CHƯƠNG 1

MỘT SỐ KHÁI NIỆM CƠ BẢN

Nội dung

1. Giới thiệu ngôn ngữ lập trình.
2. Các khái niệm cơ bản.
   - 2.1. Tập ký tự và từ khóa.
   - 2.2. Các kiểu dữ liệu cơ sở.
   - 2.3. Hằng, biến, mảng, xâu ký tự, con trỏ.
   - 2.4. Khối lệnh, toán tử, biểu thức.
   - 2.5. Các hàm thư viện chuẩn.

Giới thiệu ngôn ngữ lập trình

- Ngôn ngữ C/C++:
  - ✓ C++ là bản phát triển từ C.
  - ✓ Là ngôn ngữ lập trình cấp cao.
  - ✓ Có khả năng truy cập bộ nhớ mức thấp.
  - ✓ Phù hợp phát triển ứng dụng hệ thống.
  - ✓ Là ngôn ngữ dạng biên dịch (compile)

Giới thiệu ngôn ngữ lập trình

- Ngôn ngữ C/C++:
  - C++ kế thừa các đặc tính của ngôn ngữ C
    - ✓ Mọi chương trình viết bằng ngôn ngữ C đều có thể sử dụng lại trong C++.
    - ✓ Hỗ trợ lập trình hướng cấu trúc.
  - Hỗ trợ các nguyên lý lập trình hướng đối tượng:
    - ✓ Trừu tượng hóa (abstraction),
    - ✓ Bao đóng (encapsulation),
    - ✓ Kế thừa (inheritance),
    - ✓ Đa hình (polymorphism)

Các khái niệm cơ bản

- Từ khóa (Keywords):
  - Là những từ được dành riêng bởi ngôn ngữ lập trình cho những mục đích riêng của nó
  - Tất cả các từ khóa trong C/C++ đều là chữ thường (lowercase).
  - Danh sách các từ khóa trong C/C++

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

Kiểu dữ liệu

- Kiểu dữ liệu cơ sở của C/C++:
  - Ký tự (char)
  - Số nguyên (int)
  - Số thực (float, double)
  - Luận lý (boolean)
  - Kiểu vô định (void).
- Kích thước và phạm vi của những kiểu dữ liệu này có thể thay đổi tùy theo loại CPU và trình biên dịch.

Kiểu số nguyên

STT	TÊN KIỂU	GHI CHÚ	KÍCH THƯỚC
1	char	Ký tự	1 byte
		Số nguyên	1 byte
2	unsigned char	Số nguyên dương	1 byte
3	short	Số nguyên	2 bytes
4	unsigned short	Số nguyên dương	2 bytes
5	int	Số nguyên	4 bytes
6	unsigned int	Số nguyên dương	4 bytes
7	long	Số nguyên	4 bytes
8	unsigned long	Số nguyên dương	4 bytes

Kiểu số nguyên

```
1 #include<iostream.h>
2 void main()
3 {
4       int a;
5       long c;
6       unsigned int b;
7       unsigned long d;
8 }
```

Kiểu số thực

STT	TÊN KIỂU	GHI CHÚ	KÍCH THƯỚC
1	float		4 bytes
2	double		8 bytes
3	long double		8 bytes

```
1 #include<iostream.h>
2 void main()
3 {
4       float x;
5       double y;
6       long double z;
7 }
```

Kiểu luận lý

STT	TÊN KIỂU	GHI CHÚ	KÍCH THƯỚC
1	bool	Gồm 2 giá trị: **true** hoặc **false**	

```
#include<iostream.h>
void main()
{
    bool b=true;
    int n=0;
    while(b)
    {
        n++;
        if(n==10)
            b=false;
    }
    cout<<n;
}
```

Kiểu luận lý

```
1  #include<iostream.h>
2  bool isEven(int n);
3  void main()
4  {
5      int n;
6      cout<<"Moi ban nhap vao so n = ";
7      cin>>n;
8      if(isEven(n)==true)
9      cout<<n<<" la so chan!"
10 }
11 bool isEven(int n)
12 {
13     if(n%2==0)
14         return true;
15     else
16         return false;
17 }
```

Kiểu ký tự

- Giữ các giá trị của bộ mã ASCII (Amercican Standard Code for Information Interchange).

```
1 #include <iostream.h>
2 void main()
3 {
4       char kt='a';
5       char c;
6       cin>>c;
7 }
```

Biến

- Là định danh của một vùng trong bộ nhớ dùng để giữ một giá trị mà có thể bị thay đổi bởi chương trình.
- Tất cả biến phải được khai báo trước khi sử dụng.
- Cú pháp khai báo:

type variableNames;

  - *type*: là một trong các kiểu dữ liệu hợp lệ.
  - *variableNames*: tên của một hay nhiều biến phân cách nhau bởi dấu phẩy.

