KỸ THUẬT LẬP TRÌNH C/C++

Chương 2:
NGÔN NGỮ LẬP TRÌNH C++

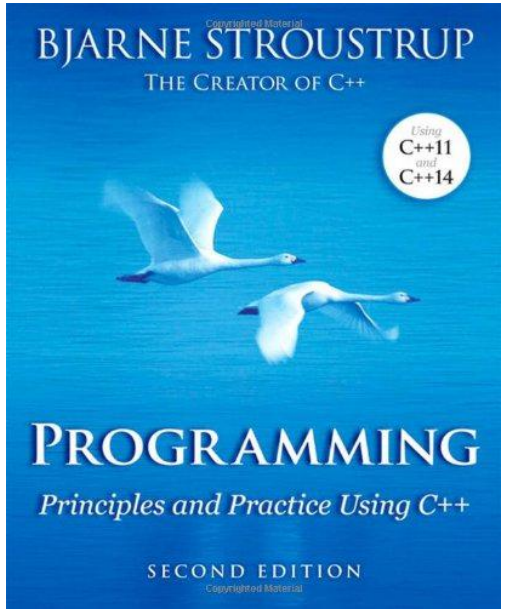
Nội dung

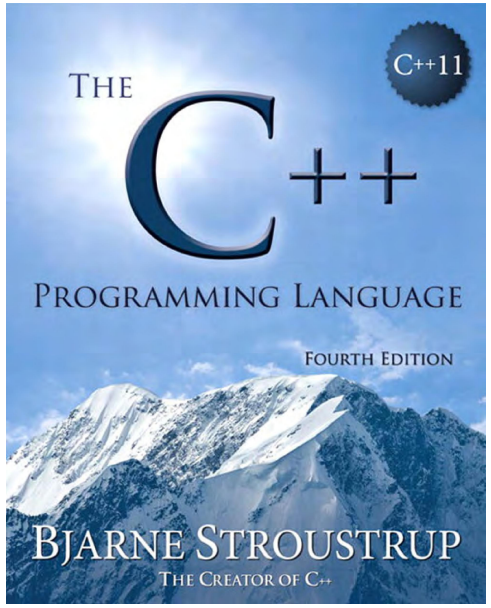
1. Giới thiệu ngôn ngữ lập trình C++
2. Cấu trúc chương trình C++
3. Kiểu dữ liệu, biến và hằng số
4. Phép gán và nhập xuất dữ liệu
5. Ép kiểu dữ liệu
6. Toán tử trong C++
7. Một số hàm toán học thông dụng trong C++

1. Giới thiệu ngôn ngữ lập trình C++

1. Giới thiệu ngôn ngữ lập trình C++

Ngôn ngữ lập trình C++ (C plus plus - Cpp) được phát triển bởi Bjarne Stroustrup vào năm 1979 tại Bell Labs. Từ thập niên 1990, C++ đã trở thành một trong những ngôn ngữ lập trình phổ biến nhất trên thế giới và được ứng dụng rộng rãi trong ngành công nghệ thông tin. C++ được sử dụng để tạo các chương trình máy tính, phát triển phần mềm máy tính, phát triển trò chơi điện tử, ...





1. Giới thiệu ngôn ngữ lập trình C++

❖ Một số đặc điểm của ngôn ngữ lập trình C++

- Đa năng: C++ có thể được dùng để lập trình nhúng, lập trình hệ thống, lập trình ứng dụng, lập trình game, ...
- Hiệu năng cao: chương trình được viết bằng C++ sẽ cho tốc độ thực thi nhanh hơn các chương trình được viết bởi các ngôn ngữ lập trình khác như Java, C#, Python, ... Vì thế với các ứng dụng nặng, cần có tốc độ xử lý nhanh hay các game 3D thường được viết bằng C++.
- Đa nền tảng: chương trình được viết bằng C++ có thể chạy được trên nhiều nền tảng khác nhau như Windows, Mac OS, Linux, ...
- Cộng đồng lập trình lớn: C++ là một trong những ngôn ngữ phổ biến nhất thế giới nên có cộng đồng lập trình viên lớn, bạn có thể dễ dàng tìm kiếm các tài liệu, các lỗi gặp phải khi lập trình trên mạng.
- Bộ thư viện hỗ trợ mạnh mẽ: C++ có bộ thư viện chuẩn và bộ thư viện của bên thứ 3 với nhiều cấu trúc dữ liệu, thuật toán, ... để giúp bạn dễ dàng phát triển chương trình một cách nhanh chóng.
- Hỗ trợ lập trình hướng đối tượng: C++ cho phép bạn lập trình theo phương pháp hướng đối tượng, giúp cho chương trình dễ phát triển và bảo trì hơn.

1. Giới thiệu ngôn ngữ lập trình C++

❖ Các phiên bản của ngôn ngữ lập trình C++

Ngôn ngữ C++ đã được tiêu chuẩn hóa vào năm 1998. Ngôn ngữ này đã được cập nhật các phiên bản vào các năm 2011, 2014, 2017 và 2020.

Năm	Tiêu chuẩn C++	Tên gọi
2011	ISO/IEC 14882:2011	C++11
2014	ISO/IEC 14882:2014	C++14
2017	ISO/IEC 14882:2017	C++17
2020	ISO/IEC 14882:2020	C++20

1. Giới thiệu ngôn ngữ lập trình C++

❖ Ứng dụng của ngôn ngữ lập trình C++

- Phát triển hệ điều hành: C++ được dùng trong việc phát triển các hệ điều hành như Windows, Linux, Mac OS, ...

- Lập trình game: hầu hết các game nổi tiếng hiện nay đều được viết bằng C++ hoặc các Game engine dựa trên C++. Ví dụ như các game Counter Strike, Warcraft III, Doom III, ...

- Lập trình ứng dụng: đây là một trong những mảng mạnh nhất của C++. Có rất nhiều ứng dụng lớn được tạo ra bởi C++ như: Word, Excel, Powerpoint, Google Chrome, Firefox, Adobe Photoshop & Illustrator, …

- Lập trình nhúng: C++ cũng được sử dụng nhiều trong các thiết bị như đồng hồ thông minh, thiết bị y tế, …

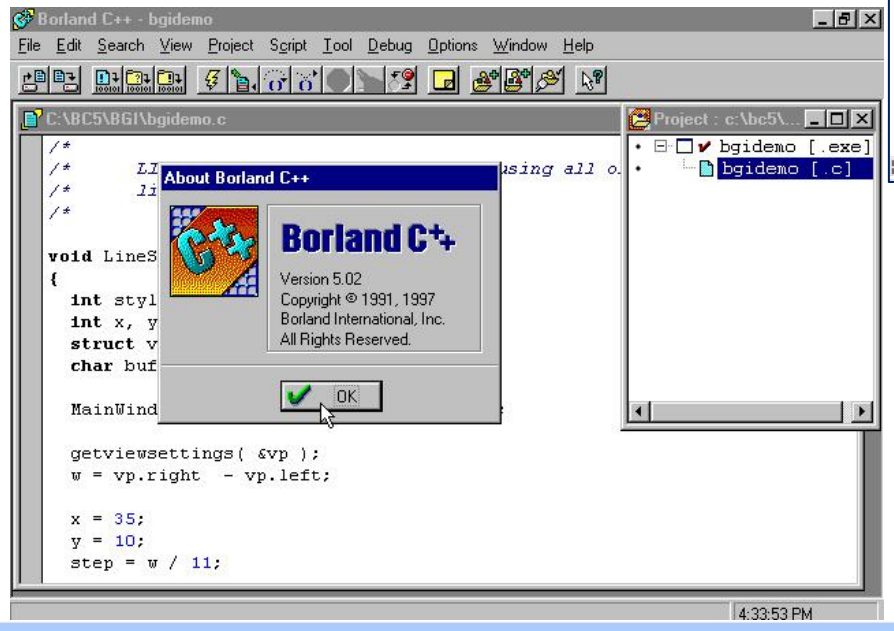
- Ngoài ra C++ còn được dùng để tạo ra các tình biên dịch, các hệ quản trị cơ sở dữ liệu, ...

