Biến & Kiểu dữ liệu

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC MỞ TPHCM

Mục tiêu

- Hiểu các tính chất và phạm vi biểu diễn của các kiểu dữ liệu cơ bản trong C++.
- Biết cách sử dụng các kiểu dữ liệu cơ bản (bool, char, int, double, để lưu trữ dữ liệu trong chương trình.
- Biết cách định nghĩa, khởi tạo biến và hằng trong C++.
- Biết cách viết một biểu thức số học và câu lệnh gán, biết cách sử dụng một số hàm toán học trong biểu thức trong C++.
- Biết cách viết biểu thức luận lý để so sánh giá trị dữ liệu.
- Biết cách dùng cin/cout để viết câu lệnh nhập/xuất.
- Có khả năng viết một chương trình C++ đơn giản, nhập và xử lý dữ liệu, sau đó trình bày kết quả.
- Biết cách nhập/xuất chuỗi ký tự bằng cách dùng kiểu string trong thư viện.

Nội dung

1. Tên và Từ khóa
2. Kiểu dữ liệu
3. Biến và Hằng
4. Nhập dữ liệu
5. Xuất dữ liệu và Định dạng xuất
6. Biểu thức

Từ khóa (keyword)

- Từ khóa hay từ dành riêng (keywords/reserved words) là các từ có ý nghĩa đặc biệt đối với chương trình dịch.

- Ví dụ một số từ khóa trong C++: `int double char const void return` …

- Các ký tự trong từ khóa C++ là chữ thường. Không được đặt các tên trong chương trình trùng với từ khóa.

Tên (name)

- Tên (name) hay danh hiệu, định danh (identifier) là tên biến, tên hằng, tên hàm, tên kiểu dữ liệu do người lập trình định nghĩa và được đặt theo luật.

- Luật đặt tên trong C++:
  - Tên phải bắt đầu bằng ký tự.
  - Tên chỉ gồm ký tự chữ, số và dấu gạch dưới. Không được phép có khoảng trắng và các ký tự khác (dấu chấm, dấu phẩy, $, %...)
  - Tên không được trùng với từ khóa.
  - Tên trong C++ phân biệt chữ thường và hoa (case sensitive).

- Ví dụ các tên hợp lệ:

```
x grossPay number_of_studentes
n2
Polygon
PI
TAX_RATE
```

- Ví dụ các tên không hợp lệ:

```
2x start$time
Start menu
RATE%
2018Sales
```

- C++ phân biệt ký tự hoa và thường, x và X là hai tên khác nhau.

Lưu ý khi đặt tên

- Nên chọn tên có ý nghĩa, giúp người đọc có thể hiểu chương trình một cách dễ dàng.
- Không nên viết tắt vì sẽ gây khó hiểu:
  `mtbf  TLA  myw  nbv`
- Các tên ngắn gọn đã được quy ước:
  - `x`: biến cục bộ
  - `i`: chỉ số của vòng lặp
- Không nên dùng tên quá dài:
  `the_number_of_elements`
  `remaining_free_slots_in_the_symbol_table`
- Nên đặt tên ngắn gọn, có nghĩa:
  `partial_sum  element_count  staple_partition`

Quy ước đặt tên

- Tên biến: `lowerCamelCase`

```
double centimeter;
int totalInches;
```

- Tên hàm: `lowerCamelCase`, bắt đầu bằng động từ.

```
double calculateAverage();
void run();
void print();
```

- Tên kiểu dữ liệu do người dùng định nghĩa: `UpperCamelCase`

```
Graph
Square
```

- Tên hằng: ký tự in hoa, dùng dấu gạch dưới phân cách các từ.

```
const int MAX_PARTICIPANTS = 10;
```

Bài tập

- Hãy cho biết các tên nào là hợp lệ trong C++:

```
x1
EvilDarkness
PennslyvaniaAve1600
1600PennsylvaniaAve
Bobby_the_Robot Bobby+the+Robot whatThe???
amount
count2
count2five
5count
main
main2
```