Biến

- Có thể vừa khai báo vừa khởi tạo giá trị:

  type varName1=value, ... ,varName_n=value;

- Ví dụ:

  float mark1, mark2, mark3, average = 0;

Tầm vực

- Biến cục bộ (local variables)
  - Được khai báo bên trong một hàm.
  - Các biến cục bộ chỉ được tham chiếu đến bởi những lệnh trong khối (block) có khai báo biến.
  - Một khối được đặt trong cặp dấu { }.
  - Biến cục bộ chỉ tồn tại trong khi khối chứa nó đang thực thi, và bị hủy khi khối chứa nó thực thi xong.

Tầm vực

Ví dụ:

```
void func1(void)
{
    int x;
    x = 10;
}
 void func2(void)
{
    int x;
    x = -199;
}
```

Tầm vực

- Tham số hình thức (formal parameters)
  - Nếu một hàm có nhận các đối số truyền vào hàm thì nó phải khai báo các biến để nhận giá trị của các đối số khi hàm được gọi.
  - Những biến này gọi là các tham số hình thức. Những biến này được sử dụng giống như các biến cục bộ.

Tầm vực

Ví dụ:

```
int sum(int from, int to)
{
   int total=0;
   for(int i=from ; i<=to ; i++)
      total +=i;
   return total;
}
```

Tầm vực

- Biến toàn cục (global variables)
  - Biến toàn cục có phạm vi là toàn bộ chương trình.
  - Tất cả các lệnh có trong chương trình đều có thể tham chiếu đến biến toàn cục.
  - Biến toàn cục được khai báo bên ngoài tất cả hàm.

Tầm vực

```
#include <iostream.h>
int gVar = 100;
void increase()
{  gVar = gVar + 1;}
 void decrease()
{  gVar = gVar -1;}
void main()
{
  cout << “Value of gVar= “ << gVar;   increase();
  cout << “After increased, gVar= “ << gVar;   decrease();
  cout << “After decreased, gVar= “ << gVar;
}
```

Trị hằng - const

- Giá trị của biến thay đổi trong suốt quá trình thực thi chương trình.
- Để giá trị của biến không bị thay đổi, ta đặt trước khai báo biến từ khóa const.
- Thông thường ta dùng chữ HOA để đặt tên cho những biến này.

Ví dụ:

```
const int MAX = 200;
```

Trị hằng - const

- Hằng là những giá trị cố định (fixed values) mà chương trình không thể thay đổi. Mỗi kiểu dữ liệu đều có hằng tương ứng. Hằng còn được gọi là literals.
- Hằng ký tự được đặt trong cặp nháy đơn.

  Ví dụ: ‘a’
- Hằng nguyên là những số mà không có phần thập phân.

  Ví dụ: 100, -100

Trị hằng - const

- Hằng số thực yêu cầu một dấu chấm phân cách phần nguyên và phần thập phân.

  Ví dụ: 123.45

- Cách viết một số loại hằng số

Kiểu dữ liệu	Các ví dụ về hằng	Ghi chú
int	1, 123, 21000, 234	
long int	35000L, 34l	Có ký tự l hoặc L ở cuối
unsigned int	10000U, 987u, 40000U	Có ký tự u hoặc U ở cuối
float	123.23f, 4.34e-3F	Có ký tự f hoặc F ở cuối
double	123.23, 1.0, 0.9876324	
long double	1001.2L	Có ký tự l hoặc L ở cuối

Hằng ký tự

- Hằng chuỗi ký tự là một tập các ký tự đặt trong cặp nháy kép “”.

