Trường Đại học Mở Tp. Hồ Chí Minh

Khoa Công nghệ Thông tin

Ôn tập

CƠ SỞ LẬP TRÌNH

Võ Thị Hồng Tuyết

Các bước xây dựng chương trình

- **Phân tích (analysis):**

  Xác định vấn đề. Mô tả các yêu cầu.

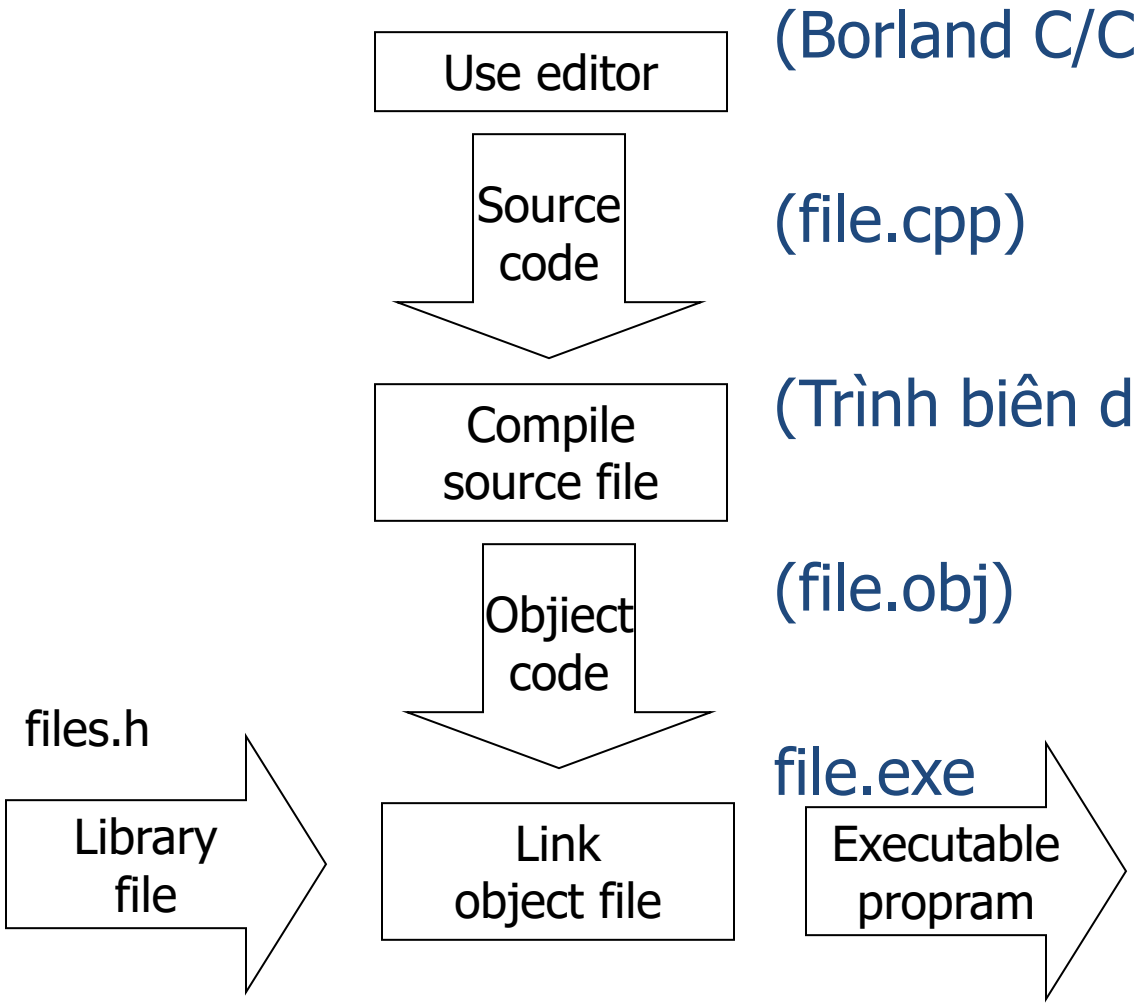
- **Thiết kế (design):**

  Mô tả thuật giải giải quyết vấn đề.