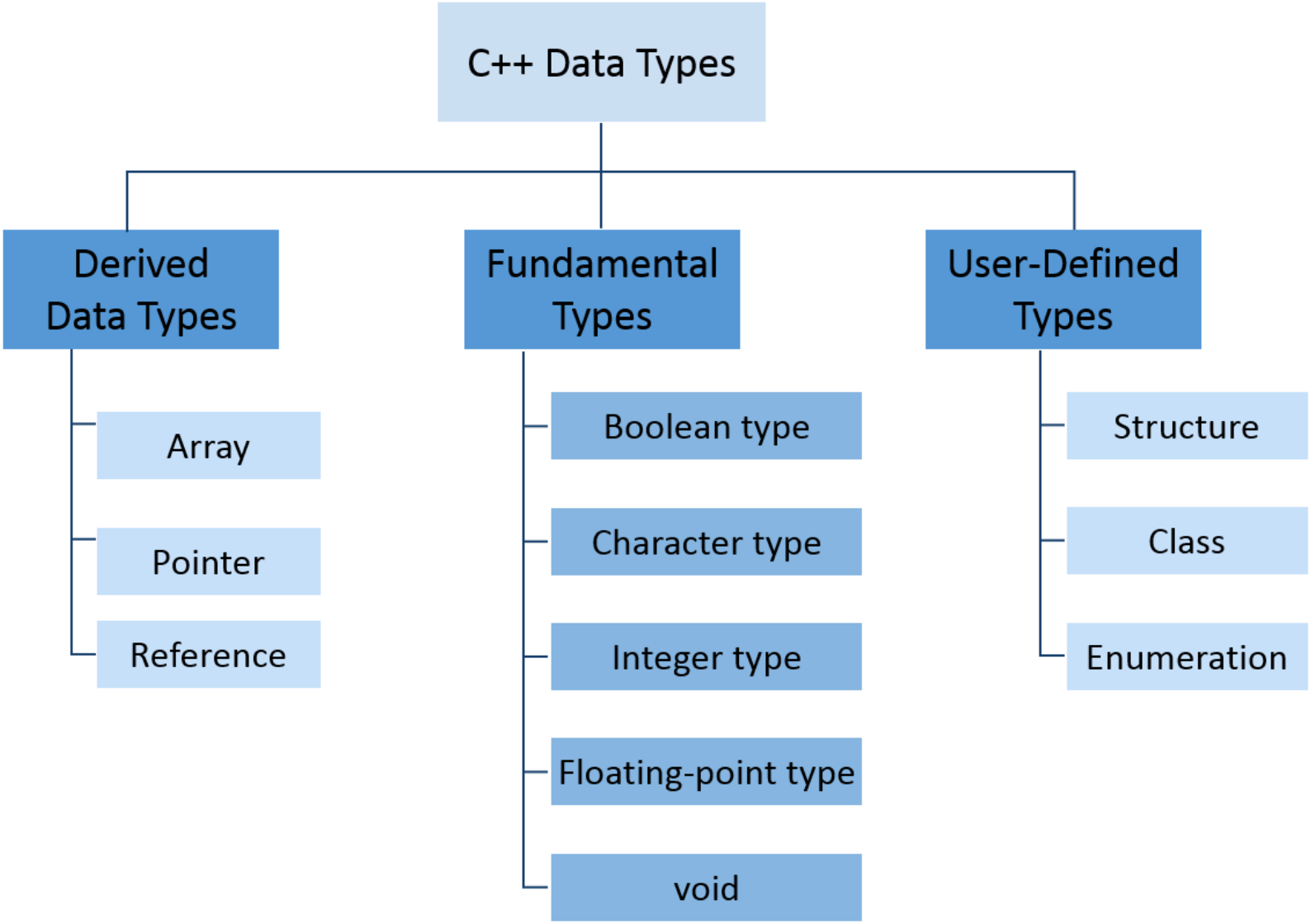
Nội dung

1. Tên và Từ khóa
2. Kiểu dữ liệu
   - Kiểu dữ liệu cơ bản
   - Kiểu `string`
   - Một số toán tử tương ứng với kiểu dữ liệu
3. Biến và Hằng
4. Nhập dữ liệu
5. Xuất dữ liệu và Định dạng xuất
6. Biểu thức

Kiểu dữ liệu

- Dữ liệu có thể là một ký tự, một số hoặc một đoạn văn bản...
- Khi viết chương trình ta phải cho máy tính biết kiểu dữ liệu (data type) cần xử lý.
- Các ngôn ngữ lập trình đều sử dụng hai kiểu dữ liệu cơ bản:
  - Dữ liệu số (numeric data type): số nguyên (integer) và số dấu chấm động (floating point).
  - Dữ liệu chuỗi ký tự (character string data type): ký tự (character) và chuỗi (string).
- Ngoài ra, các ngôn ngữ lập trình còn có kiểu dữ liệu luận lý (boolean data type).

Kiểu dữ liệu trong C++



Kiểu (type)	Mô tả	Phạm vi (range)
`char`	số nguyên 1-byte	0...255
`unsigned char`	số nguyên không dấu 1-byte	0...255
`signed char`	số nguyên có dấu 1-byte	-128...127
`short`	số nguyên 2-byte	-32,768...32,767
`unsigned short`	số nguyên không dấu 2-byte	0...65,535
`int`	số nguyên 4-byte	xấp xỉ ± 2 tỷ
`unsigned int`	số nguyên không dấu 4-byte	xấp xỉ 4 tỷ
`long`	số nguyên 4-byte	xấp xỉ ± 2 tỷ
`unsigned long`	số nguyên không dấu 4-byte	xấp xỉ 4 tỷ
`bool`	lưu giá trị luận lý	true hoặc false
`float`	số thực chính xác 7 chữ số	3.4×10^{-38}...3.4×10^{38}
`double`	số thực chính xác 15 chữ số	1.8×10^{-308}...1.8×10^{308}

Kiểu dữ liệu cơ bản

- `bool`
  - Kích thước: 1 byte.
  - Lưu giá trị `true` (khác 0) hoặc `false` (0).
- `char`
  - Kích thước: 1 byte.
  - Lưu trữ các ký tự chữ, ký tự số và ký tự đặc biệt trong bảng mã ASCII (0-255). Ký tự đặt giữa hai dấu nháy đơn:
    - `'A' 'a' '0' '*' '+' '&' ' '` (character literals)
  - Một số ký tự đặc biệt:
    - `'\n'`: newline
    - `'\t'`: tab
    - `'\b'`: backspace
    - `'\\'`: backslash
    - `'\''`: single quote
    - `'\"'`: double quote

  (`'\'`: escape character)