  Ví dụ:

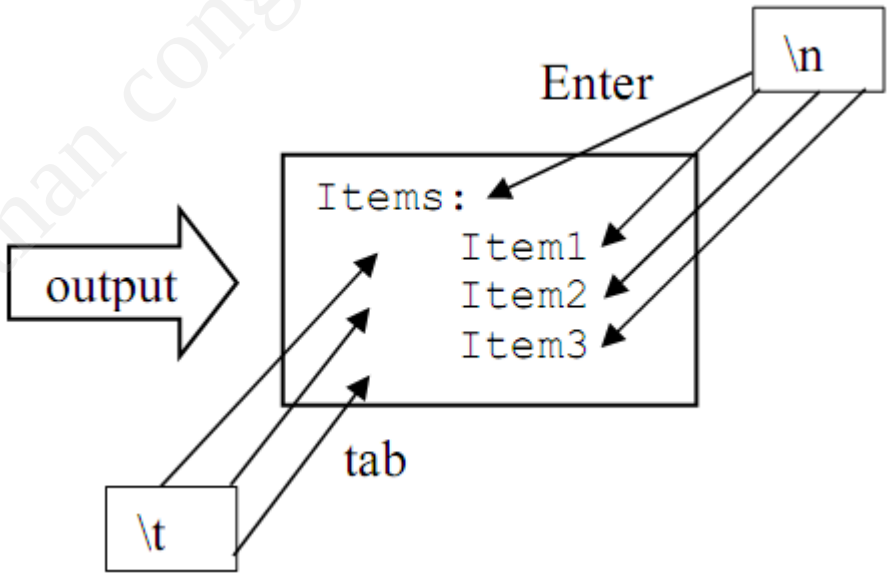
  - "This is a string" *//là một chuỗi*.
  - ‘a’ *//là một hằng ký tự*.
  - “a” *//là một hằng chuỗi*.

Hằng ký tự đặc biệt (escape sequences)

Mã	Ý nghĩa
\b	Lùi sang trái 1 ký tự
\f	Về đầu dòng
\n	Sang dòng mới
\r	Xuống dòng
\t	Tab theo chiều ngang
\"	Dấu nháy đôi
\'	Dấu nháy đơn
\0	Null
\\	Dấu \
\v	Tab theo chiều đứng
\a	Cảnh báo
\?	Dấu hỏi
\N	Hằng bát phân (với N là một hằng bát phân)
\xN	Hằng thập lục phân (với N là một hằng thập lục phân)

Hằng ký tự đặc biệt (escape sequences)

```
#include <iostream.h>
void main(void)
{
  cout <<"Items:\n";
  cout <<"\tItem1\n";
  cout <<"\tItem2\n";
  cout <<"\tItem3\n";
}
```



Định danh (Identifier Name)

- Trong C/C++, tên biến, hằng, hàm, … được gọi là định danh
- Những định danh này có thể là một hoặc nhiều ký tự. Ký tự đầu tiên phải là một chữ cái hoặc dấu _ (underscore), những ký tự theo sau phải là chữ cái, chữ số, hoặc dấu _
- C/C++ phân biệt ký tự HOA và THƯỜNG.
- Định danh không được trùng với từ khóa (keywords).

Khai báo biến

Cú pháp:

<Kiểu dữ liệu> tên biến;

Ví dụ:

int a;//Khai báo biến để lưu số nguyên tên a

float c;// Khai báo biến để lưu số thực tên c

- Khai báo nhiều biến cùng kiểu

<Kiểu dữ liệu>tên biến 1, tên biến 2, tên biến 3;

Ví dụ:

int a, x, y;

Khai báo – khởi tạo giá trị

Cú pháp:

<Kiểu dữ liệu> tên biến = giá trị;

Ví dụ:

```
int a = 5;
float b = 5.4, c=9.2;
char ch = 'n';
int a,x,y;
```

Toán tử gán (assignment operator)

- Cú pháp tổng quát

variableName = expression;

  - *variableName*: Tên biến
  - *expression*: Biểu thức

Lưu ý: phía bên trái dấu = phải là một biến hay con trỏ và không thể là hàm hay hằng.