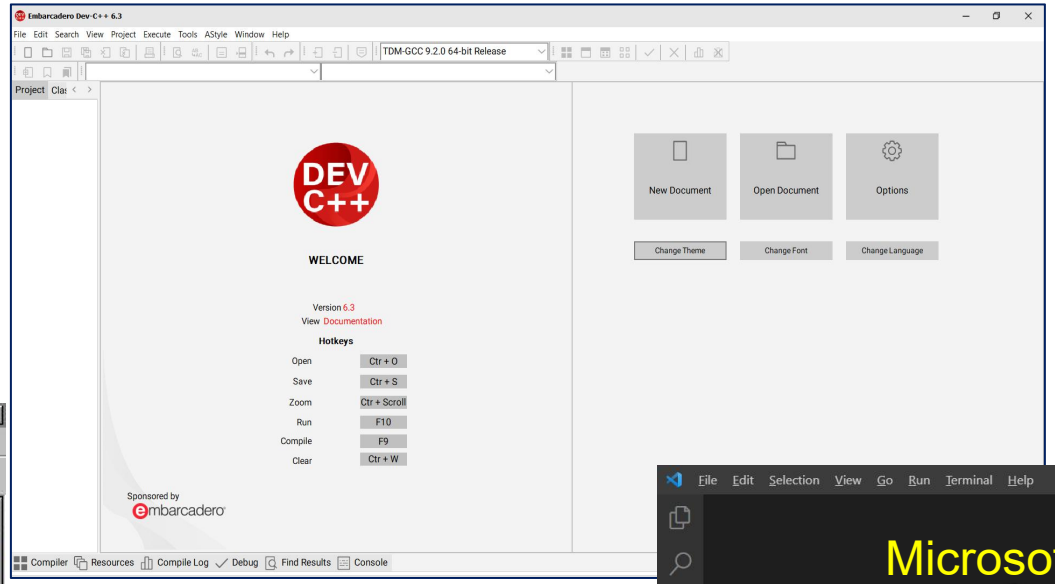
1. Giới thiệu ngôn ngữ lập trình C++

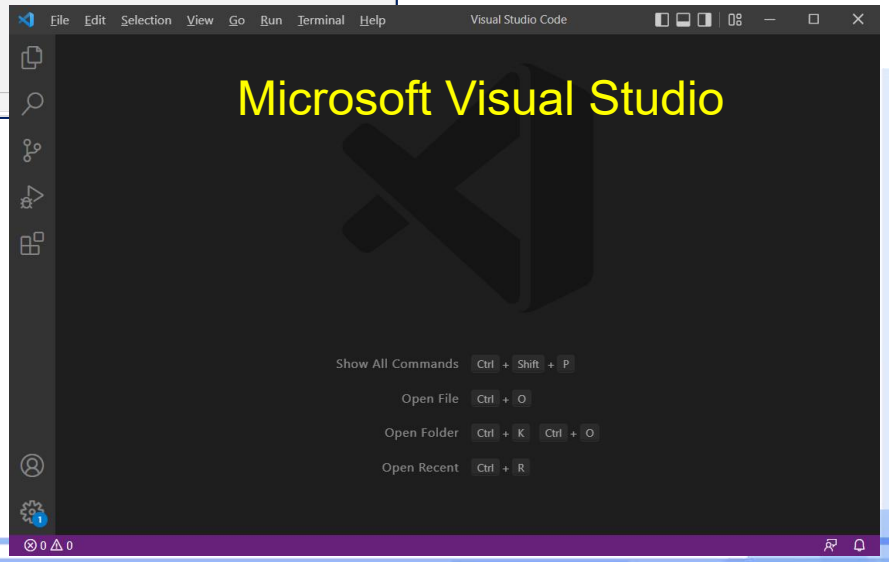
❖ Môi trường phát triển C++

Các môi trường phát triển tích hợp IDE (Integrated Development Environment) để lập trình C++ bao gồm:

- Borland C++
- Dev-C++
- Microsoft Visual Studio







2. Cấu trúc chương trình C++

2. Cấu trúc chương trình C++

❖ **Cấu trúc của một chương trình C++ gồm hai phần như sau:**

```
#include <iostream>
using namespace std;
int main()
{
  //Các câu lệnh C++
}
```

Chỉ thị tiền xử lý: khao báo thư viện và các biến toàn cục sẽ được sử dụng trong chương trình nhằm đưa ra các hướng dẫn tiền xử lý cho trình biên dịch. Chỉ thị bao hàm tệp (#include), chỉ thị định nghĩa cho tên (#define), chỉ thị biên dịch có điều kiện (#undef, #ifdef, #ifndef, #if, ...)

Hàm chính (main): Cho biết điểm bắt đầu thực thi của chương trình, trong đó chứa các lệnh được lập trình bằng ngôn ngữ C++. Mỗi hàm main phải bắt đầu bằng một dấu ngoặc nhọn { và kết thúc bằng dấu ngoặc nhọn }. Mỗi chương trình C++ có duy nhất một hàm main và bắt buộc phải có hàm main. Hàm main có thể không có kiểu trả về (void) hoặc có kiểu trả về (int). Ngoài hàm main, chương trình C++ còn có thể có các hàm khác do người lập trình tự định nghĩa.

2. Cấu trúc chương trình C++

- Chỉ thị tiền xử lý #include: được sử dụng để thêm vào file nguồn (file.c hoặc file.cpp) các nội dung từ những file.h của các thư viện chuẩn trong C/C++ (stdio.h, stdlib.h, string.h, math.h, iostream) hoặc từ những file.h do người dùng tự tạo.

Để thêm nội dung những file.h có trong thư mục cài đặt IDE (thư viện chuẩn), thi sử dụng dấu ngoặc nhọn < >

Cú pháp: #include <file.h hoặc tên thư viện >

Ví dụ: #include <string.h> ; #include <iostream>

Để thêm nội dung những file.h có trong thư mục chứa project, thì cần sử dụng dấu ngoặc kép:

Cú pháp: #include "myfile.h"

- Chỉ thị tiền xử lý #define: được sử dụng để khai báo một hằng số (constant), nó cho phép đưa giá trị số vào tên hằng (được đặt tương tự như tên biến, thường được viết in hoa).

Cú pháp: #define TÊN HẰNG giá trị

Ví dụ:

#define CHIEUDAI_HCN 70

#define CHIEURONG_HCN 50

2. Cấu trúc chương trình C++

Ví dụ 1: Chương trình C++ đơn giản như sau:

```
#include <iostream>
using namespace std;
int main()
{
    //Write C++ code here
    cout << "Hello World!";
    return 0;
}
```

Khai báo thư viện vào ra chuẩn iostream.

Cho phép sử dụng tên cho các đối tượng và biến từ thư viện chuẩn.

Hàm main(), các lệnh bên trong dấu ngoặc nhọn { } của nó sẽ được thực thi.

Bắt đầu chương trình (Begin)

Dòng chú thích.

Lệnh cout là một đối tượng được sử dụng cùng với toán tử << để xuất/in văn bản ra màn hình.

Kết thúc hàm main() và trả về mã đi sau nó, trong trường hợp này là 0.

Kết thúc chương trình (End)

➢ Kết quả chạy chương trình:

In ra màn hình dòng: Hello World!

Lưu ý:
- Mọi câu lệnh C ++ đều kết thúc bằng dấu chấm phẩy ";"
- Trình biên dịch sẽ bỏ qua dòng chú thích và khoảng trắng.

2. Cấu trúc chương trình C++

❖ Khai báo sử dụng thư viện

- Phần khai báo sử dụng thư viện:

Cú pháp:

```
#include <tên thư viện>
Hoặc
#include “tên thư viện”
```

Ví dụ:

```
#include <iostream>
Hoặc
#include "iostream"
```

❖ Khai báo biến và hằng số

- Khai báo biến:

Cú pháp:

```
Kiểu_dữ_liệu  Danh_sách_tên_biến;
```

Ví dụ:

```
int  a,b;
```

- Khai báo (định nghĩa) hằng số:

Cú pháp:

```
#define  Tên_hằng  Giá_trị
```

Ví dụ:

```
#define  number  19
```

2. Cấu trúc chương trình C++

❖ Hàm main()

Hàm main() hay còn gọi là chương trình chính. Các câu lệnh bên trong hàm main() luôn được thực hiện đầu tiên khi chương trình bắt đầu thực thi. Tất cả chương trình C++ đều phải tồn tại một hàm main ().

Cú pháp: int main () : có kiểu trả về hoặc void main() : không có kiểu trả về.

```
int main() {
    //Các câu lệnh C++ (code C++)
    return 0;
}
```

Trong hàm main() có các câu lệnh C++ (statement):

- Mỗi câu lệnh sẽ là một yêu cầu Compiler thực hiện một nhiệm vụ nhất định.
- Mỗi câu lệnh được kết thúc bằng dấu ; để phân biệt giữa các lệnh với nhau.