- `int`
  - Kích thước: 4 byte.
  - Lưu các giá trị từ -2,147,483,648 đến 2,147,483,647.
  - Giá trị cực đại: INT_MAX
  - Gía trị cực tiểu: INT_MIN
  - Lưu ý khi ghi các giá trị số nguyên (number literals):
    - Số nguyên dương không cần dấu +.
    - Không có dấu phẩy (,) trong số nguyên.
  - Ví dụ: -6728 -67 0 78 35267 763
- `double`
  - Kích thước: 8 byte
  - Lưu các giá trị từ 1.7×10^{-308} đến 1.7×10^{308} (15 chữ số).
  - C++ biểu diễn số thực dùng dấu chấm động.
  - Ví dụ: 1.23 .23 0.23 1.0 1.2e10 1.23e-15

Kiểu `string`

- Kiểu string (class string) lưu trữ chuỗi ký tự. Đây là kiểu dữ liệu có sẵn trong thư viện.
- Chuỗi ký tự gồm 0 hoặc nhiều ký tự và được đặt giữa hai dấu nháy kép.

```
"It is a beautiful day."
"Mickey"
""
```

- Độ dài chuỗi tùy ý.
- Để sử dụng kiểu string ta phải thêm chỉ thị:

```
#include <string>
```

Một số toán tử tương ứng với kiểu dữ liệu

	bool	char	int	double	string
gán	=	=	=	=	=
cộng			+	+	
nối chuỗi					+
trừ			-	-	
nhân			*	*	
chia			/	/	
chia lấy dư			%		
tăng 1			++	++	
giảm 1			--	--	
tăng n			+=n	+=n	
thêm vào cuối					+=
giảm n			-=n	-=n	

	bool	char	int	double	string
nhân và gán			*=	*=	
chia và gán			/=	/=	
chia dư và gán			%=		
đọc từ s vào x	s >> x	s >> x	s >> x	s >> x	s >> x
ghi từ x ra s	s << x	s << x	s << x	s << x	s << x
so sánh bằng	==	==	==	==	==
không bằng	!=	!=	!=	!=	!=
lớn hơn	>	>	>	>	>
lớn hơn hoặc bằng	>=	>=	>=	>=	>=
nhỏ hơn	<	<	<	<	<
nhỏ hơn hoặc bằng	<=	<=	<=	<=	<=

Nội dung

1. Tên và Từ khóa
2. Kiểu dữ liệu
3. Biến và Hằng
   - Biến (variable)
   - Hằng (named constant)
4. Nhập dữ liệu
5. Xuất dữ liệu và Định dạng xuất
6. Biểu thức

Biến

- Biến (variable) là một vùng nhớ dùng để lưu trữ một giá trị dữ liệu. Mỗi biến có tên (name) và kiểu dữ liệu (type).
- Dữ liệu được lưu trữ trong biến gọi là giá trị (value) của biến.

- Khi thực hiện, chương trình có thể đọc, ghi, hoặc thay thế các giá trị của biến.
- C++ yêu cầu phải khai báo biến trước khi sử dụng.
- Để khai báo một biến, ta cần phải biết kiểu dữ liệu (data type) mà chương trình sẽ sử dụng (số, chuỗi...).

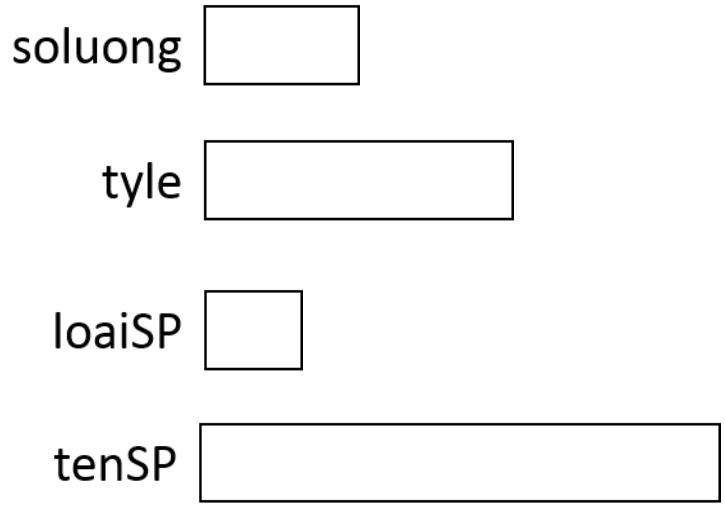
- Cú pháp khai báo biến:

```
dataType variableName;
```

  - `dataType`: kiểu dữ liệu (có trong C++, là kiểu cơ bản hoặc kiểu do người dùng tạo).
  - `variableName`: tên biến (theo luật đặt tên của C++).