Ví dụ:

total = a + b + c + d;

Toán tử gán – assignment operator

❖ Chuyển đổi kiểu trong câu lệnh gán

- Giá trị của biểu thức bên phải lệnh gán (=) tự động chuyển thành kiểu dữ liệu của biến bên trái.

Ví dụ:

```
int i=100;
double d = 123.456;
```

- Nếu thực thi lệnh **i** = d; thì **i** = 123 (chuyển đổi kiểu mất mát thông tin).
- Nếu thực thi lệnh **d** = **i**; thì **d** =100.0 (chuyển đổi kiểu không mất mát thông tin).

Toán tử gán – assignment operator

❖ Chuyển đổi kiểu trong câu lệnh gán

- Khi chuyển đổi từ kiểu dữ liệu có miền giá trị nhỏ sang kiểu dữ liệu có miền giá trị lớn hơn: char → int → long → float → double, thì việc chuyển đổi kiểu này là không mất mát thông tin
- Khi chuyển đổi từ kiểu dữ liệu có miền giá trị lớn sang kiểu dữ liệu có miền giá trị nhỏ hơn: double → float → long → int → char, thì việc chuyển đổi kiểu này là mất mát thông tin

Toán tử số học (arithmetic operators)

Toán tử	Tên	Ví dụ
+	Cộng	12 + 4.9 = 16.9
-	Trừ	3.98 - 4 = -0.02
*	Nhân	2 * 3.4 = 6.8
/	Chia	9 / 2.0 = 4.5
%	Lấy phần dư	13 % 3 = 1

Toán tử số học (arithmetic operators)

- Khi tử số và mẫu số của phép chia là số nguyên thì đó là phép chia nguyên nên phần dư của phép chia nguyên bị cắt bỏ.

Ví dụ: 5/2 = 2.

- Toán tử lấy phần dư % (modulus operator) chỉ áp dụng với số nguyên.

Toán tử gán phức hợp

Toán Tử	Ví dụ	Tương đương với
+=	n += 25	n = n + 25
-=	n -= 25	n = n – 25
*=	n *= 25	n = n * 25
/=	n /= 25	n = n / 25
%=	n %= 25	n = n % 25

Toán tử gán phức hợp

```
#include <iostream.h>
void main ()
{
  int a, b=3;
  a = b;
  a+=2;          //tương đương với a=a+2
  cout << a;
}
```

Toán tử ++ và -- (increment and decrement operators)

- Toán tử tăng (++) và toán tử giảm (--) có tác dụng làm tăng hoặc giảm 1 đơn vị lưu trong biến.

  Ví dụ:

  a++;//*tương đương với a+=1; và a=a+1*

  a--;//*tương đương với a-=1; và a=a-1*

Toán tử ++ và -- (increment and decrement operators)

Toán tử tăng/giảm có 2 dạng:

- Tiền tố (prefix): Toán tử **++/--** đặt trước toán hạng, hành động tăng/giảm trên toán hạng được thực hiện trước, sau đó giá trị mới của toán hạng sẽ tham gia định trị của biểu thức.

Ví dụ:

```
B=3;
A=++B;
```

Kết quả: *A chứa giá trị 4, B chứa giá trị 4*

Toán tử ++ và -- (increment and decrement operators)

- Hậu tố (postfix): Toán tử **++/--** đặt sau toán hạng, giá trị trong toán hạng được tăng/giảm sau khi đã tính toán.

  Ví dụ:

  B=3;

  A=B++;

  Kết quả: A chứa giá trị 3, B chứa giá trị 4

Toán tử ++ và -- (increment and decrement operators)

Ví dụ:

```
int x = 100;
int n,m;
n = ++x + 1; // n sẽ có giá trị là 102 (1)
n = x++ + 1; // n sẽ có giá trị là 101 (2)
```

- Sau lệnh (1), (2) thì x có giá trị là 101

```
m = --x + 1; // m sẽ có giá trị là 100 (3)
m = x-- + 1; // m sẽ có giá trị 101 (4)
```

- Sau lệnh (3), (4) thì x có giá trị là 99

Toán tử ++ và -- (increment and decrement operators)

- Khi các toán tử số học xuất hiện trong một biểu thức, thì độ ưu tiên thực hiện như sau:

Toán tử	Độ ưu tiên
++ – –	1
– (dấu âm)	2
* / %	3
+ –	4

Toán tử quan hệ & luận lý (relational & logical operators)

- Toán tử quan hệ được định trị là true hoặc false.