2. Cấu trúc chương trình C++

❖ Chú thích trong C++

Chú thích (Comment) được dùng để giải thích các dòng lệnh trong chương trình (code), giúp cho người đọc hay chính người viết code sau này hiểu được dễ dàng hơn. Chú thích sẽ không được chương trình thực thi khi chạy chương trình. Trong C++ có 2 loại chú thích: chú thích trên một dòng và chú thích trên nhiều dòng.

- Chú thích trên một dòng sẽ được bắt đầu với dấu: // nội dung cần chú thích

  Ví dụ 2: cout << "Hello world!"; // in ra màn hình "Hello world!"

- Chú thích trên nhiều dòng: bắt đầu bằng /* nội dung chú thích và kết thúc bằng */.

  Ví dụ 3: / * Đoạn mã dưới đây sẽ in dòng chữ Hello World!

  ra màn hình * /

  cout << "Hello world!";

2. Cấu trúc chương trình C++

❖ **Một số lưu ý trong lập trình C++**

➢ **Bộ ký tự:** Bộ chữ viết trong ngôn ngữ C++ bao gồm những ký tự, ký hiệu sau:

- Bộ chữ cái 26 ký tự Latinh A, B, C, …, Z, a, b, c, …, z.
- Phân biệt chữ hoa và chữ thường.
- Bộ chữ số thập phân : 0, 1, 2, …, 9
- Các ký hiệu toán học : + – * / = < > ( )
- Các ký tự đặc biệt : . , : ; [ ] % \ # $ ' “ @ { }
- Ký tự gạch nối _ và khoảng trắng ‘ ’

2. Cấu trúc chương trình C++

❖ Một số lưu ý trong lập trình C++

➢ **Từ khoá:** Từ khóa là các từ dành riêng (reserved words) của C++ với mục đích đã được xác định trước. Không được đặt tên biến, hằng, tên hàm, ... trùng với từ khoá.

Danh sách các từ khóa:

C++ Keywords					
alignas	alignof	and	and_eq	asm	auto
bitand	bitor	bool	break	case	catch
char	char16_t	char32_t	class	compl	const
constexpr	const_cast	continue	decltype	default	delete
do	double	dynamic_cast	else	enum	explicit
extern	false	float	for	friend	goto
if	inline	int	long	mutable	namespace
new	noexcept	not	not_eq	nullptr	operator
or	or_eq	private	protected	public	register
reinterpret_cast	return	short	signed	sizeof	static
static_assert	static_cast	struct	switch	template	this
thread_local	throw	true	try	typedef	typeid
typename	union	unsigned	using	virtual	void
volatile	wchar_t	while	xor	xor_eq	

3. Kiểu dữ liệu, biến và hằng số

3. Kiểu dữ liệu, biến và hằng số

❖ **Kiểu dữ liệu:** Kiểu dữ liệu dùng để xác định loại dữ liệu mà một biến có thể lưu trữ như số nguyên, số thực, ký tự, ... Dựa trên kiểu dữ liệu của một biến, hệ thống sẽ cấp phát bộ nhớ để lưu dữ liệu. Kiểu dữ liệu có sẵn trong C++ còn gọi là kiểu dữ liệu cơ sở hay kiểu dữ liệu cơ bản.

Kiểu dữ liệu	**Từ khóa**
Boolean	bool
Ký tự	char
Số nguyên	int
Số thực	float
Số thực dạng Double	double
Kiểu không có giá trị	void
Kiểu Wide character	wchar_t

- Một số kiểu cơ bản có thể được sửa đổi bởi sử dụng một hoặc nhiều modifier sau:
  - signed (kiểu có dấu)
  - unsigned (kiểu không có dấu)
  - short
  - long
- Kích thước bộ nhớ của các loại dữ liệu cơ bản có thể thay đổi theo hệ điều hành 32 hoặc 64 bit.

3. Kiểu dữ liệu, biến và hằng số

❖ **Kiểu dữ liệu:** Các kiểu dữ liệu cơ bản trong C++ theo kiến trúc 32 bit.

STT	Kiểu dữ liệu	Kích thước	Phạm vi giá trị	
1	char	1 byte	Từ -128 đến 127	**Kiểu ký tự**
2	unsigned char	1 byte	Từ 0 đến 255 (256 ký tự trong bảng mã ASCII)	
3	wchar_t	2 bytes	Từ 0 đến 65.535 (wide character)	
4	string	32 bytes	Chuỗi ký tự	
5	int	2 bytes/4 bytes	Từ -2.147.483.648 đến 2.147.483.647	**Kiểu số nguyên**
6	unsigned int	2 bytes/4 bytes	Từ 0 đến 4,294,967,296	
7	short int	2 bytes	Từ -32.768 đến 32.767	
8	long	4 bytes	Từ -2.147.483.648 đến 2.147.483.647	
9	long long	8 bytes	Từ -2^{63} đến $2^{63} - 1$	
10	unsigned long	4 bytes	Từ 0 đến 4.294.967.295	
11	float	4 bytes	Từ $3,4 * 10^{-38}$ đến $3,4 * 10^{38}$	**Kiểu số thực**
12	double	8 bytes	Từ $1,7 * 10^{-308}$ đến $1,7 * 10^{308}$	
13	long double	10 bytes	Từ $3,4 * 10^{-4932}$ đến $1,1 * 10^{4932}$	
14	bool	1 byte	True hoặc false (1 hoặc 0)	**Luận lý (Boolean)**

3. Kiểu dữ liệu, biến và hằng số

- Xem kích cỡ kiểu dữ liệu cơ bản trong C++.

```
#include <iostream>
using namespace std;
 int main() {
   cout << "Kich thuoc cua char la: " << sizeof(char) <<" byte" << endl;
   cout << "Kich thuoc cua int la: " << sizeof(int) <<" byte" << endl;
   cout << "Kich thuoc cua short int la: " << sizeof(short int) <<" byte" << endl;
   cout << "Kich thuoc cua long int la: " << sizeof(long int) <<" byte" << endl;
   cout << "Kich thuoc cua float la: " << sizeof(float) <<" byte" << endl;
   cout << "Kich thuoc cua double la: " << sizeof(double) <<" byte" << endl;
   cout << "Kich thuoc cua wchar_t la: " << sizeof(wchar_t) <<" byte" << endl;
   return 0;
}
```

=> Kết quả:

```
Kich thuoc cua char la: 1 byte
Kich thuoc cua int la: 4 byte
Kich thuoc cua short int la: 2 byte
Kich thuoc cua long int la: 4 byte
Kich thuoc cua float la: 4 byte
Kich thuoc cua double la: 8 byte
Kich thuoc cua wchar_t la: 2 byte
```

3. Kiểu dữ liệu, biến và hằng số

❖ **Biến:** Biến (variable) đại diện cho vùng nhớ lưu trữ dữ liệu (trên RAM), giá trị dữ liệu này có thể thay đổi trong quá trình chạy chương trình. Mỗi biến phải thuộc về một kiểu dữ liệu xác định và có phạm vi giá trị thuộc kiểu đó.

- Cú pháp khai báo biến:

<Kiểu dữ liệu> <Danh sách tên biến>;

- Khi khai báo biến cần lưu ý:

+ Xác định kiểu dữ liệu cho biến

+ Cách đặt tên biến:

. Không được đặt trùng tên với từ khoá

. Ký tự đứng đầu của tên biến thường là chữ (a, A, b, B, …) hay gạch dưới "_"

. Các ký tự tiếp theo gồm: chữ, số, gạch dưới và không có khoảng trắng

. Không có ký tự đặc biệt: !,@,#,$,%,^,&,*,(,),…

3. Kiểu dữ liệu, biến và hằng số

❖ **Biến (tt):** Ví dụ 4: Khai các báo biến như sau:

```
int a;  //biến a có kiểu dữ liệu là số nguyên (int)
float b; //biến b có kiểu dữ liệu là số thực (float)
char c; //biến c có kiểu dữ liệu là ký tự (char)
```

Các biến có thể được khởi tạo một giá trị khi khai báo biến hoặc gán giá trị sau khi khai báo biến:

```
int a = 15, b = 20; //Khai báo hai biến kiểu số nguyên và gán giá trị ban đầu cho hai biến này.
float c = 17.9; //Khai báo biến kiểu số thực và gán giá trị cho biến.
char d = 'A'; //Khai báo biến kiểu ký tự và gán ký tự cho biến.
```

Khi khai báo biến mà không khởi tạo giá trị cho biến, thì biến sẽ nhận các giá trị mặc định được quy định bởi số ít trình biên dịch C++. Đối với Dev-C++ có khởi tạo giá trị mặc định cho biến.