- Ví dụ:

```
int soluong;
double tyle;
char loaiSP;
string tenSP;
```



- Có thể khai báo nhiều biến có cùng kiểu dữ liệu:

```
int a, b, c, d;
```

- Cú pháp khai báo biến và khởi tạo giá trị:

```
dataType variableName = initialValue;
```

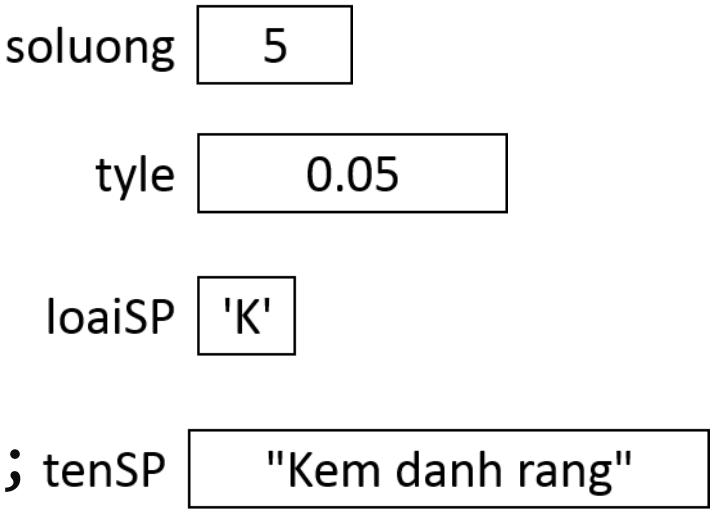
- Ví dụ:

```
int soluong = 5;

double tyle = 0.05;

char loaiSP = 'K';

string tenSP = "Kem danh rang";
```



  - 5, 0.05, 'K', "Kem danh rang" được gọi là *literal constant*.

Ví dụ

```
//In gia tri cua cac bien
#include <iostream>
using namespace std;

int main()
{
    int soTao = 5;
    int soCam = 20;
    int tongcong = soTao + soCam;

    cout << "So trai tao: " << soTao << endl;
    cout << "So trai tao: " << soCam << endl;
    cout << "Tong cong: " << tongcong << endl;
    return 0;
}
```

Một số lỗi thường gặp

- Sử dụng biến chưa khai báo.

```
double x = 12 * y; //loi vi bien y chua khai bao
double y = 0.0296;
```

- Sử dụng biến chưa được khởi tạo giá trị.

```
double x;
double y = x * 2;
```

- Gán giá trị khác kiểu cho biến.

```
string hoten = 39; //loi
int n = "Hoai Thu"; //loi
```

Hằng

- Hằng (named constant) là một vùng nhớ dùng để lưu trữ một giá trị dữ liệu không thay đổi trong suốt thời gian chương trình thực thi.
- Cú pháp định nghĩa hằng:

```
constdataType constantName = value;
```

  - dataType: kiểu dữ liệu
  - constantName: tên hằng (theo luật đặt tên của C++), thường dùng chữ in hoa.
  - value: giá trị phù hợp với kiểu dữ liệu.

- Ví dụ:

```
const double PI = 3.141593;
const double CM_PER_INCH = 2.54;
const int SISO = 20;
const char BLANK = ' ';
```

Bài tập

- Cho lưu đồ IPO như sau:

Input	Processing	Output
số lượng bán, giá mua, giá bán	*Processing items*: tiền chênh lệch *Algorithm*: 1. nhập số lượng bán, giá mua và giá bán 2. tính tiền chênh lệch giữa giá mua và giá bán 3. tính tiền lời bằng cách nhân tiền chênh lệch với số lượng bán 4. tiền lời	tiền lời

  - Hãy cho biết cần có bao nhiêu biến, bao nhiêu hằng.
  - Viết khai báo số lượng bán có kiểu int, các dữ liệu còn lại có kiểu double.

Nội dung

1. Tên và Từ khóa
2. Kiểu dữ liệu
3. Biến và Hằng
4. Nhập dữ liệu
   - Câu lệnh nhập dữ liệu
   - Câu lệnh gán
5. Xuất dữ liệu và Định dạng xuất
6. Biểu thức

Nhập dữ liệu

- Dùng câu lệnh nhập (input statement): đọc dữ liệu từ bàn phím.

```
cin >> variableName;
```

- Ví dụ:

```
cin >> bankinh;
```

- Dùng câu lệnh gán (asignment statement).

```
variableName = expression;
```

  - Giá trị của biểu thức phải phù hợp với kiểu dữ liệu của biến.
  - Dấu = gọi là toán tử gán (assignment operator).

- Ví dụ:

```
dientich = 3.14 * bankinh * bankinh;
```

Câu lệnh nhập dữ liệu

keyboard → `cin` object → >> (extraction operator) → internal memory

- Đối tượng `cin` được sử dụng cùng với toán tử >>.
- Toán tử >> sẽ lấy ra các ký tự từ đối tượng và chuyển vào bộ nhớ, cho đến khi gặp ký tự khoảng trắng (tab, space, enter).

Ví dụ 4.1.1

- Viết chương trình nhập vào bán kính hình tròn, tính và in ra diện tích hình tròn.

Input	**Processing**	**Output**
bán kính	*Algorithm*: 1. nhập độ bán kính 2. tính diện tích = 3.14 * bán kính * bán kính 3. in ra độ diện tích	diện tích

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

int main()
{
    const double pi = 3.14;
    double bankinh;
    double dientich;

    cout << "Nhap ban kinh hinh tron: ";
    cin >> bankinh;
    dientich = pi * bankinh * bankinh;

    cout << "Dien tich hinh tron la: " << dientich
         << endl;

    return 0;
}
```

Ví dụ 4.1.2

- Viết chương trình nhập vào độ Celsius, chuyển sang độ Fahrenheit và in ra kết quả.

Input	**Processing**	**Output**
celsius	Algorithm: 1. nhập độ celsius 2. tính độ fahrenheit = (độ celsius * 1.8) + 32 3. in ra độ Fahrenheit	fahrenheit

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

int main()
{
    double doC = 0.0;    //do Celsius
    double doF = 0.0;    //do Fahrenheit

    cout << "Nhap do Celsius: ";
    cin >> doC;

    doF = (doC * 1.8) + 32;

    cout << "Do Fahrenheit la: " << doF << endl;

    return 0;
}
```

Ví dụ 4.1.3

- Viết chương trình nhập một số và in ra bình phương của số đó.