- **Lập trình (programming):**

  Mô tả giải pháp bằng mã lệnh (chương trình).

- **Kiểm thử (testing).**

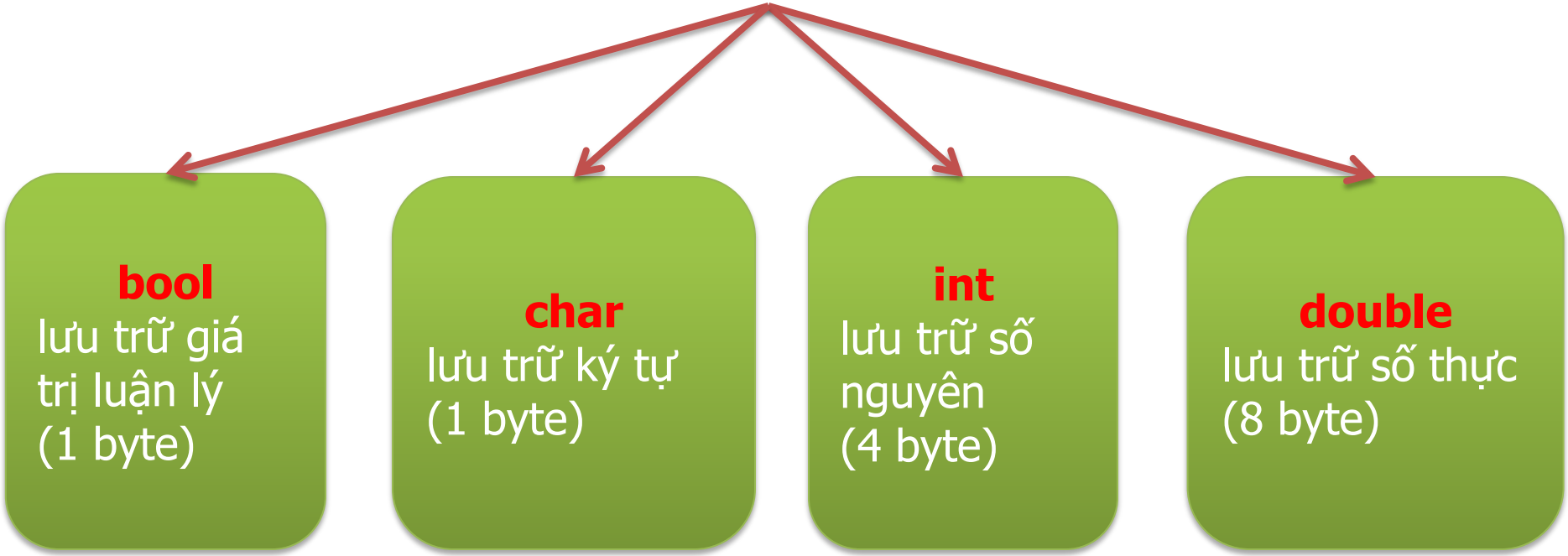

Use editor
(Borland C/C++ editor)
Source code
(file.cpp)
Compile source file
(Trình biên dịch bc.exe)
Objiect code
(file.obj)
files.h
Library file
Link object file
file.exe
Executable propram

Cấu trúc chương trình C++

```
//mô tả chương trình
#include <iostream>          //chỉ thị tiền xử lý
using namespace std;

int main()
{
      //các lệnh
}
```

Kiểu cơ bản


bool
lưu trữ giá trị luận lý (1 byte)
char
lưu trữ ký tự (1 byte)
int
lưu trữ số nguyên (4 byte)
double
lưu trữ số thực (8 byte)

Kiểu string

- Kiểu string có trong thư viện string. Độ dài chuỗi tùy ý.

```
#include <string>
```

