dennis Ritchie
- Mục tiêu:
  - Đề cao tính hiệu quả
  - Có khả năng truy xuất phần cứng ở cấp thấp
  - Ngôn ngữ có cấu trúc (thay cho lập trình bằng hợp ngữ)
- C là ngôn ngữ trung gian giữa cấp thấp…
  - Có khả năng truy xuất bộ nhớ trực tiếp
  - Cú pháp ngắn gọn, ít từ khoá
- … và cấp cao
  - Không phụ thuộc phần cứng
  - Cấu trúc, hàm, khả năng đóng gói
  - Kiểm tra kiểu dữ liệu

63

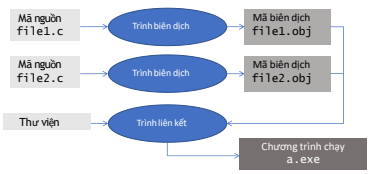
Lịch sử ngôn ngữ C++

- Ra đời năm 1979 bằng việc mở rộng ngôn ngữ C. Tác giả: Bjarne Stroustrup
- Mục tiêu:
  - Thêm các tính năng mới
  - Khắc phục một số nhược điểm của C
- Bổ sung những tính năng mới so với C:
  - Lập trình hướng đối tượng (OOP)
  - Lập trình tổng quát (template)
  - Nhiều tính năng nhỏ giúp lập trình linh hoạt hơn nữa (thêm kiểu bool, khai báo biến bất kỳ ở đâu, kiểu mạnh, định nghĩa chồng hàm, namespace, xử lý ngoại lệ,…)

64

Biên dịch chương trình C/C++

- Là quá trình chuyển đổi từ mã nguồn (do người viết) thành chương trình ở dạng mã máy để có thể thực thi được



65

Biên dịch chương trình C/C++

- Cho phép dịch từng file riêng rẽ giúp:
  - Dễ phân chia và quản lý từng phần của chương trình
  - Khi cần thay đổi, chỉ cần sửa đổi file liên quan
    → giảm thời gian bảo trì, sửa đổi
  - Chỉ cần dịch lại những file có thay đổi khi cần thiết
    → giảm thời gian dịch
- Các trình biên dịch hiện đại còn cho phép tối ưu hoá dữ liệu và mã lệnh
- Một số trình biên dịch thông dụng: MS Visual C++, gcc, Intel C++ Compiler, Watcom C/C++,…

66

IDE lập trình

- Dev - C++
- Codeblock: http://www.codeblocks.org/downloads/26

Link download: https://www.fosshub.com/Code-Blocks.html?dwl=codeblocks-17.12mingw-setup.exe

67

THUẬT TOÁN VÀ BIỂU DIỄN THUẬT TOÁN

Ngôn ngữ lập trình (Programming Language)
Chúng ta sẽ tìm hiểu một số khái niệm căn bản về thuật toán, chương trình, ngôn ngữ lập trình. Thuật ngữ "thuật giải" và "thuật toán" dĩ nhiên có sự khác nhau song trong nhiều trường hợp chúng có cùng nghĩa

- **Thuật giải** (Algorithm)

Là một dãy hữu hạn các thao tác xác định trên một đối tượng, sao cho sau khi thực hiện một số hữu hạn các bước thì đạt được mục tiêu. Theo R.A.Kowalski thì bản chất của thuật giải:

Thuật giải = Logic + Điều khiển

68

THUẬT TOÁN VÀ BIỂU DIỄN THUẬT TOÁN

* *Logic*: Đây là phần khá quan trọng, nó trả lời câu hỏi "Thuật giải làm gì, giải quyết vấn đề gì?", những yếu tố trong bài toán có quan hệ với nhau như thế nào v.v... Ở đây bao gồm những kiến thức chuyên môn mà bạn phải biết để có thể tiến hành giải bài toán.

Ví dụ 1: *Để giải một bài toán tính diện tích hình cầu, mà bạn không còn nhớ công thức tính hình cầu thì bạn không thể viết chương trình cho máy để giải bài toán này được.*

* *Điều khiển*: Thành phần này trả lời câu hỏi: giải thuật phải làm như thế nào?. Chính là cách thức tiến hành áp dụng thành phần logic để giải quyết vấn đề.

69

THUẬT TOÁN VÀ BIỂU DIỄN THUẬT TOÁN

Một thuật giải tốt là thuật giải:

- Chính xác
- Rõ ràng
- Đúng đắn
- Hiệu quả
- Và có thể bảo trì được.