- Các biến kiểu số được khởi tạo giá trị mặc định là 0.
- Các biến kiểu ký tự có giá trị khởi tạo là null ('\0').

3. Kiểu dữ liệu, biến và hằng số

❖ **Biến (tt):** Một số ví dụ về khai báo biến.

Ví dụ 5:

```
#include <iostream>
using namespace std;
int main() {
    int  a = 150;
    long b = 520;
    float c = 3.50;
    double d = 4.325796;
    char  kytu = 'R';
    bool  dung = true;
    cout << a << endl;
    cout << b << endl;
    cout << c << endl;
    cout << d << endl;
    cout << kytu << endl;
    cout << dung << endl;
    return 0;
}
```

Kết quả:

```
150
520
3.5
4.3258
R
1
```

3. Kiểu dữ liệu, biến và hằng số

❖ **Biến (tt):** Một số ví dụ về khai báo biến và sử dụng biến.

Ví dụ 6 (a):

```
#include <iostream>
using namespace std;
int main() {
    int  value = 17;
    cout << value << endl;
    value = 19;
    cout << value << endl;
    return 0;
}
```

➢ Kết quả: 17
19

Ví dụ 6 (b):

```
#include <iostream>
using namespace std;
int main() {
    int x = 7;
    int y = 5;
    int sum = x + y;
    cout << sum;
    return 0;
}
```

=> Có thể viết lại 3 câu lệnh này như sau:

```
int x = 7, y = 5, sum;
sum = x + y;
```

➢ Kết quả: 12

3. Kiểu dữ liệu, biến và hằng số

❖ **Biến (tt):** Một số ví dụ về khai báo biến và sử dụng biến.

Ví dụ 7:

```
#include <iostream>
using namespace std;
int main() {
    int x = 5, y = 7, z = 30;
    cout << x + y + z;
    return 0;
}
```

➢ Kết quả: in ra màn hình là 42

Ví dụ 8:

```
#include <iostream>
using namespace std;
int main() {
    int x, y, z;
    x = y = z = 30;
    cout << x + y + z;
    return 0;
}
```

➢ Kết quả: in ra màn hình là 90

3. Kiểu dữ liệu, biến và hằng số

❖ **Biến (tt):** Vị trí khai báo biến hay còn gọi là phạm vi của biến. Đó là vùng chương trình mà một biến tồn tại và sử dụng được. Có 2 cách đặt vị trí của biến như sau:

- Biến toàn cục (global): Các biến này được đặt bên ngoài tất cả các hàm và được sử dụng trong toàn chương trình.

- Biến cục bộ (local): Các biến này được đặt ở bên trong hàm hay khối lệnh, nó chỉ có tác dụng bên trong hàm hay khối lệnh chứa nó.

Ví dụ 9:

```
#include <iostream>
using namespace std;
    int x = 9;  ⟶ Biến toàn cục
    int main() {
        int x = 7, y = 5, sum;  ⟶ Biến cục bộ
        sum = x + y;
        cout << sum;
        return 0;
    }
```

➢ Kết quả chạy chương trình: 12

Nếu sửa code lại như sau:

```
#include <iostream>
using namespace std;
    int x = 9;  ⟶ Biến toàn cục
    int main() {
        int  y = 5, sum;  ⟶ Biến cục bộ
        sum = x + y;
        cout << sum;
        return 0;
    }
```

➢ Kết quả chạy chương trình: 14

3. Kiểu dữ liệu, biến và hằng số

❖ **Biến (tt): Biến static (tĩnh):** Biến tĩnh cho phép giữ lại giá trị của nó sau nhiều lần gọi hàm.

Để khai báo biến tĩnh thì thêm từ khóa static phía trước tên biến.

Ví dụ:

```
#include <iostream>
using namespace std;
void function1() {
    int x = 10;  // Biến cục bộ
    static int y = 10; // Biến tĩnh
    x = x + 1;
    y = y + 1;
    cout << "x = " << x << ", y = " << y <<endl;
}

int main() {
    function1();
    function1();
    function1();
    return 0;
}
```

Kết quả:

x = 11, y = 11
x = 11, y = 12
x = 11, y = 13

Giải thích kết quả:

Hàm function1() được gọi 3 lần, biến cục bộ x sẽ in cùng một giá trị là 11 cho mỗi lần gọi hàm. Nhưng biến tĩnh sẽ in giá trị được tăng lên 1 trong mỗi lần gọi hàm, giá trị đó là 11, 12, 13.

3. Kiểu dữ liệu, biến và hằng số

- **Hằng số:** Hằng số (constant) là đại lượng không đổi trong suốt quá trình thực thi chương trình. Chúng thường được định nghĩa trong vùng toàn cục và được đặt trước bởi từ khóa const hoặc thông qua cú pháp #define

  Cú pháp khai báo hằng:

  #define <Tên hằng số> <Giá trị>

  Hoặc

  const <Kiểu dữ liệu> <Tên hằng số> = <Giá trị>;

Ví dụ 10: Khai báo hằng số PI như sau:

```
#define PI 3.14
```

hoặc

```
const float PI = 3.14;
```

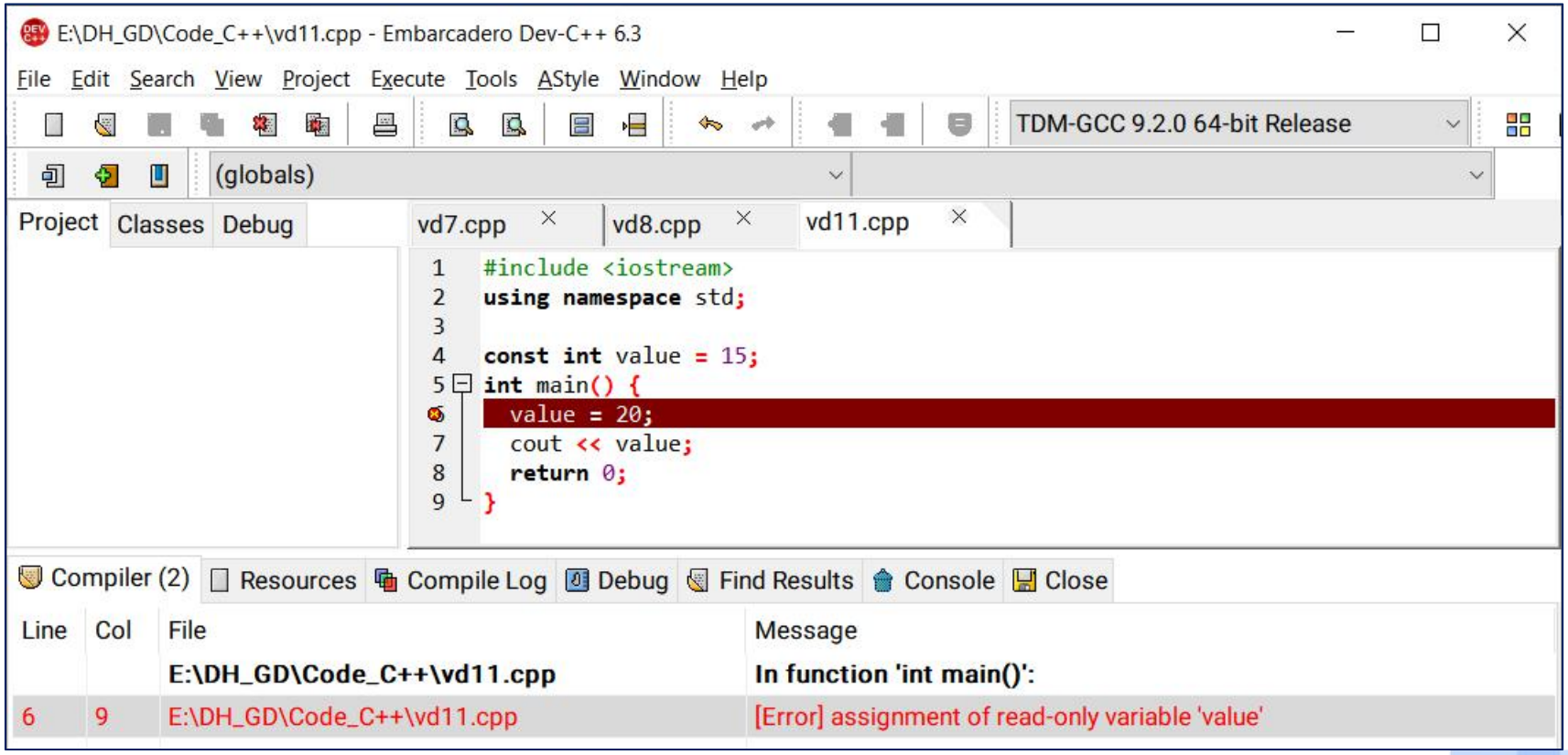
3. Kiểu dữ liệu, biến và hằng số

❖ **Hằng số (tt):** Ví dụ về khai báo hằng số

Ví dụ 11:

```
#include <iostream>
using namespace std;
const int value = 15;
int main() {
  value = 20;
  cout << value;
  return 0;
}
```