Toán tử và kiểu dữ liệu

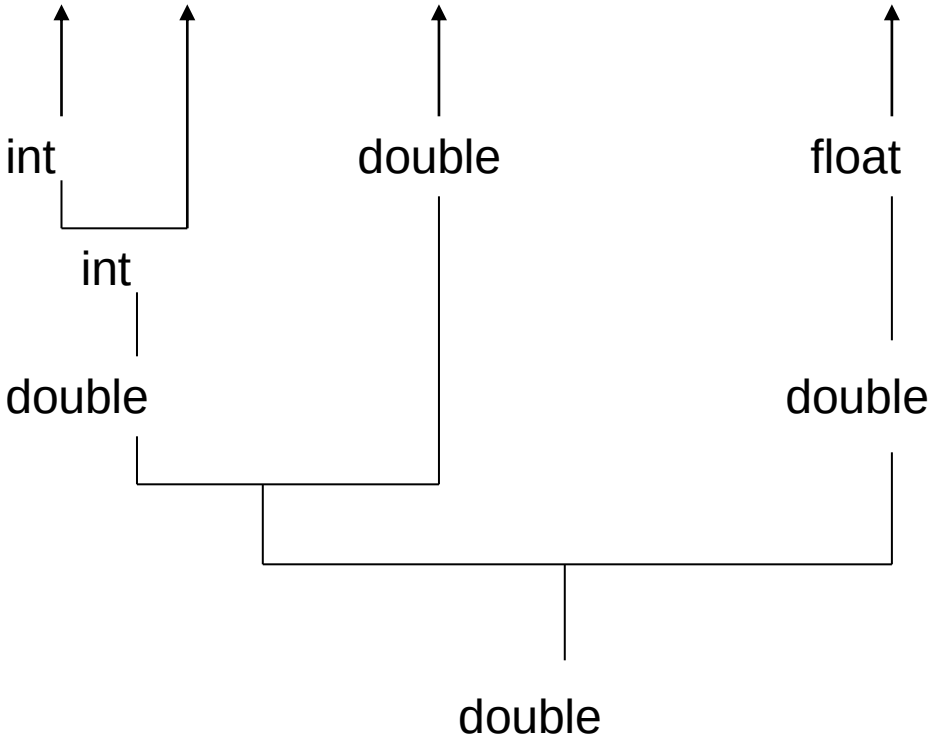
	bool	char	int	double	string
gán	=	=	=	=	=
cộng			+	+	
nối chuỗi					+
trừ			-	-	
nhân			*	*	
chia			/	/	
chia lấy dư			%		
tăng 1			++	++	
giảm 1			--	--	
tăng n			+=n	+=n	
thêm vào cuối					+=
giảm n			-=n	-=n	

Biểu thức

- Chuyển đổi kiểu trong biểu thức
  - Khi các hằng và biến chứa trong biểu thức có các kiểu khác nhau. Trình biên dịch sẽ thực hiện chuyển thành cùng kiểu của toán hạng có kiểu lớn nhất, trước khi các phép toán được thực hiện

Ví dụ:

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



Một số hàm toán học

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

```
#include <cmath>
```

Các hàm trong thư viện cmath

Hàm	Công dụng	Kiểu tham số	
abs(x)	$\|x\|$	int (double)	int (double)
exp(x)	$e^x, e = 2.718 \ldots$	double	double
pow(x, y)	x^y	double	double
sqrt(x)	$\sqrt{x}$	double	double
sin(x)	sinx, x: radian	double	double
cos(x)	cosx, x: radian	double	double
tan(x)	tanx, x: radian	double	double
log(n)	ln(x) , x>0	double	double
log10(x)	$log_{10}(x), x > 0$	double	double

Thứ tự ưu tiên

Thứ tự ưu tiên:

!

< <= >= >

== !=

&&

||

- Biểu thức luận lý được sử dụng trong các câu lệnh rẽ nhánh if, if...else và lặp for, while, do...while.

Biến

- Cú pháp khai báo biến:

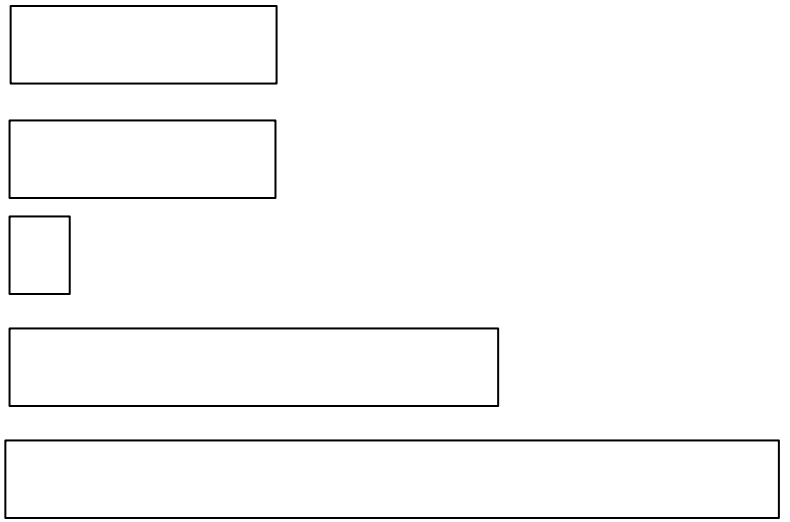
  type <variable_list>;

  – type: kiểu dữ liệu

  – variable_list: một hoặc nhiều danh hiệu (identifier), cách nhau bởi dấu phẩy.

- Ví dụ:

```
int a, b;
char c;
double x;
string s;
```



Biến

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

```
type <variable_name = value>;
```

- Ví dụ:

int a = 7, b = 9;	a	7
	b	9
char c = ‘a’;	c	a
double x = 1.2;	x	1.2
string s = “Hello, world”;	s	Hello, world

Hằng

- *Hằng* (*named constant*) là nơi lưu trữ giá trị không thay đổi trong suốt thời gian chương trình thực thi.
- Cú pháp định nghĩa hằng:

```
const <type constant_name = value>;
```

  - type: kiểu dữ liệu
  - constant_name: là danh hiệu, thường dùng chữ in hoa.
  - value: giá trị phù hợp với kiểu dữ liệu.

❑ Biến

Lưu ý khi sử dụng biến:

- Phải khai báo biến trước khi sử dụng.
- Tên của những biến khác nhau thì khác nhau.
- Dung lượng bộ nhớ dành cho các biến khác nhau tuỳ thuộc vào kiểu dữ liệu của biến đó.

❑ Cách đặt tên

- Tên là một dãy các ký tự bao gồm chữ cái, chữ số hoặc gạch nối. Ký tự đầu tiên của tên phải là chữ cái hoặc gạch nối.
- Tên không trùng với các từ khoá.
- Tên là duy nhất.
- Phân biệt chữ hoa, thường.
- Đặt tên gợi nhớ, không quá dài.

Từ khóa

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

Ví dụ: **`int, double, char, const,`** …

- Tên/ danh hiệu hay định danh (*identifier*): là tên biến, tên hằng, tên hàm và tên các đối tượng do người dùng định nghĩa.

Nhập dữ liệu

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

```
cin >> variable_name1 >> variable_name2… ;
```

Ví dụ:

```
cin >> banKinh;
```

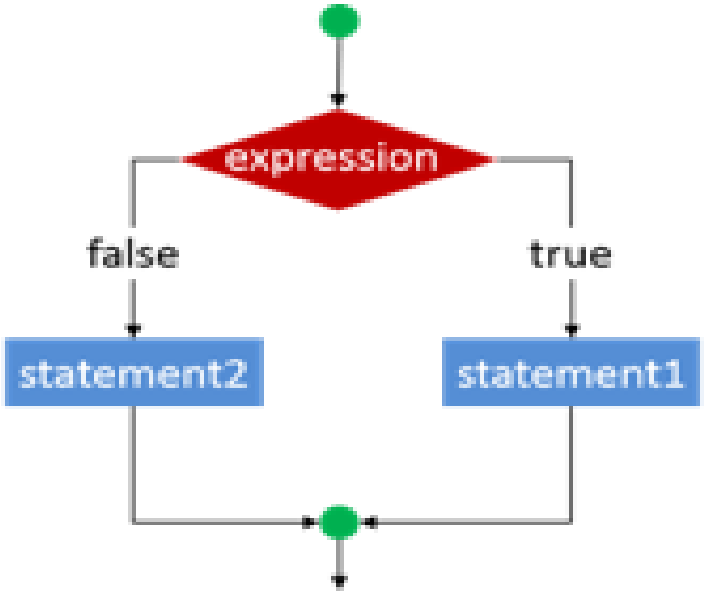
Xuất dữ liệu

- Câu lệnh xuất????

