NGÔN NGỮ LẬP TRÌNH C++

Các phương pháp lập trình và ngôn ngữ lập trình C++

Nội dung

1. Lập trình tuyến tính
2. Lập trình hướng cấu trúc
3. Lập trình hướng đối tượng
4. Giới thiệu về ngôn ngữ lập trình C++

Lập trình tuyến tính

Đặc trưng

Đơn giản: chương trình được tiến hành đơn giản, theo lối tuần tự, không phức tạp

Đơn luồng: chỉ có một luồng duy nhất, các công việc được thực hiện tuần tự trong luồng đó

Ưu điểm

Chương trình đơn giản dễ hiểu; Phù hợp với các chương trình đơn giản

Nhược điểm

Không sử dụng được cho các chương trình phức tạp

Ngày nay, lập trình tuyến tính chỉ được sử dụng trong các chương trình, mô đun nhỏ

Lập trình hướng cấu trúc

Đặc trưng

- Chương trình chính được chia thành nhiều chương trình con, mỗi chương trình con thực hiện một công việc xác định
- **Chương trình = Cấu trúc dữ liệu + giải thuật**

Ưu điểm

- Chương trình sáng sủa, dễ hiểu, dễ theo dõi
- Tư duy giải thuật rõ ràng

Nhược điểm

- Không hỗ trợ việc sử dụng lại mã nguồn, Giải thuật phụ thuộc chặt chẽ vào cấu trúc dữ liệu, Khi thay đổi cấu trúc dữ liệu phải viết lại chương trình
- Không phù hợp với phần mềm lớn. Chỉ phù hợp với các bài toán nhỏ

Phương pháp thiết kế top-down

- Phương pháp thiết kế top-down tiếp cận bài toán theo hướng từ trên xuống dưới, từ tổng quan đến chi tiết. Theo đó, một bài toán được chia thành các bài toán con nhỏ hơn. Mỗi bài toán con lại được chia nhỏ tiếp, nếu có thể, thành các bài toán con nhỏ hơn nữa
- Quá trình này còn được gọi là quá trình làm mịn dần. Quá trình làm mịn dần sẽ dừng lại khi các bài toán con không cần chia nhỏ thêm nữa. Nghĩa là khi mỗi bài toán con đều có thể giải quyết bằng một chương trình con với một giải thuật đơn giản

Lập trình hướng đối tượng

- Coi các thực thể trong chương trình là các đối tượng và sau đó, người ta trừu tượng hoá đối tượng thành lớp đối tượng
- Dữ liệu được tổ chức thành các thuộc tính của lớp
- Quan hệ giữa các đối tượng là quan hệ ngang hàng hoặc quan hệ kế thừa: Nếu lớp B kế thừa từ lớp A thì A được gọi là *lớp cơ sở* và B được gọi là *lớp dẫn xuất*

Đặc trưng

- **Đóng gói dữ liệu**, dữ liệu được tổ chức thành các thuộc tính của đối tượng và truy cập thông qua phương thức
- **Khả năng sử dụng lại mã nguồn** thông qua tính kế thừa

Ưu điểm

- Hạn chế nguy cơ bị thay đổi dữ liệu tự do trong chương trình
- Khi thay đổi dữ liệu của một đối tượng, chỉ cần thay đổi một số thành phần trong chương trình, không ảnh hưởng đến các đối tượng khác
- Có thể sử dụng lại mã nguồn, tiết kiệm thời gian, tài nguyên hệ thống
- Phù hợp với các dự án lớn, phức tạp

Đặc điểm

- Kế thừa các đặc tính của ngôn ngữ lập trình **C** (Mọi chương trình viết bằng ngôn ngữ C đều có thể thực thi được trên C++)
- Hỗ trợ các nguyên lý lập trình hướng đối tượng
  - Trừu tượng (Data abstraction): Cung cấp công cụ tạo lớp (class) để mô tả các đối tượng
  - Đóng gói (Data encapsulation): Điều khiển việc truy cập các thuộc tính của đối tượng
  - Kế thừa (Inheritance): Kế thừa một hoặc nhiều lớp cơ sở để mô tả các đối tượng
  - Đa hình (Polymorphism): Cài đặt các cơ chế thực hiện chương trình khác nhau