Khi biên dịch chương trình này sẽ thông báo lỗi, vì biến value đã được khai báo là hằng số (15) mà trong hàm main() thay đổi giá trị của biến này (20).



3. Kiểu dữ liệu, biến và hằng số

- **Hằng số (tt):** Một số kí tự trong C++ khi được đứng trước bởi dấu \ thì chúng sẽ mang một ý nghĩa đặc biệt được gọi là hằng ký tự. Một số hằng ký tự phổ biến:

Hằng ký tự	Ý nghĩa khi in ra màn hình
\'	Ký tự '
\"	Ký tự "
\?	Ký tự ?
\b	Backspace - xoá lùi một ký tự
\n	Newline - chèn một ký tự xuống dòng
\r	Carriage return - xuống dòng
\t	Tab - chèn một khoảng tab ngang

4. Phép gán và nhập xuất dữ liệu

4. Phép gán và nhập xuất dữ liệu

- **Phép gán:** Phép gán dùng để gán giá trị cho biến, giá trị này sẽ được lưu vào vùng nhớ đại diện bởi tên biến.

  Cú pháp của phép gán:

  <Tên biến> = <giá trị>;

  <Tên biến> = <biến>;

  <Tên biến> = <biểu thức>;

  Ví dụ 12:

```
int a, b, c; //khai báo 3 biến a, b, c có kiểu dữ liệu là int
a = 15; //gán giá trị 15 vào biến a, giá trị 15 được lưu vào vùng nhớ có tên là a
b = 9; //gán giá trị 9 vào biến b, giá trị 9 được lưu vào vùng nhớ có tên là b
c = a + b; //vế phải được tính là 24, giá trị 24 được lưu vào vùng nhớ c
```

4. Phép gán và nhập xuất dữ liệu

❖ **Xuất dữ liệu:** Lệnh cout được sử dụng kết hợp với toán tử chèn luồng (insertion operator), ký hiệu là <<, để thực hiện lệnh xuất dữ liệu ra màn hình.

Cú pháp lệnh cout:

cout << Nội_dung_xuất;

hoặc

cout << Nội_dung_xuất1 << Nội_dung_xuất2 << …;

hoặc

cout << Nội_dung_xuất1;

cout << Nội_dung_xuất2;

Trong đó, Nội_dung_xuất có thể là:
- Thông báo: nội dung thông báo phải đặt trong dấu nháy kép “ ”
- Biến, biểu thức: giá trị của biến hoặc biểu thức
- endl hoặc \n: chèn một ký tự xuống dòng
- \t: chèn một khoảng tab ngang

4. Phép gán và nhập xuất dữ liệu

❖ **Xuất dữ liệu (tt):**

Ví dụ 13 (a):

```
#include <iostream>
using namespace std;
int main() {
  cout << "Hello World!";
  cout << "I am learning C++";
  return 0;
}
```

➢ Kết quả:

```
Hello World!I am learning C++
```

```
#include <iostream>
using namespace std;
int main() {
  cout << "Hello World!\n";
  cout << "I am learning C++";
  return 0;
}
```

➢ Kết quả:

```
Hello World!
I am learning C++
```

```
#include <iostream>
using namespace std;
int main() {
  cout << "Hello World!" << endl;
  cout << "I am learning C++";
  return 0;
}
```

➢ Kết quả:

```
Hello World!
I am learning C++
```

4. Phép gán và nhập xuất dữ liệu

❖ **Xuất dữ liệu (tt):**

Ví dụ 13 (b1):

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
	int so1 = 10;
	int so2 = 11;
	int so3 = 12;
	cout << "Cac so nguyen la: " << so1 << "," << so2 << "," << so3 << endl;
	return 0;
}
```

➢ Kết quả: Cac so nguyen la: 10,11,12

4. Phép gán và nhập xuất dữ liệu

❖ Xuất dữ liệu (tt):

Ví dụ 13 (b2):

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
	int so1 = 10;
	int so2 = 11;
	int so3 = 12;
	cout << "So nguyen thu nhat la: " << so1 << endl;
	cout << "So nguyen thu hai la: " << so2 << endl;
	cout << "So nguyen thu ba la: " << so3 << endl;
	return 0;
}
```

➢ Kết quả:

So nguyen thu nhat la: 10

So nguyen thu hai la: 11

So nguyen thu ba la: 12

4. Phép gán và nhập xuất dữ liệu

❖ **Nhập dữ liệu:** Lệnh cin kết hợp với toán tử trích luồng (extraction operator), ký hiệu là >>, để thực hiện lệnh nhập dữ liệu từ bàn phím trong lúc chạy chương trình.

Cú pháp lệnh cin: cin >> Tên_biến; hoặc cin >> Tên_biến1 >> Tên_biến2 >> …;

Ví dụ 14 (a):

```
#include <iostream>
using namespace std;
int main() {
    int tuoi;
    cout << "Nhap vao tuoi: ";
    cin >> tuoi;
    cout << "Tuoi cua ban la: " << tuoi;
    return 0;
}
```

Ví dụ 14 (b):

```
#include <iostream>
using namespace std;
int main() {
  int a, b;
  cout << "Nhap vao hai so nguyen: ";
  cin >> a >> b;
  cout << "So nguyen thu nhat la: " << a << endl;
  cout << "So nguyen thu hai la: " << b << endl;
  return 0;
}
```

4. Phép gán và nhập xuất dữ liệu

❖ Nhập dữ liệu (tt):

Ví dụ 15 (a):

```
#include <iostream>
using namespace std;
int main() {
    int a, b, tong;
    cout << "Nhap so nguyen thu nhat: ";
    cin >> a;
    cout << "Nhap so nguyen thu hai: ";
    cin >> b;
    tong = a + b;
    cout << "Tong hai so nguyen la: " << tong;
    return 0;
}
```

Ví dụ 15 (b):

```
#include <iostream>
using namespace std;
int main() {
    float a, b, tong;
    cout << "Nhap so thuc thu nhat: ";
    cin >> a;
    cout << "Nhap so thuc thu hai: ";
    cin >> b;
    tong = a + b;
    cout << "Tong hai so thuc la: " << tong;
    return 0;
}
```

5. Ép kiểu dữ liệu

5. Ép kiểu dữ liệu

Ép kiểu dữ liệu trong C++ là việc gán giá trị của một biến có kiểu dữ liệu này sang kiểu dữ liệu khác.

Cú pháp: **(kiểu_dữ_liệu) tên_biến; hoặc kiểu_dữ_liệu (tên_biến);**

Ví dụ 16:

```
#include <iostream>
using namespace std;
int main() {
        float a = 25.7;
        int b = (int)a + 1;
        cout << b;
        return 0;
}
```

➢ Kết quả: 26

Có hai loại ép kiểu dữ liệu:

1. Nới rộng (widening): Là quá trình làm tròn số từ kiểu dữ liệu có kích thước nhỏ hơn sang kiểu dữ liệu có kích thước lớn hơn. Kiểu biến đổi này không làm mất thông tin.

2. Thu hẹp (narrowwing): Là quá trình làm tròn số từ kiểu dữ liệu có kích thước lớn hơn sang kiểu dữ liệu có kích thước nhỏ hơn. Kiểu biến đổi này có thể làm mất thông tin.