Câu lệnh if...else

- Cú pháp câu lệnh if-else:

```
if(expression)
    statement1;
[else
    statement2;]
```



Trong đó:

- expression: biểu thức.
- statement1, statement2: câu lệnh rỗng, câu lệnh đơn hoặc khối lệnh.

Câu lệnh if...else

- ***Lỗi thường gặp***
  - Sử dụng phép gán (=) thay cho phép so sánh bằng nhau (==)
  - Gõ dấu chấm phẩy ngay sau biểu thức điều kiện
  - **Sau else có biểu thức điều kiện**

```
if ( a > = b)
        cout << a << " lon hon hoac bang " << b;
else (a < b)
        cout << a << " nho hon " << b;
```

Câu lệnh if…else lồng nhau

- **Cú pháp câu lệnh if-else-if (nested if-else):**

```
if (expression)
        statement;
else
        if (expression)
                statement;
        else
                if (expression)
                        statement;
                ...
                        else
                                statement;
```

Bài tập

Cho biết kết quả đoạn chương trình sau:

```
int a = 1024, b = 2;
if ( a % b == 0 )
      a += 3;
else
      a -= 1;
cout << a << " va "<< b;
```

1027 va 2

Bài tập

Cho biết kết quả đoạn chương trình sau:

```
int a = 1024, b = 2;
if ( a % b == 0 )
      a += 3;
if ( a % b == 1 )
      a -= 1;
cout << a << " va "<< b;
```

1026 va 2

Bài tập

Cho biết kết quả đoạn chương trình sau:

```
int a = 1024, b = 2;
if ( a % b == 1 );
      a += 3;
cout << a << " va "<< b;
```

