KỸ THUẬT LẬP TRÌNH

CHƯƠNG 1. TỔNG QUAN VỀ C/C++

1. Lập trình và ngôn ngữ lập trình
2. Quy trình viết và thực thi chương trình
3. Cấu trúc của một chương trình C/C++
4. Bộ kí tự và từ khóa
5. Tên gọi
6. Các kiểu dữ liệu cơ bản
7. Biến và hằng
8. Chú thích
9. Các lệnh vào ra

Lập trình và ngôn ngữ lập trình

- **Lập trình** (Programming): kỹ thuật cài đặt thuật toán bằng NNLT tạo CTMT.
- **NNLT** (Programming language) được thiết kế, chuẩn hóa để truyền chỉ thị cho MT.
- NNLT tạo ra CT nhằm điều khiển MT hoặc mô tả thuật toán để người khác đọc hiểu.
- **Phân loại NNLT**
  - NN máy (mã máy: 0/1)
    - NN nền tảng của bộ vi xử lý, NN duy nhất máy tính hiểu
    - Các CT viết các NN khác =>NNM=> thực thi

Lập trình và ngôn ngữ lập trình

- Hợp ngữ
  - Gần như NNM, dùng ký hiệu gợi nhớ (mã lệnh hình thức) biểu diễn các mã lệnh của máy.
  - CT hợp ngữ => mã máy: thông qua trình hợp dịch (assembler).
- NN cấp cao
  - Danh từ, động từ, ký hiệu toán học, liên hệ, thao tác luận lý
  - CT viết bằng NN cấp cao chạy trên các loại MT khác nhau.

❖ NNLT thông dụng

- Visual Basic (Visual Basic .NET)
- JAVA

Lập trình và ngôn ngữ lập trình

- C/C++
- C#
- FORTRAN – (FORmula TRANslator)
- PASCAL
- PHP (Hypertext Preprocessor)
- JavaScript
- SQL (Structured Query Language)
- Lisp

Qui trình viết và thực thi CT

- Ý tưởng, phân tích yêu cầu (requirements analysis);
- Đặc tả (specification);
- Thiết kế (design and architecture);
- Lập trình (coding);
- Biên dịch (compilation);
- Kiểm thử (testing);
- Viết tài liệu (documentation);
- Bảo trì (maintenance).

Bộ kí tự và từ khóa (Character set and keyword)

- **Bộ kí tự (Character set):** Có phân biệt hoa thường
  - 26 chữ cái Latinh lớn: A, B, C..., Z
  - 26 chữ cái Latinh nhỏ: a, b, c ..., z
  - 10 chữ số thập phân: 0, 1, 2...9
  - Các ký hiệu toán học: +, -, *, /, =, <, >
  - Các ký hiệu đặc biệt: . , ; : " ' _ % # ! ^ [ ] { } ( ) ...
  - Dấu cách hay khoảng trống, xuống hàng (\n) và tab (\t)

Bộ kí tự và từ khóa (Character set and keyword)

❑ Bộ từ khóa (Keywords):

asm	*auto*	*bool*	*break*
case	*catch*	*char*	*class*
const	*const_cast*	*continue*	*default*
delete	*else*	*extern*	*do*
enum	*false*	*double*	*explicit*
float	*dynamic_cast*	*export*	*for*
friend	*goto*	*if*	*inline*
int	*long*	*mutable*	*namespace*
new	*operator*	*private*	*protected*
public	*register*	*reinterpret_cast*	*return*
short	*signed*	*sizeof*	*static*
static_cast	*struct*	*switch*	*template*
this	*throw*	*true*	*try*
typedef	*typeid*	*typename*	*union*
unsigned	*using*	*virtual*	*void*
volatile	*wchar_t*	while	

Định danh (identifier)

- Một dãy kí tự để đặt tên: biến, hằng, hàm, mảng,…
- Qui tắc:
  - Dùng: chữ cái (A..Z,a..z), chữ số (0..9)
  - Dấu gạch dưới '_'
  - Không bắt đầu bằng số.
  - Không trùng với từ khóa.