5. Ép kiểu dữ liệu

1. Nới rộng (widening): Là quá trình làm tròn số từ kiểu dữ liệu có kích thước nhỏ hơn sang kiểu dữ liệu có kích thước lớn hơn. Kiểu biến đổi này không làm mất thông tin. Chuyển kiểu dữ liệu loại này không yêu cầu chỉ định ép kiểu mà nó được thực hiện ngầm định bởi trình biên dịch.

Ví dụ 17 (a):

```
#include <iostream>
using namespace std;
int main() {
    int a = 150;
    long b = a;
    float c = b;
    cout << "Gia tri Int: " << a << endl;
    cout << "Gia tri Long: " << b << endl;
    cout << "Gia tri Float: " << c << endl;
    return 0;
}
```

➢ Kết quả:

```
Gia tri Int: 150
Gia tri Long: 150
Gia tri Float: 150
```

5. Ép kiểu dữ liệu

1. Nới rộng (widening):

Ví dụ 17 (b1):

```
#include<iostream>
using namespace std;
int main() {
    int a = 473529475;
    int b = 977952617;
    cout << "Tich so la: " << a * b;
    return 0;
}
```

➢ Kết quả: Tich so la: -1801015621

Kết quả sai, do kết quả của phép nhân 2 số a và b kiểu int sẽ là 1 số kiểu int mà miền giá trị của kiểu int là không đủ để lưu trữ kết quả của phép nhân 384847522 * 988347273.

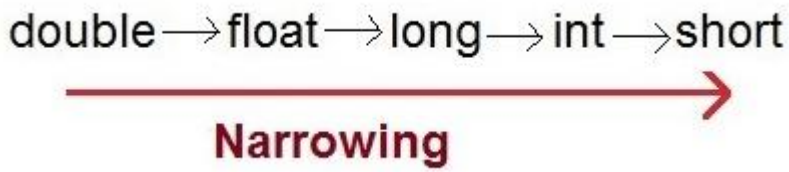
Ví dụ 17 (b2):

```
#include<iostream>
using namespace std;
int main() {
    int a = 473529475;
    int b = 977952617;
    cout << "Tich so la: " << (long long)a * b;
    return 0;
}
```

Kết quả: Tich so la: 463089389302886075

5. Ép kiểu dữ liệu

2. Thu hẹp (narrowwing): Là quá trình làm tròn số từ kiểu dữ liệu có kích thước lớn hơn sang kiểu dữ liệu có kích thước nhỏ hơn. Kiểu biến đổi này có thể làm mất thông tin. Chuyển kiểu dữ liệu loại này thì người dùng phải thực hiện chuyển kiểu dữ liệu tường minh (ép kiểu dữ liệu).



Ví dụ 18:

```
#include <iostream>
using namespace std;
int main() {
    double a = 150.27;
    long b = (long) a;
    int c = (int) b;
    cout << "Gia tri Double: " << a << endl;
    cout << "Gia tri Long: " << b << endl;
    cout << "Gia tri Int: " << c << endl;
    return 0;
}
```

➢ Kết quả:

```
Gia tri Double: 150.27
Gia tri Long: 150
Gia tri Int: 150
```

6. Toán tử trong C++

6. Toán tử trong C++

Toán tử (operator) trong C++ là một ký hiệu được sử dụng để thực hiện một phép tính/chức năng nào đó. Ngôn ngữ lập trình C++ cung cấp các dạng toán tử sau:

- Toán tử số học.
- Toán tử quan hệ.
- Toán tử logic.
- Toán tử trên bit.
- Toán tử gán.
- Biểu thức (expression)

6. Toán tử trong C++

1. Toán tử số học: Các toán tử số học trong C++ cho phép thực hiện các phép tính số học.

Ví dụ 19: Cho hai biến: A = 20 và biến B = 10, áp dụng vào trong các phép toán tử số học.

Toán tử	Miêu tả	Ví dụ
+	Cộng hai toán hạng	A + B sẽ cho kết quả là 30
-	Trừ toán hạng thứ nhất cho toán hạng thứ hai	A - B sẽ cho kết quả là 10
*	Nhân hai toán hạng	A * B sẽ cho kết quả là 200
/	Chia lấy phần nguyên hai toán hạng	A / B sẽ cho kết quả là 2
%	Chia lấy phần dư hai toán hạng (đối với số nguyên)	A % B sẽ cho kết quả là 0
++	Tăng giá trị toán hạng lên 1 đơn vị A = A +1	C = A++ : gán A vào C sau đó tăng A lên 1 => C = 20 C = ++A : tăng A lên 1 sau đó gán vào C => C = 21
--	Giảm giá trị toán hạng đi 1 đơn vị	C = B-- : gán B vào C sau đó giảm B đi 1 => C = 10 C = --B : giảm B đi 1 sau đó gán vào C => C = 9

6. Toán tử trong C++

1. Toán tử số học (tt):

➢ Lưu ý về kiểu dữ liệu khi thực hiện các phép toán tử số học:

Toán hạng thứ nhất	Các toán tử	Toán hạng thứ hai	Kết quả
Số nguyên	+	Số nguyên	Số nguyên
Số thực	-	Số nguyên	Số thực
Số nguyên	*	Số thực	Số thực
Số thực	/	Số thực	Số thực
Ký tự (mã ASCII)		Số nguyên	Số nguyên

```
#include<iostream>
using namespace std;
int main() {
    float a = 5.2;
    int b = 2.7;
    cout << "Ket qua la: " << a + b;
    return 0;
}
```

Ket qua la: 7.2

6. Toán tử trong C++

1. Toán tử số học (tt):

Ví dụ 19 (a):

```
#include <iostream>
using namespace std;
int main() {
        int a, b, tong, hieu, tich, sodu;
        cout << "Nhap so nguyen thu nhat: ";
        cin >> a;
        cout << "Nhap so nguyen thu hai: ";
        cin >> b;
        tong = a + b;
        hieu = a - b;
        tich = a * b;
        sodu = a % b;
        cout << "Tong hai so nguyen la: " << tong << endl;
        cout << "Hieu hai so nguyen la: " << hieu << endl;
        cout << "Tich hai so nguyen la: " << tich << endl;
        cout << "So du la: " << sodu << endl;
        return 0;
}
```

Kết quả:

```
Nhap so nguyen thu nhat: 5
Nhap so nguyen thu hai: 2
Tong hai so nguyen la: 7
Hieu hai so nguyen la: 3
Tich hai so nguyen la: 10
So du la: 1
```

6. Toán tử trong C++

1. Toán tử số học (tt): Ví dụ 19 (b):

```
#include <iostream>
using namespace std;
int main() {
    int a, b;
    float thuong;
    cout << "Nhap so nguyen thu nhat: ";
    cin >> a;
    cout << "Nhap so nguyen thu hai: ";
    cin >> b;
    thuong = (float)a / b;
    cout << "Thuong so la: " << thuong << endl;
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main() {
    int a, b;
    float thuong;
    cout << "Nhap so nguyen thu nhat: ";
    cin >> a;
    cout << "Nhap so nguyen thu hai: ";
    cin >> b;
    thuong = a / (float)b;
    cout << "Thuong so la: " << thuong << endl;
    return 0;
}
```

6. Toán tử trong C++

1. Toán tử số học (tt): Đối với phép toán tử ++ (tăng) và -- (giảm) cần lưu ý:

Tăng trước: C = ++A : tăng A lên 1 sau đó gán vào C.
Tăng sau: C = A++ : gán A vào C sau đó tăng A lên 1.
Giảm trước: C = --B : giảm B đi 1 sau đó gán vào C
Giảm sau: C = B-- : gán B vào C sau đó giảm B đi 1

Ví dụ 19 (c):

```
#include <iostream>
using namespace std;
int main() {
	int a, b, tang, giam;
	cout << "Nhap so nguyen thu nhat: ";
	cin >> a;
	cout << "Nhap so nguyen thu hai: ";
	cin >> b;
	tang = ++a;
	giam = --b;
	cout << "Ket qua tang la: " << tang << endl;
	cout << "Ket qua giam la: " << giam << endl;
	return 0;
}
```

```
#include <iostream>
using namespace std;
int main() {
	int a, b, tang, giam;
	cout << "Nhap so nguyen thu nhat: ";
	cin >> a;
	cout << "Nhap so nguyen thu hai: ";
	cin >> b;
	tang = a++;
	giam = b--;
	cout << "Ket qua tang la: " << tang << endl;
	cout << "Ket qua giam la: " << giam << endl;
	return 0;
}
```

6. Toán tử trong C++

2. Toán tử quan hệ: Các toán tử quan hệ được sử dụng để kiểm tra mối quan hệ giữa hai toán hạng. Kết quả của một biểu thức có dùng các toán tử quan hệ là những giá trị Boolean (logic true = 1 hoặc false = 0). Các toán tử quan hệ được sử dụng trong các cấu trúc điều khiển. Bảng dưới đây mô tả các toán tử quan hệ được hỗ trợ bởi ngôn ngữ C++. Ví dụ 20: cho hai biến A = 20 và B = 10, áp dụng vào toán tử quan hệ.