Danh sách các từ khoá trong C++

alignas (since C++11)	char8_t (since C++20)	default(1)	inline(1)	reflexpr (reflection TS)	throw
alignof (since C++11)	char16_t (since C++11)	delete(1)	int	register(2)	true
and	char32_t (since C++11)	do	long	reinterpret_cast	try
and_eq	class(1)	double	mutable(1)	requires (since C++20)	typedef
asm	compl	dynamic_cast	namespace	return	typeid
atomic_cancel (TM TS)	concept (since C++20)	else	new	short	typename
atomic_commit (TM TS)	const	enum	noexcept (since C++11)	signed	union
atomic_noexcept (TM TS)	consteval (since C++20)	explicit	not	sizeof(1)	unsigned
auto(1)	constexpr (since C++11)	export(1)(3)	not_eq	static	using(1)
bitand	constinit (since C++20)	extern(1)	nullptr (since C++11)	static_assert (since C++11)	virtual
bitor	const_cast	false	operator	static_cast	void
bool	continue	float	or	struct(1)	volatile
break	co_await (since C++20)	for	or_eq	switch	wchar_t
case	co_return (since C++20)	friend	private	synchronized (TM TS)	while
catch	co_yield (since C++20)	goto	protected	template	xor
char	decltype (since C++11)	if	public	this	xor_eq
				thread_local (since C++11)	

Cấu trúc chương trình trong C++

Chương trình viết trên ngôn ngữ C++ gồm 3 thành phần

- Khai báo thư viện
- Mô tả các hàm, lớp
- Chương trình chính

Các kiểu dữ liệu trong C++

Các kiểu dữ liệu

Kiểu dữ liệu (type) là tên dùng để chỉ các đối tượng thuộc miền xác định cùng với các phép toán trên nó

Kiểu (type)	Từ khóa (keyword)
Kiểu logic	bool
Kiểu ký tự	char
Kiểu số nguyên	int
Kiểu số nguyên dài	long
Kiểu số thực	float
Kiểu số thực có độ chính xác kép	double
Kiểu ký tự rộng	wchar_t
Không có kiểu	void

Các kiểu dữ liệu mở rộng

Kiểu	Kích thước	Phạm vi
char	1byte	-127 đến 127 hoặc 0 đến 255
unsigned char	1byte	0 đến 255
signed char	1byte	-127 đến 127
int	4bytes	-2147483648 đến 2147483647
unsigned int	4bytes	0 đến 4294967295
signed int	4bytes	-2147483648 đến 2147483647
short int	2bytes	-32768 đến 32767
unsigned short int	2bytes	0 đến 65,535
signed short int	2bytes	-32768 đến 32767

Kiểu	Kích thước	Phạm vi
long int	8bytes	-2,147,483,648 đến 2,147,483,647
signed long int	8bytes	như long int
unsigned long int	8bytes	0 đến 2^32 - 1
long long int	8bytes	-(2^63) đến (2^63)-1
unsigned long long int	8bytes	0 đến 2^64 -1
float	4bytes	
double	8bytes	
long double	12bytes	
wchar_t	2 or 4 bytes	1 wide character

Từ các kiểu dữ liệu cơ bản, một số kiểu dữ liệu sửa đổi được kết hợp với các từ khóa signed (có dấu), unsigned (không dấu), short (ngắn), long (dài) để tạo nên các kiểu dữ liệu khác.

Biến trong C++

Biến (variable) là một tên thuộc kiểu. Mỗi tên chứa đựng bộ ba: tên biến, địa chỉ và giá trị

Mọi biến đều phải khai báo trước khi sử dụng

Khai báo biến trong C++

[Kiểu dữ liệu] biến1, biến2 = giá trị;

Ví dụ:

```
int a, b;
char c;
float d = 1.0;
```

Biến

Biến toàn cục

- Biến toàn cục (global variable): Được khai báo ngoài tất cả các hàm (kể cả hàm main)
- Biến toàn cục được cấp phát không gian nhớ từ khi bắt đầu chương trình cho đến khi kết thúc
- Biến toàn cục được truy cập và thay đổi nội dung từ bất kỳ vị trí nào trong chương trình

Biến cục bộ

- Biến cục bộ (local variable): Được khai báo bên trong một hàm hoặc bên trong thân của khối lệnh
- Biến cục bộ được cấp phát bộ nhớ khi có lời gọi hàm hoặc gọi đến khối lệnh chứa nó
- Biến cục bộ có phạm vi hoạt động bị giới hạn trong hàm hoặc khối lệnh

Ưu tiên sử dụng biến cục bộ, hạn chế sử dụng biến toàn cục

Hằng

HẰNG

Cú pháp

#define <constant name> <value>

Ví dụ

```
#define   MAX    100          // không có = hay ;
#define   PI     3.14
```