70

THUẬT TOÁN VÀ BIỂU DIỄN THUẬT TOÁN

Chương trình (Program)

Là một tập hợp các mô tả, các phát biểu, nằm trong một hệ thống qui ước về ý nghĩa và thứ tự thực hiện, nhằm điều khiển máy tính làm việc. Theo Niklaus Wirth thì:

Chương trình = Thuật toán + Cấu trúc dữ liệu

Các thuật toán và chương trình đều có cấu trúc dựa trên 3 cấu trúc điều khiển cơ bản

* **Tuần tự (Sequential):** Các bước thực hiện tuần tự một cách chính xác từ trên xuống, mỗi bước chỉ thực hiện đúng một lần.

* **Chọn lọc (Selection):** Chọn 1 trong 2 hay nhiều thao tác để thực hiện.

* **Lặp lại (Repetition):** Một hay nhiều bước được thực hiện lặp lại một số lần.

71

THUẬT TOÁN VÀ BIỂU DIỄN THUẬT TOÁN

Ngôn ngữ lập trình (Programming language)
Ngôn ngữ lập trình là hệ thống các ký hiệu tuân theo các qui ước về ngữ pháp và ngữ nghĩa, dùng để xây dựng thành các chương trình cho máy tính.
Một chương trình được viết bằng một ngôn ngữ lập trình cụ thể (ví dụ Pascal, C…) gọi là chương trình nguồn, chương trình dịch làm nhiệm vụ dịch chương trình nguồn thành chương trình thực thi được trên máy tính.

72

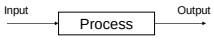
THUẬT TOÁN VÀ BIỂU DIỄN THUẬT TOÁN

Các bước lập trình
Bước 1: Phân tích vấn đề và xác định các đặc điểm.
(xác định I-P-O)
Bước 2: Lập ra giải pháp.
(đưa ra thuật giải)
Bước 3: Cài đặt.
(viết chương trình)
Bước 4: Chạy thử chương trình.
(dịch chương trình)
Bước 5: Kiểm chứng và hoàn thiện chương trình.
(thử nghiệm bằng nhiều số liệu và đánh giá)

73

THUẬT TOÁN VÀ BIỂU DIỄN THUẬT TOÁN

Biểu diễn thuật toán
1. I-P-O Cycle (Input-Process-Output Cycle) (Quy trình nhập-xử lý-xuất)
Quy trình xử lý cơ bản của máy tính gồm I-P-O.



Ví dụ : Xác định Input, Process, Output của việc làm 1 ly nước chanh nóng
Input : ly, đường, chanh, nước nóng, muỗng.
Process:- cho hỗn hợp đường, chanh, nước nóng vào ly.
- dùng muỗng khuấy đều.
Output : ly chanh nóng đã sẵn sàng để dùng.

74

Phương pháp lập trình

Ví dụ : Xác định Input, Process, Output của chương trình giải phương trình bậc nhất ax + b = 0

Input : hệ số a, b

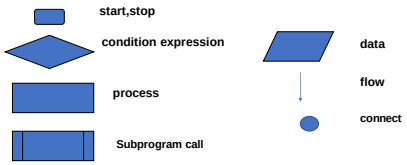
Process: chia - b cho a

Output : nghiệm x

75

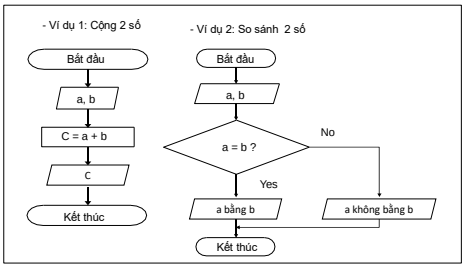
THUẬT TOÁN VÀ BIỂU DIỄN THUẬT TOÁN

2. Sử dụng lưu đồ (Flowchart)
Sử dụng sơ đồ xử lý để minh họa quá trình xử lý một chương trình.



76

THUẬT TOÁN VÀ BIỂU DIỄN THUẬT TOÁN



THUẬT TOÁN VÀ BIỂU DIỄN THUẬT TOÁN

3.Giả mã (PseudoCode).
Ưu điểm:
- Dễ viết và dễ sửa đổi, cho phép bổ sung, chèn thêm các lệnh một cách đơn giản
- Gần gũi với mọi người sử dụng