Toán tử	Miêu tả	Ví dụ
==	Kiểm tra nếu 2 toán hạng có bằng nhau hay không. Nếu bằng thì điều kiện là đúng (true), ngược lại thì điều kiện sai (false).	(A == B) kết quả là false.
!=	Kiểm tra 2 toán hạng có giá trị khác nhau hay không. Nếu khác nhau thì điều kiện là true, ngược lại thì false.	(A != B) kết quả là true.
>	Kiểm tra nếu toán hạng bên trái có giá trị lớn hơn toán hạng bên phải hay không. Nếu lớn hơn thì điều kiện là true, ngược lại thì false.	(A > B) kết quả là true.
<	Kiểm tra nếu toán hạng bên trái có giá trị nhỏ hơn toán hạng bên phải hay không. Nếu nhỏ hơn thì là true, ngược lại thì false.	(A < B) kết quả là false.
>=	Kiểm tra nếu toán hạng bên trái có giá trị lớn hơn hoặc bằng giá trị của toán hạng bên phải hay không. Nếu đúng là true, ngược lại thì false.	(A >= B) kết quả là true.
<=	Kiểm tra nếu toán hạng bên trái có giá trị nhỏ hơn hoặc bằng toán hạng bên phải hay không. Nếu đúng là true, ngược lại thì false.	(A <= B) kết quả là false.

6. Toán tử trong C++

2. Toán tử quan hệ (tt):

Ví dụ 20 (tt)

```
#include <iostream>
using namespace std;
int main() {
	int a, b;
	cout << "Nhap so nguyen thu nhat: ";
	cin >> a;
	cout << "Nhap so nguyen thu hai: ";
	cin >> b;
	cout << "Kiem tra bang nhau?: " << (a == b) << endl;
	cout << "Kiem tra khac nhau?: " << (a != b) << endl;
	cout << "Kiem tra lon hon?: " << (a > b) << endl;
	cout << "Kiem tra nho hon?: " << (a < b) << endl;
	cout << "Kiem tra lon hon hoac bang?: " << (a >= b) << endl;
	cout << "Kiem tra nho hon hoac bang?: " << (a <= b) << endl;
	return 0;
}
```

Kết quả:

```
Nhap so nguyen thu nhat: 20
Nhap so nguyen thu hai: 10
Kiem tra bang nhau?: 0
Kiem tra khac nhau?: 1
Kiem tra lon hon?: 1
Kiem tra nho hon?: 0
Kiem tra lon hon hoac bang?: 1
Kiem tra nho hon hoac bang?: 0
```

- Ghi chú: 1 là True ; 0 là False

6. Toán tử trong C++

3. Toán tử logic: Bảng dưới đây mô tả các toán tử logic được hỗ trợ bởi ngôn ngữ C++.

Ví dụ 21: cho biến A = true (hoặc 1) và biến B = false (hoặc 0):

Toán tử	Miêu tả	Ví dụ
&&	Được gọi là toán tử logic AND (và). Nếu cả hai toán tử đều có giá trị khác 0 thì điều kiện trở lên true.	(A && B) kết quả là false.
||	Được gọi là toán tử logic OR (hoặc). Nếu một trong hai toán tử khác 0, thì điều kiện là true.	(A || B) kết quả là true.
!	Được gọi là toán tử NOT (phủ định). Sử dụng để đảo ngược lại trạng thái logic của toán hạng đó. Nếu điều kiện toán hạng là true thì phủ định nó sẽ là false.	!(A && B) kết quả là true.

6. Toán tử trong C++

3. Toán tử logic (tt):

Ví dụ 21 (tt): cho biến A = true (hoặc 1) và biến B = false (hoặc 0):

```
#include <iostream>
using namespace std;
int main() {
    bool a = true; // a = 1
    bool b = false; // b = 0
    cout << "Toan tu logic AND: " << (a && b) << endl;
    cout << "Toan tu logic OR: " << (a || b) << endl;
    cout << "Toan tu logic NOT: " << !(a && b) << endl;
    return 0;
}
```

Kết quả:

```
Toan tu logic AND: 0
Toan tu logic OR: 1
Toan tu logic NOT: 1
```

6. Toán tử trong C++

4. Toán tử gán: Toán tử gán dùng để gán một giá trị vào một biến và có thể gán nhiều giá trị cho nhiều biến cùng một lúc.

Toán tử	Miêu tả	Ví dụ
=	Gán giá trị toán hạng bên phải cho toán hạng trái của dấu =	C = A + B sẽ gán giá trị của A + B vào C
+=	Cộng giá trị toán hạng phải tới toán hạng trái và gán kết quả này cho toán hạng trái.	C += A tương đương với C = C + A
-=	Trừ giá trị toán hạng phải cho toán hạng trái và gán kết quả này cho toán hạng trái.	C -= A tương đương với C = C - A
*=	Nhân giá trị toán hạng phải với toán hạng trái và gán kết quả này cho toán hạng trái.	C *= A tương đương với C = C * A
/=	Chia toán hạng trái cho toán hạng phải và gán kết quả thương số cho toán hạng trái.	C /= A tương đương với C = C / A
%=	Chia toán hạng trái cho toán hạng phải và gán kết quả dư số cho toán hạng trái.	C %= A tương đương với C = C % A
<<=	Dịch trái toán hạng trái với số bit cần dịch là giá trị toán hạng phải.	C <<= 2 tương đương với C = C << 2
>>=	Dịch phải toán hạng trái với số bit cần dịch là giá trị toán hạng phải.	C >>= 2 tương đương với C = C >> 2
&=	Phép AND bit	C &= 2 tương đương với C = C & 2
^=	Phép OR loại trừ bit	C ^= 2 tương đương với C = C ^ 2
|=	Phép OR bit.	C |= 2 tương đương với C = C | 2

6. Toán tử trong C++

4. Toán tử gán (tt):

Ví dụ 22:

```
#include <iostream>
using namespace std;
int main() {
    int a, b, c;
    cout << "Nhap so nguyen thu nhat: ";
    cin >> a;
    cout << "Nhap so nguyen thu hai: ";
    cin >> b;
    c = a + b;
    cout << "Tong a va b la: " << c << endl;
    c += a;
    cout << "Tong c va a la: " << c << endl;
    c -= a;
    cout << "Hieu c va a la: " << c << endl;
    c *= a;
    cout << "Tich c va a la: " << c << endl;
    return 0;
}
```

Kết quả:

```
Nhap so nguyen thu nhat: 20
Nhap so nguyen thu hai: 10
Tong a va b la: 30
Tong c va a la: 50
Hieu c va a la: 30
Tich c va a la: 600
```

6. Toán tử trong C++

5. Biểu thức (expression): Biểu thức được tạo thành từ toán tử (operator) và toán hạng (operand). Toán hạng có thể là biến, hằng, lời gọi hàm/thủ tục,…

Ví dụ 23:

- Biểu thức: S = X + Y; trong đó X, Y là toán hạng, + là toán tử hai ngôi vì có hai toán hạng và S là biến chứa kết quả.
- Biểu thức: S = ! (A && B); trong đó (A && B) là biểu thức con, ! là toán tử một ngôi vì chỉ cần 1 toán hạng, && là toán tử hai ngôi với hai toán hạng là A và B.

Lưu ý: Toán hạng phải có kiểu dữ liệu tương thích với kiểu mà toán tử có thể thực hiện được.

6. Toán tử trong C++

5. Biểu thức (expression) (tt): Thứ tự ưu tiên của các toán tử trong biểu thức. Trong một biểu thức có thể có nhiều toán tử, câu hỏi đặt ra là toán tử nào sẽ được thực hiện trước? Thứ tự thực hiện của các toán tử có thể ảnh hưởng đến kết quả của biểu thức.