Input	Processing	Output
x	*Algorithm:* 1. nhập giá trị vào x 2. in ra x * x	x * x

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

int main()
{
    double x = 0.0;

    cout << "Nhap mot so: ";
    cin >> x;

    cout << "Binh phuong la: " << x * x << endl;

    return 0;
}
```

Ví dụ 4.1.4

- Viết chương trình nhập vào chiều cao (m) và cân nặng (kg), tín và in ra chỉ số BMI (body mass index) của người đó.
- Công thức tính như sau:

$$bmi = \frac{\text{cân nặng (kg)}}{\text{chiều cao} \times \text{chiều cao (m)}}$$

Input	**Processing**	**Output**
chiều cao	*Algorithm:*	BMI
cân nặng	1. nhập chiều cao	
	2. nhập cân nặng	
	3. tính BMI = cân nặng / (chiều cao * chiều cao)	
	4. in ra BMI	

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

int main()
{
    double cannang = 0.0;
    double chieucao = 0.0;
    double bmi = 0.0;

    cout << "Nhap can nang va chieu cao: ";
    cin >> cannang >> chieucao;

    bmi = cannang / (chieucao * chieucao);
    cout << "BMI la " << bmi << endl;

    return 0;
}
```

Bài tập

- Viết chương trình nhập vào độ Fahrenheit, chuyển sang độ Celsius và in ra kết quả.
  - Công thức chuyển đổi là doC = (doF - 32) / 1.8.)

- Viết chương trình nhập một số và in ra lập phương của số đó.

Nhập dữ liệu kiểu string

```
getline(cin, str);
```

- str là biến có kiểu string
- đọc một chuỗi, kể cả khoảng trắng và lưu vào str

- Ví dụ:

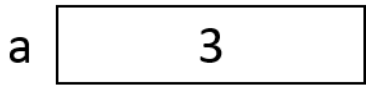
```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string hoten;
    cout << "Nhap vao ho va ten: ";
    getline(cin, hoten);
    cout << "Chao ban " << hoten << endl;
}
```

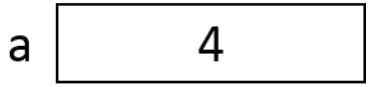
Câu lệnh gán

- Câu lệnh gán (assignment statement) dùng để gán giá trị cho biến trong khi chương trình đang thực hiện.

```
int a = 3;
```




```
a = 4;
```




```
int b = a;
```

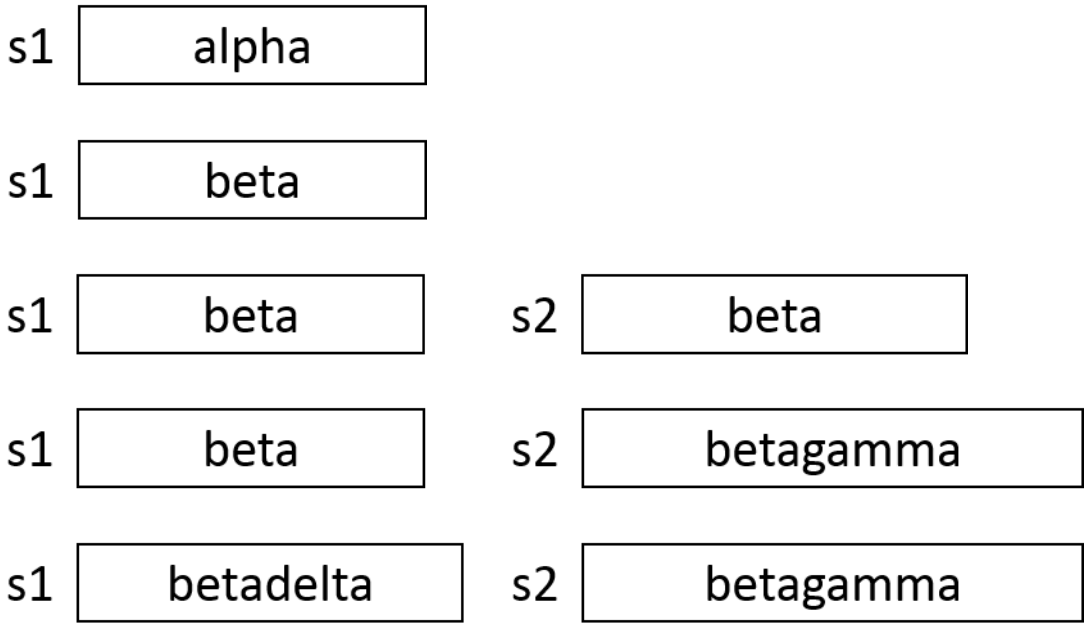

```
b = a + 5;
```


```
a = a + 7;
```

- Kiểu dữ liệu của biểu thức trong câu lệnh gán phải phù hợp với kiểu dữ liệu của biến.