Các kiểu dữ liệu cơ bản (Base type)

Kiểu	Kích cỡ thông dụng (tính bằng bit)	Phạm vi tối thiểu
char	8	-127 to 127
signed char	8	-127 .. 127
unsigned char	8	0 .. 255
int	16/32	-32767 .. 32767
signed int	16/32	-nt-
unsigned int	16/32	0 .. 65535
short	16	-32767 .. 32767
signed short	16	nt
unsigned short	16	0 .. 65535
long	32	-2147483647..2147483647
signed long	32	- nt-
unsigned long	32	0 .. 4294967295
float	32	Độ chính xác 6 chữ số
double	64	Độ chính xác 15 chữ số
long double	80	Độ chính xác 17 chữ số
bool (C++)	-	-
wchar_t (C++)	16	-32767 .. 32767

Biến và hằng (Variable and constant)

- Biến (Variable):
  - Để lưu trữ dữ liệu trong bộ nhớ máy tính do người LT định nghĩa (khai báo biến) đặt bởi một tên.
  - Có thể thay đổi giá trị.
  - Kiểu dữ liệu thì không.
  - Dùng thì phải khai báo.
  - Khai báo:
    - <Kiểu_dữ _liệu> tên_biến_1, tên_biến_2,… ;

    Ví dụ: int x,y,z;
    - Khởi tạo:

      Khai báo và khởi tạo: int x=7; int x(7); int x{7};

Biến và hằng (Variable and constant)

- Hằng (Constant):
  - Không đổi trong suốt quá trình thực thi của CT.
  - Có thể : một chuỗi ký tự, ký tự, con số xác định.
  - Khai báo
    - ***#define*** ***Tên_hằng*** ***Giá_trị***
    - ***const*** ***Kiểu_dữ_liệu*** ***Tên_hằng*** **=** ***Giá_trị*** **;**

Ví dụ:

```
#define  PI  3.14
const int MAX = 100;
```

Biểu thức (Expressions)

Sự kết hợp các toán tử và các toán hạng

Toán Tử

Ví dụ: **2 * y + 5**

Toán hạng

Biểu thức số học

Biểu thức số học có thể được biểu diễn trong C/C++ bằng cách sử dụng các toán tử số học

Ví dụ :

++i % 7

5 + (c = 3 + 8)

a * (b + c/d) - 22

Toán tử gán (Assignment operator)

Toán tử gán (=) có thể được dùng với bất kỳ biểu thức C/C++ hợp lệ nào

variable_name = expression;

(Tên biến) → **lvalue** **(Giá trị trái)**

= → **Assignment operator** **(Toán tử gán)**

(Biểu thức) → **rvalue** **(Giá trị phải)**

Gán liên tiếp

Nhiều biến có thể được gán với cùng một giá trị trong một câu lệnh đơn

a = b = c = 10; ✓

Tuy nhiên, không thể áp dụng quy tắc trên khi khai báo biến

int a = int b = int b = int c = 10 ✗

Toán tử số học (Arithmetic operator)

Các phép toán hai ngôi số học là

Phép toán	Ý nghĩa	Ví dụ
+	Phép cộng	a+b
-	Phép trừ	a-b
*	Phép nhân	a*b
/	Phép chia	a/b (Chia số nguyên sẽ chặt phần thập phân)
%	Phép lấy phần dư	a%b (Cho phần dư của phép chia a cho b)

Có phép toán một ngôi - ví du -(a+b) sẽ đảo giá trị của phép cộng (a+b).

Các toán tử gán phức hợp (Compound assignment operators)

Các toán tử:

+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=

Ví dụ:

a += 5; tương đương với **a = a + 5;**

a -= 5; tương đương với **a = a - 5;**

a /= b; tương đương với **a = a / b;**

a *= b + 1; tương đương với **a = a * (b + 1);**

Toán tử tăng và giảm một đơn vị (Increment and decrement)

Các toán tử: ++ và --

a++;

a+=1;

a=a+1

Các câu lệnh là tương đương.

Lưu ý: các toán tử ++/-- đặt trước hay sau trong biểu thức

Ví dụ 1

B=3;

A=++B;

// A is 4, B is 4

Ví dụ 2

B=3;

A=B++;

// A is 3, B is 4

Toán tử quan hệ và so sánh (Relational and comparison operators)

Toán tử	Ý nghĩa
>	Lớn hơn
>=	Lớn hơn hoặc bằng
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
==	Bằng
!=	Không bằng
Kết quả cho true hoặc false	

Toán tử logic (Logical operator)