Ví dụ 24: Thứ tự ưu tiên các toán tử trong biểu thức

- Biểu thức: n = 2 + 3 * 5 => n = 2 + (3 * 5) => n = 17
- Biểu thức: a > 1 && b < 2 => (a > 1) && (b < 2)

```
#include <iostream>
using namespace std;
int main() {
	int n;
	n = 2 + 3 * 5; //n = 2 + (3 * 5)
	cout << "Ket qua la: " << n << endl;
	return 0;
}
```

```
Ket qua la: 17
```

6. Toán tử trong C++

5. Biểu thức (expression) (tt):

Mệnh đề cần được chuyển thành biểu thức để đưa vào lập trình.

Ví dụ 25: Viết biểu thức cho các mệnh đề sau

a) Mệnh đề: a lớn hơn hay bằng 5 thì viết biểu thức trong lập trình là: (a >= 5)

b) Mệnh đề: a và b cùng dấu thì viết biểu thức trong lập trình là:

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

c) Mệnh đề: a bằng b bằng c thì viết biểu thức trong lập trình là:

(a == b) && (b == c) hoặc (a == b && b == c)

d) Mệnh đề: a lớn hơn -5 và nhỏ hơn 5 (-5 < a < 5) thì viết biểu thức trong lập trình là:

(a > -5) && (a < 5) hoặc (a > -5 && a < 5)

6. Toán tử trong C++

5. Biểu thức (expression) (tt):

Ví dụ 25 (a): Viết chương trình nhập vào số nguyên từ bàn phím trong lúc chạy chương trình, sau đó gán số nguyên này vào biến a. Kiểm tra a có lớn hơn hay bằng 5 không?

```
#include <iostream>
using namespace std;
int main() {
    int a, kt;
    cout << "Nhap vao so nguyen: ";
    cin >> a;
    kt = (a >= 5);
    cout << "Ket qua kiem tra = " << kt << endl;
    return 0;
}
```

6. Toán tử trong C++

5. Biểu thức (expression) (tt):

Ví dụ 25 (b): Viết chương trình nhập vào 2 số nguyên từ bàn phím trong lúc chạy chương trình, gán 2 số nguyên lần lượt vào 2 biến a và b. Kiểm tra a và b có cùng dấu không?

```
#include <iostream>
using namespace std;
int main() {
    int a, b, kt;
    cout << "Nhap vao so nguyen thu nhat: ";
    cin >> a;
    cout << "Nhap vao so nguyen thu hai: ";
    cin >> b;
    kt = (a>0 && b>0) || (a<0 && b<0);
    cout << "Ket qua kiem tra = " << kt << endl;
    return 0;
}
```

6. Toán tử trong C++

5. Biểu thức (expression) (tt):

Ví dụ 25 (c): Viết chương trình nhập vào 3 số nguyên từ bàn phím trong lúc chạy chương trình, gán 3 số nguyên lần lượt vào 3 biến a, b, c. Kiểm tra 3 số có bằng nhau không?

```
#include <iostream>
using namespace std;
int main() {
    int a, b, c, kt;
    cout << "Nhap vao so nguyen thu nhat: ";
    cin >> a;
    cout << "Nhap vao so nguyen thu hai: ";
    cin >> b;
    cout << "Nhap vao so nguyen thu ba: ";
    cin >> c;
    kt = (a == b) && (b == c);
    cout << "Ket qua kiem tra = " << kt << endl;
    return 0;
}
```

6. Toán tử trong C++

5. Biểu thức (expression) (tt):

Ví dụ 25 (d): Viết chương trình nhập vào số nguyên từ bàn phím trong lúc chạy chương trình, sau đó gán số nguyên này vào biến a. Kiểm tra -5 < a < 5 không?

```
#include <iostream>
using namespace std;
int main() {
    int a, kt;
    cout << "Nhap vao so nguyen: ";
    cin >> a;
    kt = (a > -5) && (a < 5);
    cout << "Ket qua kiem tra = " << kt << endl;
    return 0;
}
```

7. Một số hàm toán học thông dụng trong C++

7. Một số hàm toán học thông dụng trong C++

Ngôn ngữ C++ cung cấp một số hàm toán học cơ bản để thực hiện các phép toán trên các biến số. Để sử dụng các hàm toán học này, cần phải khai báo thư viện <cmath> như sau:

```
#include <cmath>
```

Hàm toán học	**Miêu tả**	**Ví dụ**
min(x,y)	Tìm giá trị nhỏ nhất giữa 2 số nguyên x và y.	min(7,9); kết quả là 7
max(x,y)	Tìm giá trị lớn nhất giữa 2 số nguyên x và y.	max(7,9); kết quả là 9
sqrt(x)	Trả về giá trị căn bậc 2 của x.	sqrt(9); kết quả là 3
round(x)	Trả về số nguyên là giá trị làm tròn lên của x nếu phần thập phân >= 5. Trả về số nguyên là giá trị làm tròn xuống nếu phần thập phân < 5.	round(4.3); kết quả là 4 round (4.7); kết quả là 5
pow(x,y)	Trả về giá trị của x lũy thừa y (hay x^y)	pow(2,3); kết quả là 8
log(x)	Trả về giá trị ln(x) của x.	log(4); kết quả là 1.38629
fmod(x,y)	Trả về phần dư của phép chia số thực x/y.	fmod(19,7); kết quả là 5

7. Một số hàm toán học thông dụng trong C++

Hàm toán học thông dụng trong C++ (tt)

Hàm toán học	Miêu tả	Ví dụ
fmin(x,y)	Tìm giá trị nhỏ nhất giữa 2 số thực x và y.	fmin(4.5, 2.7); kết quả là 2.7
fmax(x,y)	Tìm giá trị lớn nhất giữa 2 số thực x và y.	fmin(4.5, 2.7); kết quả là 4.5
floor(x)	Trả về số nguyên là giá trị làm tròn xuống của x.	floor(5.3); kết quả là 5 floor(5.7); kết quả là 5
abs(x)	Trả về giá trị tuyệt đối của x.	abs(-7); kết quả là 7
cbrt(x)	Trả về giá trị căn bậc ba của x.	cbrt(1000); kết quả là 10
exp(x)	Trả về của e^x.	exp(2); kết quả là 7.38906
fabs(x)	Trả về giá trị tuyệt đối của số thực x.	fabs(-5.3); kết quả là 5.3

7. Một số hàm toán học thông dụng trong C++

Ví dụ 26 (a): Nhập 2 số nguyên từ bàn phím tính giá trị min, max, tính luỹ thừa hai của số thứ nhất và căn bậc hai của số thứ 2.

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
	int a, b;
	cout << "Nhap vao so nguyen thu nhat: ";
	cin >> a;
	cout << "Nhap vao so nguyen thu hai: ";
	cin >> b;

	cout << "Min giua hai so la: " << min(a,b) << endl;
	cout << "Max giua hai so la: " << max(a,b) << endl;
	cout << "Luy thua hai cua so thu nhat la: " << pow(a,2) << endl;
	cout << "Can bac hai cua so thu hai la: " << sqrt(b) << endl;
	return 0;
}
```

Kết quả:

```
Nhap vao so nguyen thu nhat: 4
Nhap vao so nguyen thu hai: 9
Min giua hai so la: 4
Max giua hai so la: 9
Luy thu hai cua so thu nhat la: 16
Can bac hai cua so thu hai la: 3
```

7. Một số hàm toán học thông dụng trong C++

Ví dụ 26 (b): Nhập 2 số nguyên từ bàn phím tính giá trị min, max, tính luỹ thừa hai của số thứ nhất và căn bậc hai của số thứ 2.

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
	int a, b, mi, ma, po, sq;
	cout << "Nhap vao so nguyen thu nhat: ";
	cin >> a;
	cout << "Nhap vao so nguyen thu hai: ";
	cin >> b;
	mi = min(a,b);
	ma = max(a,b);
	po = pow(a,2);
	sq = sqrt(b);
	cout << "Min giua hai so la: " << mi << endl;
	cout << "Max giua hai so la: " << ma << endl;
	cout << "Luy thua hai cua so thu nhat la: " << po << endl;
	cout << "Can bac hai cua so thu hai la: " << sq << endl;
	return 0;
}
```

Kết quả:

```
Nhap vao so nguyen thu nhat: 4
Nhap vao so nguyen thu hai: 9
Min giua hai so la: 4
Max giua hai so la: 9
Luy thu hai cua so thu nhat la: 16
Can bac hai cua so thu hai la: 3
```