Toán tử	Tên	Ví dụ
==	So sánh bằng	5 == 5 // kết quả 1
!=	So sánh không bằng	5 != 5 // kết quả 0
<	So sánh nhỏ hơn	5 < 5.5 // kết quả 1
<=	So sánh nhỏ hơn hoặc bằng	5 <= 5 // kết quả 1
>	So sánh lớn hơn	5 > 5.5 // kết quả 0
>=	So sánh lớn hơn hoặc bằng	6.3 >= 5 //kết quả1

Toán tử quan hệ & luận lý (relational & logical operators)

- Toán tử luận lý:

Toán tử	Tác vụ	Ví dụ
!	Not	!(5 == 5) // kết quả là 0
&&	and	5 < 6 && 6 < 6// kết quả là 0
||	or	5 < 6 || 6 < 5 // kết quả là 1

- Bảng chân trị:

P	Q	P&&Q	P||Q	!P
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Toán tử quan hệ & luận lý (relational & logical operators)

- Độ ưu tiên của toán tử quan hệ và luận lý:

Toán tử	Độ ưu tiên
!	1
> >= < <=	2
== !=	3
&&	4
||	5

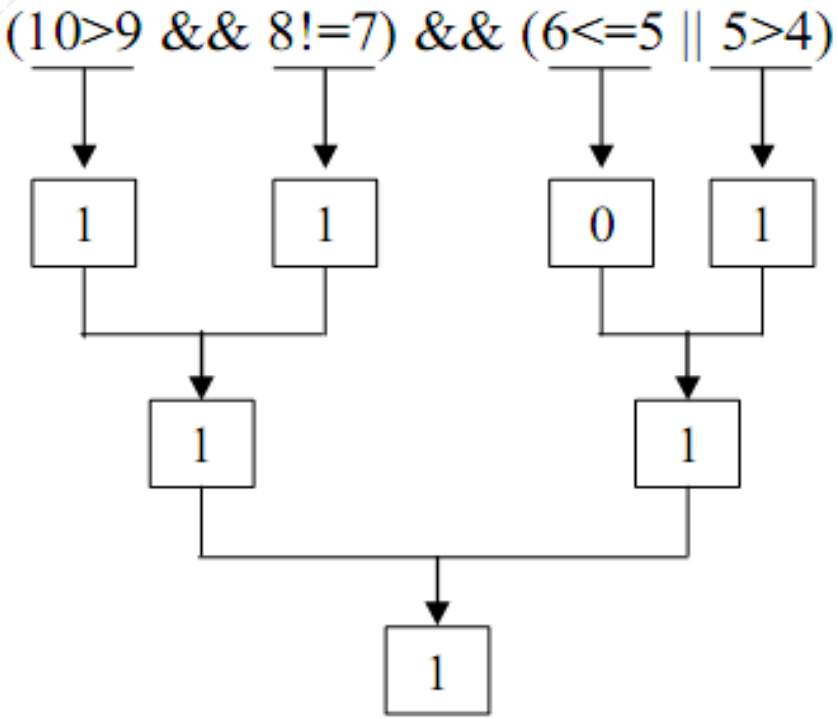
Toán tử quan hệ & luận lý (relational & logical operators)

Ví dụ:

Biểu thức:

(10>9 && 8!=7) || (6<=5 || 5>4)

Được định trị như sau:



Toán tử ? (? operator)

- ✓ Toán tử **?** là một toán tử ba ngôi do đó phải có ba toán hạng.
- ✓ Dạng tổng quát của toán tử ? là:

Exp1 ? Exp2 : Exp3;

- ✓ Exp1, Exp2, và Exp3 là các biểu thức.
- ➢ Ý nghĩa:
  - Nếu Exp1 đúng thì Exp2 được định trị và nó trở thành giá trị của biểu thức.
  - Ngược lại, nếu Exp1 sai, Exp3 được định trị và trở thành giá trị của biểu thức.

Toán tử ? (? operator)

Ví dụ:

X = 10

Y = X > 9 ? 100*X : 200*X

Vì X>9 là true nên giá trị của biểu thức sẽ là 1000.

