KIỂU DỮ LIỆU CƠ BẢN

1. **Kiểu số nguyên**

Các kiểu thông dụng:

- Số nguyên là số không có phần thập phân.
- Biểu diễn số nguyên: kiểu **int** (4 byte), phạm vi biểu diễn từ -2147483648 đến 2147483647
- Biểu diễn kí tự: kiểu **char** (1 byte)
- Biểu diễn số nguyên dương: **unsigned int**, phạm vi biểu diễn từ 0 đến 4294967295.

Các phép toán trên kiểu dữ liệu số nguyên:

- Phép gán.

Ví dụ:

```
#include<iostream>
using namespace std;
int main()
{
	// Kieu int
	int a;
	a = 5;
	int b = 10;
	const int c = 20;
	cout << "a = " << a << ", b = " << b << ", c = " << c << endl;

	// Kieu char
	char c1 = 65; // ki tu 'A' co ma ASCII la 65
	char c2 = 't';
	cout << "c1 = " << c1 << ", c2 = " << c2 << endl;
}
```

Kết quả chạy chương trình:

```
a = 5, b = 10, c = 20
c1 = A, c2 = t
```

- Phép cộng (+), trừ (-) và nhân (*). Ví dụ:

```
#include<iostream>
using namespace std;
int main()
{
	int a, b, c;
	a = 2;
	b = 5;
	c = -4;
	int x = a + b; // Phep cong
	int y = a - b; // Phep tru
	int z = a * b; // Phep nhan
	int t = 9 * c;
	cout << "a + b = " << x << endl;
	cout << "a - b = " << y << endl;
	cout << "a * b = " << z << endl;
	cout << "9 * c = " << t << endl;
}
```

Kết quả chạy chương trình:

```
a + b = 7
a - b = -3
a * b = 10
9 * c = -36
```

- Phép chia nguyên (/) và phép chia lấy phần dư (%). Ví dụ:

```
#include<iostream>
using namespace std;
int main()
{
	int a = 8;
	int b = 5;
	int chia_nguyen = a / b;
	int chia_lay_du = a % b;
	cout << "a / b = " << chia_nguyen << endl;
	cout << "a % b = " << chia_lay_du << endl;
}
```

Kết quả chạy chương trình:

```
a / b = 1
a % b = 3
```

Giải thích: 8 chia 5 được 1 dư 3, nên 8 / 5 = 1 và 8 % 5 = 3.

2. Kiểu số thực

Kiểu dữ liệu số thực thông dụng:

- **float** (4 byte)
- **double** (8 byte)
- Sử dụng để lưu trữ các giá trị số có phần thập phân.

Các phép toán trên kiểu số thực:

- Phép gán. Ví dụ:

```
#include<iostream>
using namespace std;
int main()
{
	float a = 1.5;
	double b = 2.4;
	cout << "a = " << a << endl;
	cout << "b = " << b << endl;

	// kha nang luu tru
	float x = 1234.5678912345;
	double y = 1234.5678912345;
	cout.precision(10);
	cout << fixed << "x = " << x << ", y = " << y << endl;
}
```

Kết quả chạy chương trình:

```
a = 1.5
b = 2.4
x = 1234.5678710938, y = 1234.5678912345
```

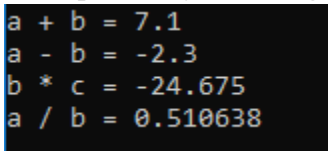
Như vậy, kiểu double có khả năng lưu trữ chính xác cao hơn kiểu float.

- Phép cộng trừ nhân chia (+ , - , *, /). Ví dụ:

```
#include<iostream>
using namespace std;
int main()
{
	float a = 2.4;
	float b = 4.7;
	double c = -5.25;

	float cong = a + b; // Phep cong
	float tru = a - b; // Phep tru
	double nhan = b * c; // Phep nhan
	float chia = a / b; // Phep chia
	cout << "a + b = " << cong << endl;
	cout << "a - b = " << tru << endl;
	cout << "b * c = " << nhan << endl;
	cout << "a / b = " << chia << endl;
}
```

Kết quả chạy chương trình:



3. Kiểu luận lý

- Trong C++ định nghĩa kiểu **bool** với các giá trị là true, false tương ứng với các số nguyên 1 và 0.
- Quy đổi giá trị các kiểu dữ liệu khác sang bool: giá trị số 0 sẽ tương đương với giá trị đúng, và giá trị khác không tương đương với giá trị sai. Ví dụ:

```cpp
#include<iostream>
using namespace std;
int main()
{
	bool a = true;
	bool b = false;
	cout << "a = " << a << ", b = " << b << endl;

	bool x = 10;
	bool y = 0;
	cout << "x = " << x << ", y = " << y << endl;
}
```

Kết quả chạy chương trình:

```
a = 1, b = 0
x = 1, y = 0
```

Kết quả in ra tương ứng với true là 1, và false là 0.

Các phép toán thông dụng trên kiểu luận lý:

- Phép phủ định (!), phép và (&&), phép hoặc (||).
  Ví dụ:

```
#include<iostream>
using namespace std;
int main()
{
	int a = 5;
	int b = 7;
	bool x = !(a > b); // true
	bool y = (a > b) && (b < 5); // false
	bool z = (a < b) || (b > 10); // true
	cout << "x = " << x << endl;
	cout << "y = " << y << endl;
	cout << "z = " << z << endl;
}
```

  Kết quả chạy chương trình:

  Giải thích:
  - Biểu thức **`!(a > b)`** kiểm tra xem có phải a không lớn hơn b hay không (không phải là a lớn hơn b). Do 5 < 7 nên kết quả phép tính này là **true**.
  - Biểu thức **`(a > b) && (b < 5)`** kiểm tra xem có phải a lớn hơn b và b nhỏ hơn 5 hay không. Do a > b sai, và b < 5 đúng, có 1 điều kiện không thỏa mãn nên kết quả của phép tính trên là **false**.
  - Biểu thức **`(a < b) || (b > 10)`** kiểm tra xem có phải a nhỏ hơn b hoặc là b lớn hơn 10 không, nếu ít nhất một trong hai điều kiện thỏa mãn thì kết quả sẽ là đúng. Mà a < b đúng, nên kết quả phép tính trên là **true**.
- Các phép toán trả về kiểu luận lý (đúng / sai):
  - Phép so sánh bằng (==), lưu ý: phép này ghi 2 dấu bằng liên tiếp.
  - Phép so sánh không bằng (!=).
  - Phép so sánh lớn hơn (>).
  - Phép so sánh lớn hơn hoặc bằng (>=).
  - Phép so sánh bé hơn (<).
  - Phép so sánh bé hơn hoặc bằng (<=).

Ví dụ:

```
#include<iostream>
using namespace std;
int main()
{
	int a, b, c, d;
	a = 5;
	b = 7;
	c = 6;
	d = 5;

	bool bang = (a == 5); // true
	bool khac = (b != 7); // false
	bool lon_hon = (a > b); // false
	bool lon_hon_hoac_bang = (a >= d); // true
	bool be_hon = (a < c); // true
	bool be_hon_hoac_bang = (b <= d); // false
	cout << "kiem tra a == 5: " << bang << endl;
	cout << "kiem tra b != 7: " << khac << endl;
	cout << "kiem tra a > b: " << lon_hon << endl;
	cout << "kiem tra a >= d: " << lon_hon_hoac_bang <<
endl;
	cout << "kiem tra a < c: " << be_hon << endl;
	cout << "kiem tra b <= d: " << be_hon_hoac_bang <<
endl;
}
```

Kết quả chạy chương trình:

```
kiem tra a == 5: 1
kiem tra b != 7: 0
kiem tra a > b: 0
kiem tra a >= d: 1
kiem tra a < c: 1
kiem tra b <= d: 0
```

4. Các phép toán khác

a. Phép gán phức

- Các phép toán +=, -=, *=, /=, %= .

Ví dụ:

a = a + 5; tương đương với a += 5;

x = x * 5; tương đương với x *= 5;

Kết quả phép gán phức tương đương với việc gán biến bên trái với giá trị nhận được sau khi thực hiện phép toán đối với giá trị bên phải.