Toán tử	Ý nghĩa
&&	**AND**: Kết quả là True khi cả 2 điều kiện đều đúng
||	**OR** : Kết quả là True khi chỉ một trong hai điều kiện là đúng
!	**NOT**: Tác động trên các giá trị riêng lẻ, chuyển đổi True thành False và ngược lại

Ví dụ: if (a>10) && (a<20)

Toán tử thao tác bit (Bitwise operator)

Toán tử	Mô tả
Bitwise AND (x & y)	Mỗi vị trí của bit trả về kết quả là 1 nếu bit của hai toán hạng là 1, còn lại là 0
Bitwise OR (x | y)	Mỗi vị trí của bit trả về kết quả là 1 nếu bit của một trong hai toán hạng là 1.
Bitwise NOT (~ x)	Đảo ngược giá trị của toán hạng (1 thành 0 và ngược lại).
Bitwise XOR (x ^ y)	Giống nhau cho 0, khác nhau cho 1 (0^0 = 0, 1^1 = 0, 0^1 = 1, 1^0 = 1)
Bitwise SHL (x <<n)	Dịch sang trái n bit, tương đương x* 2^n
Bitwise SHR (x >>n)	Dịch sang phải n bit, tương đương x/ 2^n

Toán tử thao tác bit (Bitwise operator)

Ví dụ

- 10 & 15→ **1010 & 1111→1010** → 10
- 10 | 15→ **1010 | 1111→1111** → 15
- 10 ^ 15→ **1010 ^ 1111→0101** → 5

Một số toán tử khác (other operators)

Toán tử	Ý nghĩa
sizeof	Toán tử sizeof trả về kích cỡ của một biến. Ví dụ: sizeof(a), với a là int, sẽ trả về 4
Điều kiện ? X : Y	Toán tử điều kiện. Nếu điều kiện là true ? thì nó trả về giá trị X : nếu không thì trả về Y s1 = (1 > 2) ? 2912 : 1706; s1=1706
,	Toán tử phẩy làm cho một dãy hoạt động được thực hiện. Giá trị của toàn biểu thức phẩy là giá trị của biểu thức cuối cùng trong danh sách được phân biệt bởi dấu phảy (biểu _thức_1, biểu_thức_2,…, biểu_thức_n) m = (t = 2, t*t + 3); ⇔ m = 7
. (dot) và -> (arrow)	Toán tử thành viên được sử dụng để tham chiếu các phần tử đơn của các lớp, các cấu trúc, và union

Một số toán tử khác (other operators)

Toán tử	Ý nghĩa
cast	Toán tử ép kiểu (Casting) trong C++ biến đổi một kiểu dữ liệu thành kiểu khác. Ví dụ: int(2.2000) sẽ trả về 2. Cú pháp: type_cast <new_type> (expression); type: const, dynamic, reinterpret, static Ví dụ: reinterpret_cast<char*>(&x);
&	Toán tử con trỏ &_ trả về địa chỉ của một biến. Ví du: &a; sẽ trả về địa chỉ thực sự của biến này
*****	Toán tử con trỏ * là trỏ tới một biến. Ví dụ: *var sẽ trỏ tới một biến va

Chuyển đổi kiểu (type conversion)

❖ **Chuyển đổi kiểu ngầm định:**

- Trong cùng 1 biểu thức, nếu các toán hạng không cùng kiểu với nhau thì trước khi tính toán giá trị của biểu thức, chương trình dịch sẽ thực hiện việc chuyển đổi kiểu ngầm định (nếu được) theo nguyên tắc "Kiểu có phạm vi giá trị biểu diễn nhỏ hơn sẽ được chuyển sang kiểu có phạm vị giá trị biểu diễn lớn hơn".
- Sơ đồ chuyển đổi kiểu ngầm định:

char* → *int* → *long* → *float* → *double* → *long double

Ép kiểu

- **Ép kiểu (type casting)**: Trong một số trường hợp, ta bắt buộc phải sử dụng ép kiểu để tạo ra một biểu thức hợp lệ như sau:
  - (<tên kiểu>) (<biểu thức>)

- Ví dụ, để tạo biểu thức số học hợp lệ sau 8.0 % 3, ta cần thực hiện ép kiểu như sau:
- (int) 8.0 % 3

Độ ưu tiên của toán tử (Precedence of operators)