Toán tử trong ngôn ngữ lập trình C++

Các phép toán số học

Phép toán	Mô tả	Ví dụ a = 10, b = 20
+	Phép cộng hai số	a + b = 30
-	Phép trừ hai số	a – b = -10
*	Phép nhân hai số	a * b = 200
/	Phép chia hai số	b/a = 2
%	Phép lấy phần dư hai số	a%b =10
++	Tăng số lên 1 đơn vị	a++ ⇔ a = a +1 =11
--	Giảm số đi 1 đơn vị	a-- ⇔ a = a -1 =9

Toán tử trong ngôn ngữ lập trình C++

Các phép toán quan hệ

Phép toán	Mô tả	Ví dụ a = 10, b = 20
==	Đúng bằng	(a == b) là sai (0)
!=	Khác nhau	(a != b) là đúng (1)
>	Lớn hơn	(a > b) là sai (0)
<	Nhỏ hơn	(a < b) là đúng (1)
>=	Lớn hơn hoặc bằng	(a >= b) là sai (0)
<=	Nhỏ hơn hoặc bằng	(a <= b) là đúng (1)

Các toán tử logic

Tổ hợp nhiều biểu thức quan hệ với nhau.

&& (and), || (or), ! (not)

&&	0	1
0	0	0
1	0	**1**

||	0	1
0	0	**1**
1	**1**	**1**

Ví dụ

```
s1 = (1 > 2) && (3 > 4);
s2 = (1 > 2) || (3 > 4);
s3 = !(1 > 2);
```

Toán tử điều kiện

Đây là toán tử 3 ngôi (gồm có 3 toán hạng)

<biểu thức 1> ? <biểu thức 2> : <biểu thức 3>

<biểu thức 1> đúng thì giá trị là <biểu thức 2>.

<biểu thức 1> sai thì giá trị là <biểu thức 3>.

Ví dụ

```
s1 = (1 > 2) ? 2912 : 1706;
int s2 = 0;
1 < 2 ? s2 = 2912 : s2 = 1706;
```

Toán tử trong ngôn ngữ lập trình C++

Các phép toán thao tác cấp bit

Phép toán	Mô tả	Ví dụ a = 60 = 0011 1100, b = 13 = 0000 1101
&	Phép và cấp bít	a ℘ b = 0000.1100 = 12
|	Phép hoặc cấp bít	(a | b) =0011.1101 = 61
∧	Phép tuyển cấp bít	(a ∧ b) = 0011.0001 = 49
~	Phép phủ định cấp bít	~(a) = 1100.0011 = -61
<<	Phép dịch trái cấp bít	(a <<2) = 1111.0000=240
>>	Phép dịch phải cấp bít	(a >> b) = 0000. 1111 =15

Các toán tử trên bit

Tác động lên các bit của toán hạng (nguyên).

& (and), | (or), ^ (xor), ~ (not hay lấy số bù 1)

>> (shift right), << (shift left)

Toán tử gộp: &=, |=, ^=, ~=, >>=, <<=

&	0	1
0	0	0
1	0	**1**

|	0	1
0	0	**1**
1	**1**	**1**

^	0	1
0	0	**1**
1	**1**	0

~	0	1
	1	0

Toán tử trong ngôn ngữ lập trình C++

Toán tử trên bit (Ví dụ)

```
int main() {
        int a = 5;   // 0000 0000 0000 0101
        int b = 6;   // 0000 0000 0000 0110
        int z1, z2, z3, z4, z5, z6;
        z1 = a & b;// 0000 0000 0000 0100
        z2 = a | b; // 0000 0000 0000 0111
        z3 = a ^ b;// 0000 0000 0000 0011
        z4 = ~a;    // 1111 1111 1111 1010
        z5 = a >> 2;// 0000 0000 0000 0001
        z6 = a << 2;// 0000 0000 0001 0100
}
```

Toán tử trong ngôn ngữ lập trình C++

Các phép toán thao tác số học mở rộng

Phép toán	Mô tả	Ví dụ
=	Phép gán	c = a + b sẽ gán giá trị a + b cho c
+=	Phép cộng và gán	c+= a tương đương c = c + a
-=	Phép trừ và gán	c -= a tương đương c = c - a
*=	Phép nhân và gán	c *= a tương đương c = c * a
/=	Phép chia và gán	c /= a tương đương c = c / a
<<=	Phép dịch trái cấp bit và gán	c <<= 1 tương đương c = c << 1
>>=	Phép dịch phải cấp bit và gán	c >>= 1 tương đương c = c >> 1
&=	Phép và cấp bit và gán	c &= 1 tương đương c = c & 1
^=	Phép tuyển cấp bit và gán	c ^= 1 tương đương c = c ^ 1
|=	Phép hoặc cấp bit và gán	c |= 1 tương đương c = c | 1

Toán tử trong ngôn ngữ lập trình C++

Toán tử phẩy

Các biểu thức đặt cách nhau bằng dấu ,

Các biểu thức con lần lượt được tính từ trái sang phải.