THUẬT TOÁN VÀ BIỂU DIỄN THUẬT TOÁN

3. Giả mã (PseudoCode).

Ví dụ (mã giả): Thay đổi giá trị của hai biến So1 và So2.

```
PROGRAM DoiCho
        INPUT So1           {Pha nhập số liệu}
        INPUT So2
        Tam = So1           {Pha xử lý}
        So1 = So2
        So2 = Tam
        OUTPUT So1 {Pha hiển thị kết quả}
        OUTPUT So2
END
```

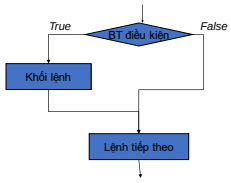
Kỹ Thuật Lập Trình

Các cấu trúc điều khiển

Cấu trúc rẽ nhánh – câu lệnh ĐK if

- **Dạng thứ nhất của lệnh if**

```
if (<biểu thức điệu kiện>)
  {
  <Khối lệnh>
  }
//lệnh tiếp theo
```



81

Ví dụ

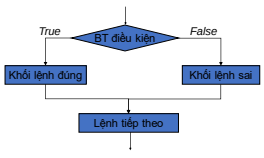
```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.    int b;
6.    cout<<"Enter a value:";
7.    cin>>b;
8.    if (b < 0)
9.              cout<<"The value is negative"<<endl;
10.   return 0;
11. }
```

82

Cấu trúc rẽ nhánh – câu lệnh ĐK if

Dạng thứ 2 của lệnh if

```
if (<Biểu thực điều kiện>)
      {
      <Khối lệnh đúng>
      }
else
      {
      <Khối lệnh sai>
      }
   //Lệnh tiếp theo
```



83

Ví dụ

```
...
cout<<"1/X is: ";
if(X)                  //if(x !=0)
  cout<<1/X;
else
  cout<<" undefined"<<endl;
...
```

84

Lỗi đơn giản nhưng dễ phạm

```
1.  #include <iostream>
2.  using namespace std;
3.  int main()
4.  {
5.      int b;
6.      cout<<"Enter a value:";
7.      cin>>b;
8.      if (b == 5)
9.          cout<<“b is ";
            cout<< “5 ”;
10.     return 0;
11. }
```

85

Lỗi đơn giản nhưng dễ phạm

```
1. Cout<<“1/X is: “;
2. if(X < 0) ;
3.      cout<<“ X is negative ”;
4. …
```

86

Ví dụ: Kiểm tra nhiều điều kiện

```
#include <iostream.h>
int main() {
        int b;
        cout<<"Enter a value:";
        cin>>b;
        if (b < 0)
                cout<<"The value is negative“<<endl;
        else if (b == 0)
                cout<<"The value is zero”<<endl;
        else
                cout<<"The value is positive“<<endl;
        return 0;
}
```

87

Điều kiện lồng nhau

- Câu lệnh if có thể được lồng vào nhau.

```
1. if ( X >= 0 ) {
2.     if ( Y < 0 )
3.             Y = Y + sqrt(X);
4. }
5. else
6.     Y = Y + sqrt(-X);
```

- Tuy nhiên, cần chú ý đến thứ tự các cặp lệnh if … else … khi lồng các lệnh if. Nếu không sẽ phát sinh lỗi.

```
1. if ( X >= 0 )
2.     if ( Y < 0 )
3.             Y = Y + sqrt(X);
4. else
5.     Y = Y + sqrt(-X);
```

88

Câu lệnh lựa chọn switch

- ***Cú pháp***

```
switch (biểu thức điều khiển)
{
    case biểu_thức_1: dãy lệnh 1 ;break;
    case biểu_thức_2: dãy lệnh 2 ; break;
    case ……..: …… ; break;
    case biểu_thức_n: dãy lệnh n ; break;
    [default: dãy lệnh n+1;]
}
```

- biểu thức điều khiển: phải có kiểu nguyên hoặc kí tự,
- các biểu_thức_i: được tạo từ các hằng nguyên hoặc kí tự,
- các dãy lệnh có thể rỗng. Không cần bao dãy lệnh bởi cặp dấu {},

89

Câu lệnh lựa chọn switch

- Ví dụ: In số ngày của một tháng bất kỳ nào đó được nhập từ bàn phím.

```
int th;
cout << “Cho biết tháng cần tính: “ ; cin >> th ;
switch (th)
{
    case 1: case 3: case 5: case 7: case 8: case 10:
    case 12: cout << "tháng này có 31 ngày" ; break ;
    case 2: cout << "tháng này có 28 ngày" ; break;
    case 4: case 6: case 9:
    case 11: cout << "tháng này có 30 ngày" ; break;
    default: cout << "Bạn đã nhập sai tháng, không có tháng này" ;
}
```