- Có thể dùng câu lệnh gán cho biến kiểu string:

```
string s1 = "alpha";
s1 = "beta";
string s2 = s1;
s2 = s1 + "gamma";
s1 = s1 + "delta";
```



- Câu lệnh khai báo và khởi tạo (initialization): tạo một biến mới và gán cho biến giá trị ban đầu.
- Câu lệnh gán (assignment): gán cho biến đã có một giá trị mới.

- Ví dụ chương trình tính giá bán tivi, tỷ lệ giảm giá là 15%.

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

int main()
{
    const double tylegiam = 0.15;
    double giaban = 0.0;
    double sotiengiam = 0.0;
    double sotientra = 0.0;

    cout << "Nhap gia tivi: ";
    cin >> giaban;
    sotiengiam = giaban * tylegiam;
    sotientra = giaban - sotiengiam;

    cout << "So tien phai tra: " << sotientra <<
endl;
}
```

Toán tử gán phức hợp

- Các toán tử gán phức hợp (composite assignment operators) dùng để rút ngắn câu lệnh gán:
  - += cộng gán
  - -= trừ gán
  - *= nhân gán
  - /= chia gán
  - %= modulo gán
- Ví dụ:
  - a += 7; //a = a + 7
  - b -= 9; //b = b - 9
  - c *= 2; //c = c * 2
- counter += 1; //counter = counter + 1; tương đương ++counter;

Nội dung

1. Tên và Từ khóa
2. Kiểu dữ liệu
3. Biến và Hằng
4. Nhập dữ liệu
5. Xuất dữ liệu và Định dạng xuất
6. Biểu thức

Câu lệnh xuất

- Câu lệnh xuất (output statement): in dữ liệu ra màn hình.

```
cout << expression or manipulator << ...;
```

  - expression: tính giá trị của biểu thức sau đó in giá trị.
  - manipulator: định dạng dữ liệu xuất.

- Ví dụ:

```
cout << 29 / 4 << endl;
cout << 'A' << endl;
cout << "Hello \nthere!" << endl;
cout << "4 + 8 = " << 4 + 8 << endl;
cout << "...1\n....2\n.....3\n";
hoặc
cout << "...1" << endl << "....2" << endl
     << ".....3" << endl;
```

Định dạng xuất

- Môi trường lập trình thường xuất 6 chữ số có nghĩa (mặc định):

```
cout << 12.345678 << endl; //12.3457
```

- Để định dạng dữ liệu xuất, ta phải thêm chỉ thị:

```
#include <iomanip>
```

- Ví dụ hiển thị 2 chữ số lẻ:

```
cout << fixed << setprecision(2)
     << 12.345678 << endl;                    //12.35
```

- Ví dụ quy định chiều rộng cột hiển thị:

```
cout << "0123456789" << endl; //0123456789
cout << setw(10) << 12.345678 << endl; // 12.3457
```

Nội dung

1. Tên và Từ khóa
2. Kiểu dữ liệu
3. Biến và Hằng
4. Nhập dữ liệu
5. Xuất dữ liệu và Định dạng xuất
6. Biểu thức
   - Biểu thức số học
   - Biểu thức luận lý

Biểu thức

- Biểu thức (expression) dùng để tính toán các giá trị, được tạo thành từ toán tử (operators) và toán hạng (operands).
  - Toán tử: thực hiện tính toán với toán hạng và cho kết quả là giá trị có một kiểu.
  - Toán hạng: dữ liệu được toán tử tính toán.
- Ví dụ:

```
2 + 3 * 5
x - y / 7
x * 10.5 + y - 16.2
2 + 3.5
6 / 4 + 3.9
5.4 * 2 - 13.6 + 18 / 2
```

Độ ưu tiên của toán tử

- Khi tính giá trị biểu thức: áp dụng luật ưu tiên của các toán tử.

Toán tử	Độ ưu tiên
! – (unary operator)	1
* / %	2
+ –	3
< <= >= >	4
== !=	5
&&	6
||	7
= (assignment operator)	8

- Ví dụ:
  - (dai + rong) * 2 khác với dai + rong * 2
  - a * b + c / d nghĩa là (a * b) + (c / d)
không phải a * (b + c) / d

Lưu ý khi viết biểu thức

- Thêm khoảng trắng trước và sau các toán tử trong biểu thức.

```
x1 - (-b + sqrt(b * b - 4 * a * c)) / (2 * a);
```

dễ đọc hơn:

```
x1-(-b+sqrt(b*b-4*a*c))/(2*a);
```

- Không thêm khoảng trắng sau tên hàm.

```
sqrt(x),
```

không viết

```
sqrt (x)
```

Biểu thức số học

- Biểu thức số gồm toán tử số học và các toán hạng là số nguyên hoặc số dấu chấm động.
  - Toán tử số học: + – * / %
  - Thứ tự ưu tiên: * /
    %
    + –
  - Lưu ý: toán tử % (modulo) chỉ dùng cho kiểu số nguyên.

- Biểu thức kiểu số nguyên nếu các toán hạng có kiểu số nguyên.

- Biểu thức kiểu số dấu chấm động nếu các toán hạng có kiểu số số dấu chấm động.

- Ví dụ biểu thức kiểu số nguyên:

```
int dai = 4;
int rong = 5;
int dientich, chuvi;

dientich = dai * rong;
chuvi = (dai + rong) * 2
```

- Ví dụ biểu thức kiểu số dấu chấm động:

```
const double PI = 3.1414296;
double bankinh = 5.5;
double dientich;

dientich = PI * bankinh * bankinh;
```

- Ví dụ viết các biểu thức số học bằng C++:

Biểu thức số học	Biểu thức C++	Ghi chú
$\frac{x+y}{2}$	`(x + y) / 2`	cần có dấu ngoặc: x + y / 2 sẽ tính x + $\frac{y}{2}$
$\frac{xy}{2}$	`x * y / 2`	không cần dấu ngoặc: toán tử cùng độ ưu tiên tính từ trái sang
$\left(1+\frac{r}{100}\right)^n$	`pow( + r / 100, n)`	
$\sqrt{a^2+b^2}$	`sqrt(a * a + b * b)`	a * a đơn giản hơn pow(a,2)
$\frac{i+j+k}{3}$	`(i + j + k) / 3.0`	nếu là số nguyên thì 3.0 chuyển sang số dấu chấm động

Lưu ý

- Nếu chia hai số nguyên thì phần dư sẽ bị cắt bỏ.

```
int tongsogiay = 3790;
int sogio = tongsogiay / 3600; // sogio == 1
```

- Phép chia lấy dư % dùng để lấy phần dư.

```
int sophut = tongsogiay % 3600 / 60;//sophut==3
int sogiay = tongsogiay % 3600 % 60;//sogiay==10
```

- Tính lũy thừa và tính căn bậc 2: sử dụng hàm thư viện.
  - $\sqrt{x}$: `sqrt(x)`
  - x^n: `pow(x, n)`

Một số hàm toán học

- Để sử dụng hàm toán học, thêm header file `cmath` vào đầu chương trình:

```
#include <cmath>
```

Hàm	Mô tả	Kiểu tham số	Kiểu trả về
abs(x)	|x|	int (double)	int (double)
exp(x)	e^x,e = 2.718	double	double
pow(x, y)	x^y	double	double
sqrt(x)	$\sqrt{x}$	double	double
sin(x)	sinx, x: radians	double	double
cos(x)	cos x, x: radians	double	double
tan(x)	tanx, x: radians	double	double
log(x)	ln(x),x > 0	double	double
log10(x)	$\log_{10}$(x),x > 0	double	double

Toán tử tăng và giảm

- Toán tử tăng 1:

```
++variable
variable++
```

- Toán tử giảm 1:

```
--variable
variable--
```

- Ví dụ:
  - `++count;` hoặc `count++;`
  - `--count;` hoặc `count--;`
  -
```
x = 5;
y = ++x; //x==6, y==6
```
  -
```
x = 5;
y = x++; //y==5, x==6
```

Ví dụ 6.1

```
//Nhap vao so thuc va in ra ket qua cua cac bieu thuc
#include <iostream>
using namespace std;

int main()
{
    cout << "Nhap so double: ";
    double n;
    cin >> n;
    cout << "n == " << n << endl
        << "n+1 == " << n + 1 << endl
        << "3*n == " << 3 * n << endl
        << "n^2 == " << n*n << endl
        << "n/2 == " << n / 2 << endl
        << "Can cua n == " << sqrt(n) << endl;
}
```

Chuyển đổi kiểu

- Biểu thức có thể gồm nhiều kiểu dữ liệu khác nhau.
  - 2.5/2: double chia cho int, kết quả là double.
- Nếu toán hạng có kiểu double thì kết quả là double, nếu toán hạng có kiểu int thì kết quả là int.
  - 5/2 kết quả là 2 (không phải 2.5).
  - 2.5/2 nghĩa là 2.5/double(2) kết quả là 1.25.
  - 'a'+1 nghĩa là int('a')+1
- Khi cần chương trình dịch sẽ chuyển đổi toán hạng kiểu int sang double, hoặc toán hạng kiểu char sang int. Sau đó chương trình dịch sẽ chuyển đổi kết quả lần nữa để sử dụng như là giá trị khởi tạo, hoặc giá trị bên vế phải phép gán.

- Ví dụ:

```
double d = 2.5;
int i = 2;
double d2 = d/i;     //d2 == 1.25
int i2 = d/i; //i2 == 1
d2 = d/i;      //d2 == 1.25
i2 = d/i;      //i2 == 1
```

- Ví dụ chuyển độ Celcius sang độ Fahrenheit: kết quả là số nguyên.

```
double doC;
cin >> doC;
double doF = 9/5 * doC + 32;   // 9/5==1,
                               // khong phai 1.5
```

Giải pháp: chuyển 9 hoặc 5 hoặc cả hai sang double:

```
double doC;
cin >> doC;
double doF = 9.0/5 * doC + 32;
```

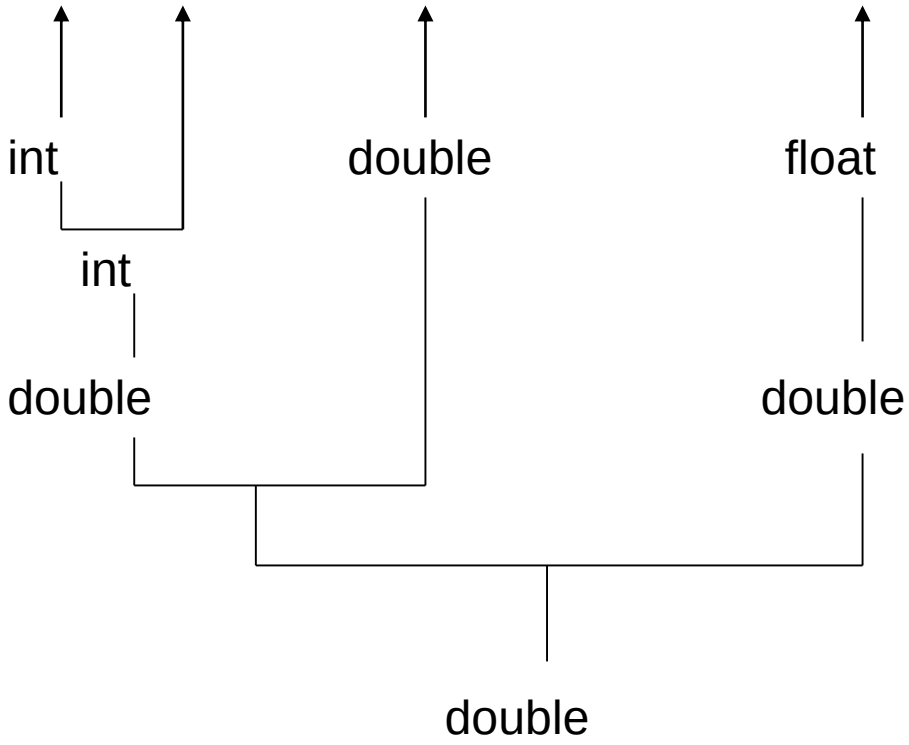
Ví dụ 6.2

```
//Chuong trinh vi du chuyen doi kieu khong tuong minh
#include<iostream>
using namespace std;

int main()
{
    cout << "3 / 2 + 5.5 = " << 3 / 2 + 5.5 << endl;
    cout << "15.6 / 2 + 5 = " << 15.6 / 2 + 5 << endl;
    cout << "4 + 5 / 2.0 = " << 4 + 5 / 2.0 << endl;
    cout << "4 * 3 + 7 / 5 - 25.5 = "
        << 4 * 3 + 7 / 5 - 25.5 << endl;
    return 0;
}
```

Chuyển đổi kiểu

```
char ch; int i; float f; double d;
result = (ch/i) + (f*d) - (f+i)
```



Chuyển đổi kiểu tường minh

```
static_cast<type>(expression)
```

- expression: biểu thức
- type: kiểu dữ liệu

- Ví dụ:

```
static_cast<int>(7.9)     7

static_cast<double>(25)   25.0

static_cast<double>(5+3)  =static_cast<double>(8)
                          =8.0
static_cast<double>(15)/2 =15.0/2 = 15.0/2.0
                          = 7.5
static_cast<double>(15/2) =static_cast<double>(7)
                          =7
```

Biểu thức luận lý

- Biểu thức luận lý có giá trị là true hoặc false, được tạo bởi toán tử luận lý và toán tử quan hệ.
  - Toán tử luận lý: `!` `&&` `||`
  - Toán tử quan hệ: `<` `>` `<=` `>=` `==` `!=`
  - Thứ tự ưu tiên:
    `!`
    `<` `<=` `>=` `>`
    `==` `!=`
    `&&`
    `||`
- Biểu thức luận lý được sử dụng trong các câu lệnh rẻ nhánh `if`, `if...else` và câu lệnh lặp `for`, `while`, `do...while`

Toán tử quan hệ

- Toán tử quan hệ dùng để so sánh các giá trị trong chương trình, cho kết kết quả là true hoặc false.

Toán tử quan hệ	Ví dụ	Ý nghĩa
>	x > y	x lớn hơn y
<	x < y	x nhỏ hơn y
>=	x >= y	x lớn hơn hoặc bằng y
<=	x <= y	x nhỏ hơn hoặc bằng y
==	x == y	x bằng y
!=	x != y	x khác y

- Các giá trị so sánh phải có cùng kiểu dữ liệu.
  - 8 < '5': không thể so sánh.

- So sánh ký tự: dựa vào mã ASCII.

```
' ' < 'a'           true
'R' > 'T'           false
'+' < '*'           false
'A' <= 'a'          true
```

- So sánh chuỗi: theo thứ tự từ điển.

```
string s1 = "Hello";
string s2 = "Hi";
string s3 = "Air";
string s4 = "Bill";
s1 < s2             true
s3 < "An"           true
s1 == "hello"       false
s3 <= s4            true
```

Toán tử luận lý

- Toán tử luận lý dùng để kết hợp các biểu thức luận lý.

Toán tử	Mô tả
!	not
&&	and
||	or

- Ví dụ:

```
(gioitinh == NU) && (tuoi >= 60)
(diemGK >= 9 ) || (diemCK >= 9)
!(xeploai == ’D’)
```

- Bảng chân trị

exp1	exp2	exp1 && exp2
false	false	false
false	true	false
true	false	false
true	true	true

exp	!exp
false	true
true	false

exp1	exp2	exp1 || exp2
false	false	false
false	true	true
true	false	true
true	true	true

- Ví dụ:

```
bool found = true;
int age = 20;
double hours = 45.30;
double overTime = 15.0;
int count = 20;
char c = 'B';

!found                                  false
hours > 40.00                           true
age == 20                               true
!found && (age >= 18)                   false
hours + overTime <= 75                  true
(count >= 0) && (count <= 100)          true
('A' <= c) && (c <= 'Z')                true
```

Q & A