Level (Độ ưu tiên)	Precedence group (Nhóm ưu tiên)	Operator (Toán tử)	Grouping (Thứ tự thực hiện)
1	Scope (Phạm vi)	::	Left-to-right
2	Postfix (unary) Hậu tố (một ngôi)	++ --	Left-to-right
		()	
		[]	
		. ->	
3	Prefix (unary) Tiền tố (một ngôi)	++ --	Right-to-left
		~ !	
		+ -	
		& *	
		new delete	
		sizeof	
		(type)	

Độ ưu tiên của toán tử (Precedence of operators)

4	**Pointer-to-member (Trỏ đến thành viên)**	**.* ->***	**Left-to-right**
5	Arithmetic: scaling (Số học: tỷ lệ)	* / %	Left-to-right
6	Arithmetic: addition (Số học: tăng giảm)	+ -	Left-to-right
7	Bitwise shift (Dịch bit)	<< >>	Left-to-right
8	Relational (So sánh hơn)	< > <= >=	Left-to-right
9	Equality (So sánh bằng)	== !=	Left-to-right
10	Bit AND	&	Left-to-right
11	Bit XOR	^	Left-to-right
12	Bit OR	|	Left-to-right
13	Logical AND	&&	Left-to-right
14	Logical OR	||	Left-to-right
15	Assignment-level expressions (Gán)	= *= /= %= += -= >>= <<= &= ^= |=	Right-to-left
		?:	
16	Sequencing (Sắp xếp)	,	Left-to-right

Câu lệnh và khối lệnh (statement and block statement)

- **Câu lệnh:**
  - Là một chỉ thị nhằm ra lệnh cho chương trình thực hiện một tác vụ cụ thể nào đó.
  - Mỗi câu lệnh có thể được viết trên một hoặc nhiều dòng.
  - Mỗi câu lệnh được kết thúc bằng ***dấu chấm phẩy.***

(*dấu chấm phẩy dùng ngăn cách các câu lệnh).*

- Câu lệnh được phân chia thành 2 loại:

  - **Câu lệnh đơn**:

  Là câu lệnh không chứa câu lệnh khác.

  Ví dụ: câu lệnh gán, lệnh khai báo, lệnh xuất nhập...

Câu lệnh và khối lệnh (statement and block statement)

- Câu lệnh phức:

Là câu lệnh có chứa câu lệnh khác bên trong nó.

Ví dụ: khối lệnh, câu lệnh rẽ nhánh, câu lệnh lặp, ...

- **Khối lệnh:**
  - Gồm một hoặc nhiều câu lệnh đơn được bao bởi cặp dấu ngoặc **{}**.
  - Một khối lệnh có thể lồng bên trong nó một hoặc nhiều khối lệnh khác.

Chú thích (Comment)

- Trên một dòng: //Nội dung
- Một hoặc nhiều dòng: /* Nội dung */
- Bất kỳ vị trí nào trong CT
- Không ảnh hưởng đến kết quả
- Nên sử dụng

Các lệnh vào/ ra dữ liệu (Input/output)

- Đưa dữ liệu ra màn hình:

```
cout << bt_1 ;
cout << bt_2 ;
cout << bt_3 ;
…
```

hoặc:

```
cout << bt_1<< bt_2 << bt_3…<<bt_n ;
```

- Bt chuỗi ký tự thì đặt nội dung trong “ ”

Các lệnh vào/ ra dữ liệu (Input/output)

- Nhập dữ liệu cho biến từ bàn phím:

cin >> biến_1 ;

cin >> biến_2 ;

cin >> biến_3 ;

hoặc:

cin >> biến_1 >> biến_2 >> biến_3 ;

- cin.get(c): nhập một kí tự vào biến kí tự c
- cin.getline(s, n): nhập tối đa n-1 kí tự vào xâu s (mảng ký tự : char s[100])
- getline(cin, s): Nhập chuỗi nhưng khai báo string s;