90

Câu lệnh lựa chọn switch (ví dụ)

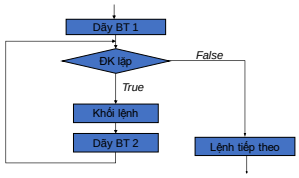
```
int main()
{  float a, b, c ;          // các toán hạng a, b và kết quả c
    char dau ;              // phép toán được cho dưới dạng kí tự
    cout << "Hãy nhập 2 số a, b: " ; cin >> a >> b ;
    cout << "và dấu phép toán: " ; cin >> dau ;
    switch (dau)
    {
        case '+': c = a + b ; break ;
        case '-': c = a - b ; break ;
        case 'x': case '.': case '*': c = a * b ; break ;
        case ':': case '/': c = a / b ; break ;
    }
    cout << setiosflags(ios::showpoint) << setprecision(4) ;
    cout << "Kết quả là: " << c ; // in 4 số lẻ
}
```

91

Cấu trúc lặp - lệnh for

- Cú pháp

for (dãy BT 1; ĐK lặp; dãy BT 2)
{
<Khối lệnh>
}



Cấu trúc lặp - lệnh for

- Ghi chú:

- Các biểu thức trong các dãy biểu thức 1, 2 cách nhau bởi dấu phẩy (,). Có thể có nhiều biểu thức trong các dãy này hoặc dãy biểu thức cũng có thể trống.
- Điều kiện lặp: là biểu thức lôgic (có giá trị đúng, sai).
- Các dãy biểu thức và/hoặc điều kiện có thể trống tuy nhiên vẫn giữ lại các dấu chấm phẩy (;) để ngăn cách các thành phần với nhau.

93

Cấu trúc lặp - lệnh for

- ***Cách thực hiện***

Khi gặp câu lệnh **for** trình tự thực hiện của chương trình như sau:

Thực hiện dãy biểu thức 1 (thông thường là các lệnh khởi tạo cho một số biến),

Kiểm tra điều kiện lặp, nếu đúng thì thực hiện khối lệnh lặp, sau đó thực hiện dãy biểu thức 2, quay lại kiểm tra điều kiện lặp và lặp lại quá trình trên cho đến bước nào đó việc kiểm tra điều kiện lặp cho kết quả sai thì dừng.

94

Lệnh for- ví dụ

- Tính tổng của dãy các số từ 1 đến 100

```
int main()
{
    int i, kq = 0;
    for (i = 1 ; i <= 100 ; i ++)
        kq += i ;
    cout << "Tổng = " << kq;
}
```

95

Lệnh for- ví dụ

In ra màn hình dãy số lẻ bé hơn một số n nào đó được nhập vào từ bàn phím

```
int main()
{
    int n, i ;
    cout << "Hãy nhập n = " ; cin >> n ;
    for (i = 1 ; i < n ; i += 2)
        cout << i << '\n' ;
}
```

96

Lệnh for lồng nhau

- Trong dãy lệnh lặp có thể chứa cả lệnh for, tức các lệnh for cũng được phép lồng nhau như các câu lệnh có cấu trúc khác.
- Ví dụ Bài toán cổ: vừa gà vừa chó bó lại cho tròn 36 con, 100 chân chẵn. Hỏi có mấy gà và mấy con chó.

```
void main()
{       int g, c ;
        for (g = 1 ; g < 50 ; g++)
          for (c = 1 ; c < 25 ; c++)
            if (g+c == 36 && 2*g+4*c == 100)
              cout << "số gà=" << g << ", số chó=" << c ;
}
```

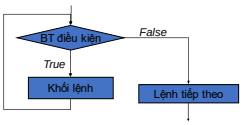
97

```
/*Tìm các phương an để có số tiền là 1000 từ các tờ tiền có mệnh giá:500, 200, 100*/
 #include<iostream>
using namespace std;
#define TONGSOTIEN 1000
int main()
{
  int i, j, k, soluong = 0;
   for(i=0; i<=TONGSOTIEN/500; i++)
    for(j=0; j<=TONGSOTIEN/200; j++)
      for(k=0; k<=TONGSOTIEN/100; k++)
            if((i*500 + j*200 + k*100 ) == TONGSOTIEN)
                {
                  cout<<"   "<<i<<"*500 + "<<j<<"*200 +
                          "<<k<<"*100\n";
                  soluong++;
                }
   cout<<"   ==>> Co tat ca "<<soluong<<" phuong an\n";
  return 0;
}
```