Biểu thức mới nhận được là giá trị của biểu thức bên phải cùng.

Ví dụ

x = (a++, b = b + 2);

⇔ a++; b = b + 2; x = b;

Toán tử trong ngôn ngữ lập trình C++

Độ ưu tiên của các toán tử

Operator	Priority
() [] -> .	→
! ++ -- - + * (cast) & sizeof	←
* / %	→
+ -	→
<< >>	→
< <= > >=	→
== !=	→
&	→
|	→
^	→
&&	→
||	→
?:	←
= += -= *= /= %= &= ...	←
,	←

Toán tử trong ngôn ngữ lập trình C++

Quy tắc thực hiện

Thực hiện biểu thức trong ( ) sâu nhất trước.

Thực hiện theo thứ tự ưu tiên các toán tử.

=> Tự chủ động thêm ( )

Ví dụ

```
n = 2 + 3 * 5;
=> n = 2 + (3 * 5);
a > 1 && b < 2
=> (a > 1) && (b < 2)
```

Viết biểu thức cho các mệnh đề

- x lớn hơn hay bằng 3

  x >= 3

- a và b cùng dấu

  ((a>0) && (b>0)) || ((a<0) && (b<0))

  (a>0 && b>0) || (a<0 && b<0)

- p bằng q bằng r

  (p == q) && (q == r) hoặc (p == q && q == r)

- –5 < x < 5

  (x > –5) && (x < 5) hoặc (x > –5 && x < 5)

Nhập xuất trong C++

Thư viện

Thư viện vào ra chuẩn trong C++ là iostream

```
#include <iostream>
using namespace std;
```

Nhập giá trị cho biến từ bàn phím

```
cin >> a;
cin >> b >> c >> d;
```

In giá trị của biến ra màn hình

```
cout << a;
cout << b << c << d;
```

Nhập xuất trong C++

Định dạng xuất dữ liệu

Thư viện định dạng xuất dữ liệu

```
#include <iomanip>
```

Ví dụ

- Căn trái, độ rộng 5 ký tự cho biến a

```
cout << setw(5) << left << a;
```

Định dạng số thập phân

```
cout << setprecision(5) << a;
cout << fixed << setprecision(5) << a;
cout << std::resetiosflags( std::cout.flags() ); //reset
```

Cấu trúc lệnh tuần tự

Dãy các chỉ thị hoặc lời gọi hàm thực hiện một cách tuần tự. Các lệnh có thể là lệnh đơn, lệnh có cấu trúc hoặc lệnh phức hợp. Các ví dụ được nêu ở trên đều được thể hiện bằng cấu trúc lệnh tuần tự.

Cấu trúc lệnh tuần tự

```
cin >> a;
cin >> b;
cout << a;
cout << a + b;
```

Cấu trúc lệnh rẽ nhánh

Bao gồm cấu trúc if..else và switch(). Điểm khác biệt duy nhất giữa hai cấu trúc lệnh này là if..else lựa chọn một trong hai khả năng đúng, sai của biểu thức điều kiện để thực hiện, trái lại switch() lựa chọn khả năng đúng của biểu thức điều kiện trong nhiều khả năng để thực hiện

Cấu trúc lệnh if..else.

```
if (biểu-thức-điều-kiện) {
        <Câu lệnh 1>;
}
else {
        <Câu lệnh 2>;
}
```

Cấu trúc lệnh if..else khuyết.

```
if (biểu-thức-điều-kiện) {
        <Câu lệnh 1>;
}
```

Cấu trúc lệnh switch.

```
switch (biểu-thức) {
        case H1: < câu lệnh 1>; break;
        case H2: < câu lệnh 2>; break;
        . . . . . .;
        case Hn: < câu lệnh n>; break;
        default: < câu lệnh n+1>; break;
}
```

Cấu trúc lệnh lặp

Sử dụng for để thực hiện các thao tác, tính toán lặp đi lặp lại với số lần có thể xác định trước

Sử dụng while, do … while để thực hiện các thao tác, tính toán lặp đi lặp lại chưa xác định số lần

Cấu trúc lệnh for

```
for (Khởi tạo; Biểu thức; Bước) {
    <Câu lệnh>;
}
```

Cấu trúc lệnh while

```
while (biểu-thức) {
    <Câu lệnh>;
}
```

Cấu trúc lệnh do … while

```
do {
    <Câu lệnh>;
} while (biểu-thức);
```