1027 va 2

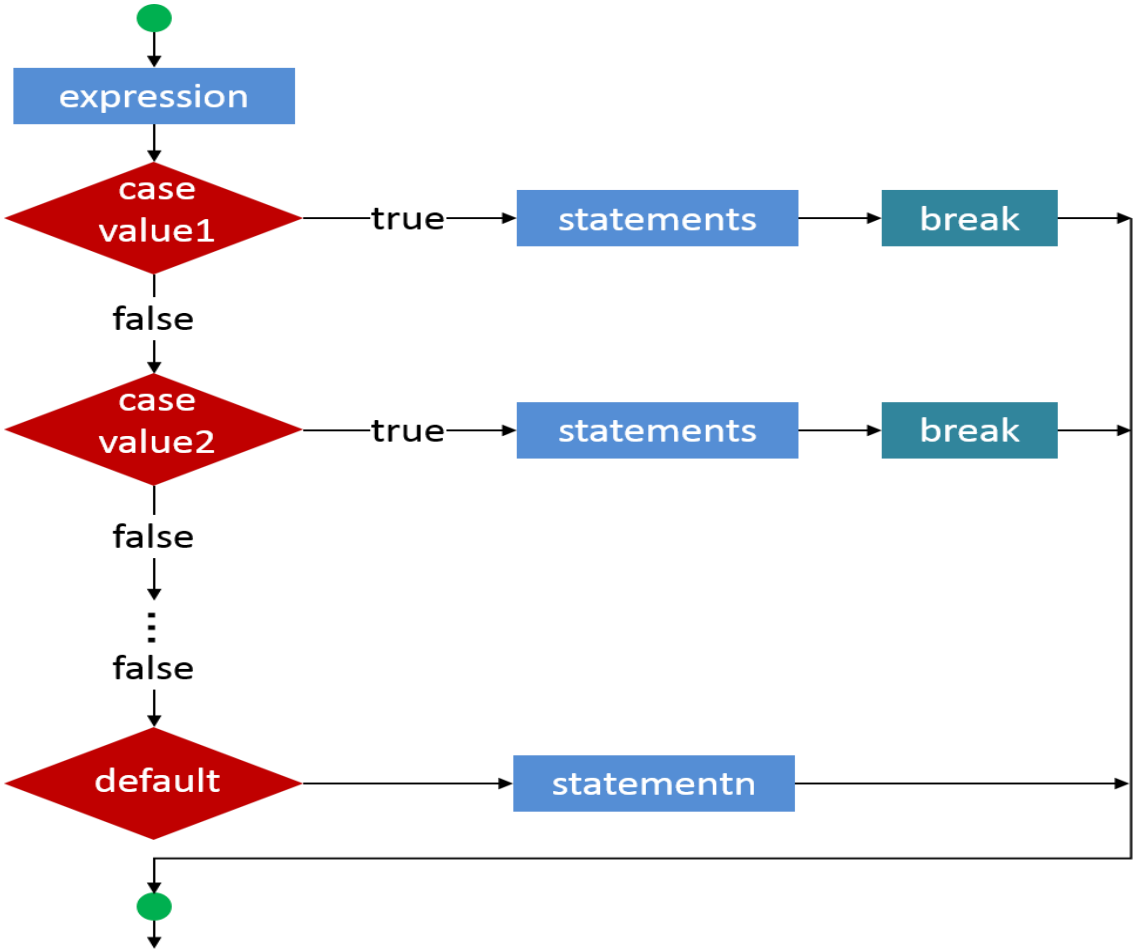
Câu lệnh switch

- **Cú pháp câu lệnh switch:**

```
switch(expression)
{
   case value 1:
      statement sequence          (S1)          //câu lệnh tuần tự
      break;
   case value 2:
      statement sequence          (S2)          //câu lệnh tuần tự
      break;
      ...
  case value n:
      statement sequence          (Sn)          //câu lệnh tuần tự
      break;
  [default:
       statement sequence         (S) ]         //câu lệnh tuần tự

}
```

Câu lệnh switch



Hình 4: Câu lệnh switch

LƯU Ý

- Không dùng câu lệnh switch với kiểu string.
- Không dùng một giá trị cho hai nhãn case.
- Có thể dùng nhiều nhãn case cho một trường hợp.
- Không bắt buộc phải có default
- Thông thường trong mỗi case có lệnh break để thoát khỏi cấu trúc switch.

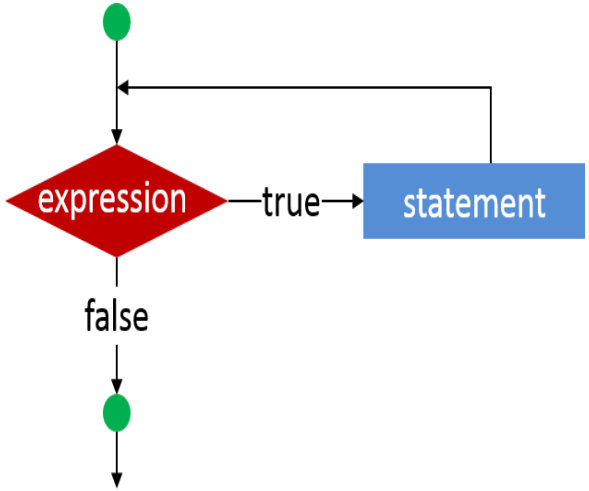
Câu lệnh while

- Cú pháp câu lệnh while:

```
while(condition)
    statement;
```

Trong đó:

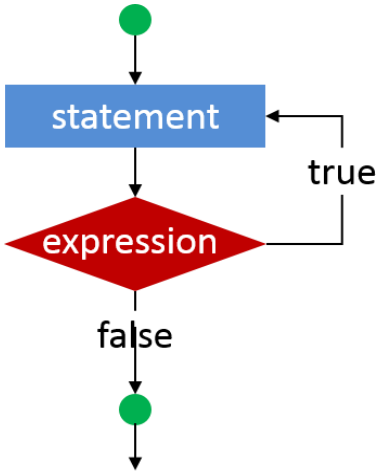
- condition = true (khác 0) hoặc false (0).
- statement: câu lệnh rỗng, câu lệnh đơn hoặc khối lệnh
- Nếu condition = true thì statement được thực hiện, ngược lại statement không thực hiện



Câu lệnh do-while

- Cú pháp câu lệnh do-while:

```
do {
    statement;
}while(condition);
```



- Thực hiện câu lệnh cho đến khi điều kiện lặp có giá trị false.

LƯU Ý

- while và do..while đều có thể lặp vô tận.
- while có thể không xảy ra lần lặp nào, nhưng do..while ít nhất phải có 1 lần lặp.