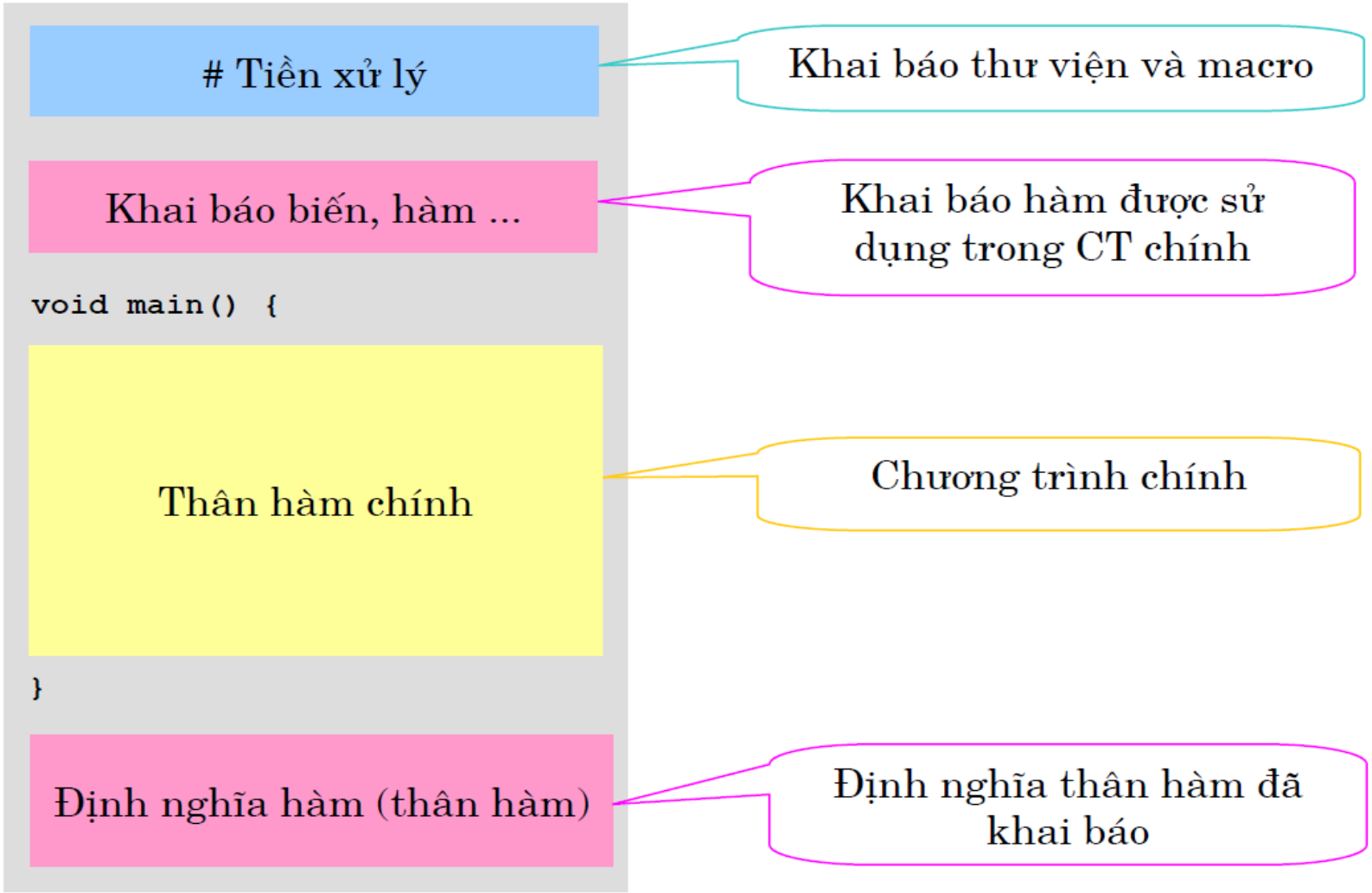
Các lệnh vào/ ra dữ liệu (Input/output)

- cin.ignore(): để lấy ra kí tự xuống dòng còn sót lại trong bộ đệm.
- endl: Xuống dòng ('\n')
- Định dạng in ra màn hình:

  Cần khai báo #include<iomanip>

- setw(n): Qui định độ rộng dành để in ra các giá trị là n cột màn hình.
- setprecision(n): Chỉ định số chữ số in ra là n.
- setprecision(n) kết hợp với fixed: Chỉ định số chữ số của phần thập phân in ra là n. Số sẽ được làm tròn trước khi in ra.

Cấu trúc CT (Structure of a program)



Cấu trúc CT (Structure of a program)

```
#include<iostream>
#include<cmath>
using namespace std;
void swap(int &,int &);
int main()
{
        int x,y;
        cout<<"Nhap x,y = ";
        cin>>x>>y;
        swap(x,y);
        cout<<" x= "<<x<<"y = "<<y;
        system("pause");
        return 0;
}
void swap(int &a,int &b)
{
        int t;
        t=a;a=b;b=t;
}
```