Chương trình ví dụ:

```
#include<iostream>
using namespace std;
int main()
{
	int a = 5;
	int b = 2;
	int c = 8;
	int d = 6;
	int e = 11;

	a += 7; // a = a + 7
	b -= c; // b = b - c
	c *= d; // c = c * d
	d /= 4; // d = d / 4
	e %= 3; // e = e % 3

	cout << "a = " << a << endl;
	cout << "b = " << b << endl;
	cout << "c = " << c << endl;
	cout << "d = " << d << endl;
	cout << "e = " << e << endl;
}
```

Kết quả chạy chương trình:

```
a = 12
b = -6
c = 48
d = 1
e = 2
```

b. Phép tăng, giảm 1 đơn vị

Phép toán tăng 1 đơn vị (++) và giảm 1 đơn vị (--) có tác dụng tăng hoặc giảm 1 đơn vị của biến đó. Có 2 kiểu tăng giảm:

+ Tăng, giảm trước: đặt trước biến, giá trị của biến sẽ thay đổi trước khi thực hiện các phép toán khác trong biểu thức. Ví dụ: ++a, --b

+ Tăng, giảm sau: đặt sau biến, giá trị của biến sẽ thay đổi sau khi thực hiện các phép toán khác trong biểu thức. (phép này thường sử dụng phổ biến hơn). Ví dụ: a++, b—

Chương trình ví dụ:

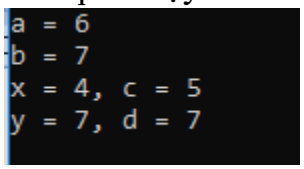
```
#include<iostream>
using namespace std;
int main()
{
	int a = 5;
	int b = 8;
	int c = 4;
	int d = 6;

	a++; // a = a + 1
	b--; // b = b - 1
	cout << "a = " << a << endl;
	cout << "b = " << b << endl;

	int x = c++;
	/* Tuong duong voi
		x = c;
		c = c + 1;
	*/
	cout << "x = " << x << ", c = " << c << endl;

	int y = ++d;
	/* Tuong duong voi
		d = d + 1;
		y = d;
	*/
	cout << "y = " << y << ", d = " << d << endl;
}
```

Kết quả chạy chương trình:



c. Phép nhóm các số hạng

Cú pháp: đặt biểu thức trong cặp dấu ngoặc tròn để ưu tiên thực hiện, tương tự như khi làm toán, ưu tiên tính trong ngoặc trước:

(biểu thức)

Ví dụ:

```
#include<iostream>
using namespace std;
int main()
{
	int a = 5;
	int b = 6;
	int c = 8;
	int result = (a + b) * (c + 2);
	cout << "result = " << result;
}
```

Kết quả chạy chương trình:

```
result = 110
```

d. Phép chuyển kiểu

- Dùng để chuyển đổi kiểu của biến thành kiểu khác trong biểu thức.
- Cú pháp:

(kiểu) biến

- Lưu ý: Trong biểu thức gồm 2 toán hạng, nếu cả 2 toán hạng đều là số nguyên thì sẽ thực hiện phép toán kiểu số nguyên, nếu một trong hai toán hạng là số thực thì sẽ thực hiện phép toán trên kiểu thực.

Ví dụ:

```
#include<iostream>
using namespace std;
int main()
{
	int a = 7;
	int b = 4;
	float x = a / b; // chia nguyen
	float y = a / (float)b;
	float z = a / 4.0; // phep chia kieu thuc
	// ep bien b sang kieu float trong bieu thuc tren
	cout << "x = " << x << endl;
	cout << "y = " << y << endl;
	cout << "z = " << z << endl;
}
```

Kết quả chạy chương trình:

```
x = 1
y = 1.75
z = 1.75
```

Giải thích:

- Trong phép tính **`x = a / b`** , a và b đều là số nguyên nên đây là phép chia nguyên.
- Trong phép tính **`y = a / (float)b`**, ta ép kiểu cho biến b trong biểu thức này thành kiểu float nên đây trở thành phép chia trên kiểu thực.
- Trong phép tính **`z = a / 4.0`**, vì 4.0 là kiểu thực nên đây là phép chia trên kiểu thực.

5. **Các hàm toán học thông dụng**

Tham khảo tài liệu chương 3 tt, trang 30-31.