Câu lệnh for

- Cú pháp câu lệnh for:

```
for(init statement; condition; expression)
    statement;
```

Trong đó:

init statement: khởi tạo giá trị cho biến điều khiển lặp.

condition: biểu thức logic xác định khi nào lặp.

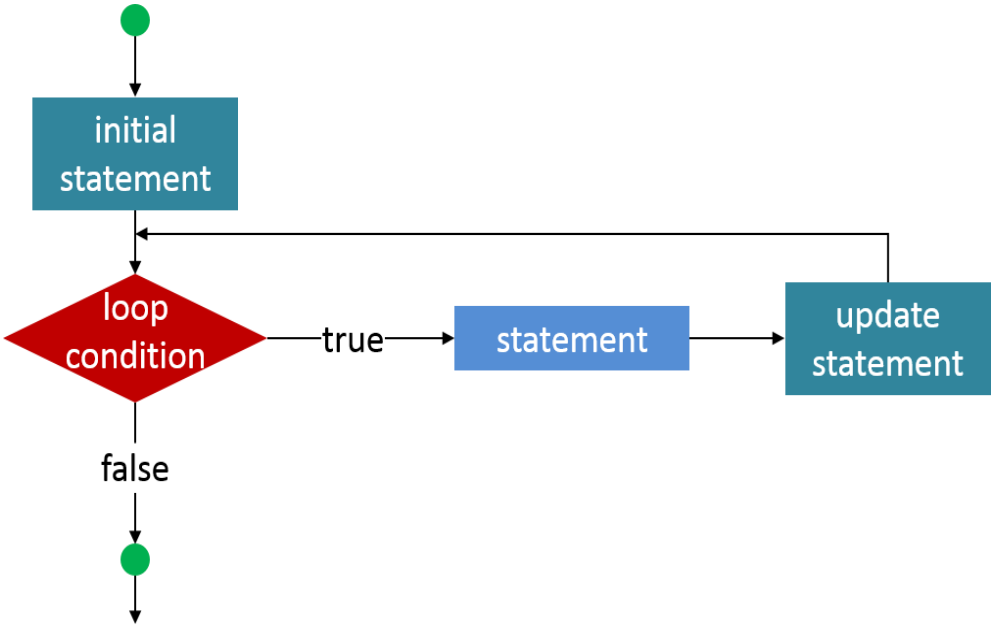
expression: cập nhật biến điều khiển lặp.

statement: câu lệnh rỗng, câu lệnh đơn hoặc khối lệnh.

- Lưu ý dấu chấm phẩy (;) phân cách ba phần.

Câu lệnh for

- Lưu đồ:



1. Khởi tạo giá trị cho biến điều khiển lặp.
2. Kiểm tra điều kiện lặp. Nếu điều kiện lặp có giá trị true:
   - Thực hiện các câu lệnh trong thân vòng lặp.
   - Thực hiện câu lệnh thay đổi biến điều khiển lặp.
3. Lặp lại bước 2 cho đến khi điều kiện lặp có giá trị false

Câu lệnh break

- Câu lệnh break dùng để bỏ qua phần còn lại trong câu lệnh switch.
- Câu lệnh break khi được thực hiện bên trong vòng lặp for, while, do-while thì sẽ thoát khỏi vòng lặp ngay lập tức.
- Ví dụ:

```
for (int count = 1; count <= 10; count++)
{
     if (count == 5)
          break;
     cout << count << " ";
}
cout << "\nNgung lap khi count = " << count << endl;
```

Câu lệnh continue

- Câu lệnh continue dùng để bỏ qua phần còn lại trong vòng lặp for, while, do-while và bắt đầu lần lặp kế tiếp.
- Ví dụ:

```
for (int count = 1; count <= 10; count++)
{
      if (count == 5)
            continue;
      cout << count << " ";
}
cout << "\n(khong in 5)" << endl;
```

MẢNG

1. Mảng
2. Các thao tác trên mảng
3. Một số lưu ý khi làm việc với mảng
4. Truyền mảng cho hàm

MẢNG

1. Mảng
   - Khái niệm
   - Khai báo
   - Khởi tạo
   - Truy xuất
2. Các thao tác trên mảng
3. Một số lưu ý khi làm việc với mảng
4. Truyền mảng cho hàm