Vậy y sẽ có giá trị là 1000.

Ví dụ:

int m = 1, n = 2, p =3;

int min =(m < n ? (m < p ? m : p) : (n < p ? n : p));

Toán tử ? (? operator)

```
#include<iostream.h>
void main()
{
      int t;
      int gio,phut,giay;
      cout<<"Nhap t: ";
      cin>>t;
      gio=(t/3600)%24;
      phut=(t%3600)/60;
      giay=(t%3600)%60;
      int giotam=(gio>12?gio-12:gio);
      cout<<giotam <<":"<<phut<<":"<<giay;
      cout<<(gio>12?"PM":"AM");
}
```

Toán tử sizeof

- **sizeof** là toán tử một ngôi mà trả về số byte của kiểu dữ liệu chiếm trong bộ nhớ. Tùy môi trường (hệ điều hành, loại CPU,...) mà mỗi kiểu dữ liệu có số byte khác nhau.
- Cú pháp:

sizeof(operand)

- *operand*: có thể là tên kiểu dữ liệu, biến, biểu thức.

Toán tử dấu phẩy (comma operator)

- Toán tử comma buộc các biểu thức cùng với nhau.
- Biểu thức bên trái của toán tử comma luôn luôn được định trị như void, biểu thức bên phải được định trị và trở thành giá trị của biểu thức.
- Dạng tổng quát của toán tử comma:

(exp_1, exp_2, ..., exp_n)

Toán tử dấu phẩy (comma operator)

- Các biểu thức được định trị từ trái sang phải, biểu thức cuối cùng (exp_n) được định trị và trở thành giá trị của toàn bộ biểu thức.

Ví dụ:

```
x = (y=3, y+1);
```

Y được gán giá trị 3, sau đó x được gán giá trị y+1 là 4.

Độ ưu tiên của các toán tử

Cao nhất	() [] -> .
	! ~ ++ -- (type) * & sizeof
	* / %
	+ -
	<< >>
	< <= > >=
	== !=
	&
	^
	|
	&&
	||
	?:
	= += -= *= /= %=
Thấp nhất	,

Thư viện chuẩn

- stdio.h, iostream.h: standard input/output
  - ✓ cout/printf(): xuất dữ liệu,
  - ✓ cin/scanf(): nhập giá trị cho biến,
  - ✓ getc(): nhận ký tự từ bàn phím,
  - ✓ putc(): in ký tự ra màn hình,
  - ✓ gets(): nhập chuỗi ký tự từ bàn phím,
  - ✓ puts(): xuất chuỗi ký tự ra màn hình,
  - ✓ fflush(): xóa vùng đệm từ bàn phím,
  - ✓ fopen(),
  - ✓ fclose(),
  - ✓ fread(),
  - ✓ fwrite(),
  - ✓ …

Thư viện chuẩn

- math.h: định nghĩa các hàm toán học.
  - ✓ abs(),
  - ✓ sqrt(),
  - ✓ log(),
  - ✓ log10(),
  - ✓ sin(),
  - ✓ cos(),
  - ✓ tan(),
  - ✓ pow(),
  - ✓ …

Thư viện chuẩn

- conio.h: định nghĩa các hàm vào/ra trong chế độ DOS.
  - ✓ clrscr(),
  - ✓ getch(),
  - ✓ getche(),
  - ✓ getpass(),
  - ✓ cgets(),
  - ✓ cputs(),
  - ✓ putch(),
  - ✓ clreol(),
  - ✓ …

Thư viện chuẩn

- alloc.h: định nghĩa các hàm liên quan đến việc quản lý bộ nhớ.
  - ✓ calloc(),
  - ✓ readllo(),
  - ✓ malloc(),
  - ✓ free(),
  - ✓ farmalloc(),
  - ✓ farcalloc(),
  - ✓ farfree(),
  - ✓ …

Thư viện chuẩn

- io.h: định nghĩa các hàm vào/ra cấp thấp.
  - ✓ open(),
  - ✓ _open(),
  - ✓ read(),
  - ✓ _read(),
  - ✓ close(),
  - ✓ _close(),
  - ✓ create(),
  - ✓ _create(),
  - ✓ eof(),
  - ✓ …