Cấu trúc lặp - lệnh while

- **Cú pháp**

while (Biểu thức điều kiện)
{
<Khối lệnh>
}



99

Cấu trúc lặp - lệnh while

Ghi chú:

- Khi <**Biểu thức điều kiện**> còn khác 0 (TRUE), <**khối lệnh**> tiếp tục được thực hiện. Nếu <**Biểu thức điều kiện**> bằng 0 (FALSE), lệnh while dừng và chương trình sẽ gọi lệnh kế tiếp sau while.
- Nếu lúc đầu <**Biểu thức điều kiện**> bằng 0 thì <**khối lệnh**> trong while không bao giờ được gọi thực hiện.
- Để vòng lặp không lặp vô hạn thì trong khối lệnh thông thường phải có ít nhất một câu lệnh nào đó gây ảnh hưởng đến kết quả của điều kiện.
- Nếu điều kiện luôn luôn nhận giá trị đúng (ví dụ biểu thức điều kiện là 1) thì trong khối lệnh lặp phải có câu lệnh kiểm tra dừng và lệnh break.

100

Lệnh while - ví dụ

Tìm ước chung lớn nhất (UCLN) của 2 số nguyên m và n.

Áp dụng thuật toán Euclide bằng cách liên tiếp lấy số lớn trừ đi số nhỏ khi nào 2 số bằng nhau thì đó là UCLN.

Trong chương trình ta qui ước m là số lớn và n là số nhỏ. Thêm biến phụ r để tính hiệu của 2 số. Sau đó đặt lại m hoặc n bằng r sao cho m > n và lặp lại.

Vòng lặp dừng khi m = n.

101

Lệnh while - ví dụ

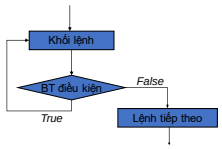
```
int main()
{
    int m, n, r;
    cout << "Nhập m, n: " ; cin >> m >> n ;
    if (m < n)
       { int t = m; m = n; n = t; }        // nếu m < n thì đổi vai trò hai số
    while (m != n)
    {
        r = m - n ;
        if (r > n) m = r; else { m = n ; n = r ; }
     }
    cout << "UCLN = " << m ;
}
```

102

Cấu trúc lặp - lệnh do.. while

Cú pháp

do
{
<Khối lệnh>
}
while (BT điều kiện) ;



103

Cấu trúc lặp - lệnh do while

Ghi chú
- Thực hiện lệnh <Khối lệnh>, sau đó kiểm tra <Biểu thức điều kiện>. Nếu <Biểu thức điều kiện> bằng 0, lệnh sẽ dừng.
- Nếu không, thực hiện tiếp <Khối lệnh>.
- Lệnh do.. while thực hiện <Khối lệnh> ít nhất một lần.

104

Lệnh do while – ví dụ

Ví dụ: Kiểm tra một số n có là số nguyên tố.
Để kiểm tra một số n > 3 có phải là số nguyên tố ta lần lượt chia n cho các số i đi từ 2 đến một nửa của n. Nếu có i sao cho n chia hết cho i thì n là hợp số ngược lại n là số nguyên tố.

```
int main()
{   int i, n ;                    // n: số cần kiểm tra
    cout << "Cho biết số cần kiểm tra: " ; cin >> n ;
    i = 2 ;
    do {
      if (n%i == 0)
        { cout << n << "là hợp số" ;
          return ;                    // dừng chương trình
        } i++;
    } while (i <= n/2);
    cout << n << "là số nguyên tố" ;
}
```

105

Một số lệnh sử dụng với vòng lặp

- Lệnh break Dùng để thoát khỏi vòng lặp giữa chừng.

  cú pháp:
  break;

- Thường sử dụng cùng với lệnh if để kiểm tra điều kiện dừng trước khi dùng lệnh break.
- Bài tập: Viết chương trình nhập vào một số rồi tìm số nguyên tố đầu tiên lớn hơn số vừa nhập

106

Một số lệnh sử dụng với vòng lặp

Lệnh continue Bỏ qua các lệnh kế tiếp trong một vòng lặp và bắt đầu vòng lặp tiếp theo.

cú pháp:
continue;

- Chỉ áp dụng với lệnh lặp.
- Bài tập: Viết chương trình nhập vào một số và tìm ra tất cả các thừa số nguyên tố của số đó.

107