MẢNG

1. Mảng
   - Khái niệm
   - Khai báo
   - Khởi tạo
   - Truy xuất
2. Các thao tác trên mảng
3. Một số lưu ý khi làm việc với mảng
4. Truyền mảng cho hàm

Khái niệm

- **Mảng/dãy** (*array*) là một tập hợp gồm các phần tử có *cùng kiểu dữ liệu*, được lưu trữ tại các *vị trí liên tục* trong bộ nhớ.
  - Mảng 1 chiều (one – dimensional array): là mảng có các phần tử được sắp xếp theo dạng danh sách.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
list	**35**	**12**	**33**	**21**	**9**	**0**	**-5**

  - Mảng 2 chiều (two – dimensional array): là mảng có các phần tử được sắp xếp theo dạng bảng (table).

table	[0]	[1]	[2]
[0]	21	4	-5
[1]	3	6	9
[2]	8	11	36

MẢNG

1. Mảng
   - Khái niệm
   - Khai báo
   - Khởi tạo
   - Truy xuất
2. Các thao tác trên mảng
3. Một số lưu ý khi làm việc với mảng
4. Truyền mảng cho hàm

Khai báo

Cú pháp khai báo mảng 1 chiều:

```
dataType arrayName [numberOfElements];
```

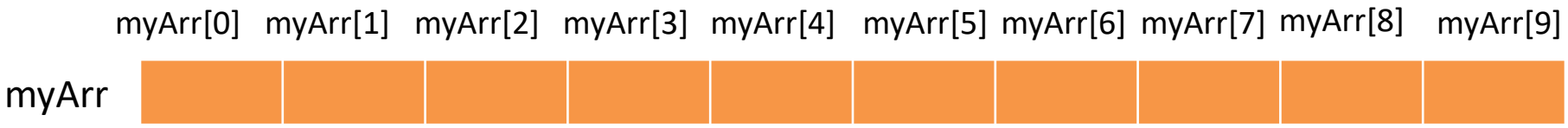
- dataType: kiểu dữ liệu của mỗi phần tử trong mảng.
- arrayName: tên của mảng
- numberOfElements: là 1 số nguyên, cho biết số phần tử tối đa của mảng.

VD: `int myArr [10];`

Lưu ý khi khai báo

- *Số phần tử* được *xác định* khi khai báo và *không thay đổi*.
- Các phần tử được đánh số từ `0` đến `numberOfElements - 1`
- Khi khai báo phải biết trước số lượng phần tử tối đa của mảng.

VD: `int myArr [10];`



Kích thước bộ nhớ: = 4 byte x 10 = 40 byte

Lưu ý khi khai báo

- Khai báo gây lỗi:

```
int arrSize;
cout << "Nhap so phan tu cua mang: ";
cin >> arrSize;
int myArr[arrSize];
```

- Định nghĩa số phần tử của mảng là 1 hằng số:

```
const int MAX = 20;
int num[MAX];
```

- Định nghĩa kiểu dữ liệu là mảng:

```
const int MAX = 20;
typedef double List[MAX];
List myarr;
List yourarr;
```

Tương đương với:

```
double myarr[20];
double yourarr[20];
```

MẢNG

1. Mảng
   - Khái niệm
   - Khai báo
   - Khởi tạo
   - Truy xuất
2. Các thao tác trên mảng
3. Một số lưu ý khi làm việc với mảng
4. Truyền mảng cho hàm

Khởi tạo

- ***Khởi tạo khi khai báo:***

```
dataType arrayName [numberOfElements] = {initialValues};
```

- *initialValues*: các giá trị phân cách bằng dấu phẩy (,)

Ví dụ:

```
int a[3] = {2, 12, 1};
```

Tương đương:

```
int  a[3];
a[0] = 2;
a[1] = 12;
a[2] = 1;
```

Một số trường hợp khởi tạo

```
double a[] = {5.1, 2.2, 5.3, 7.9, 10};
```

Tương đương với:

```
double a[5] = {5.1, 2.2, 5.3, 7.9, 10};
```

```
int List[10] = {0}; //khởi tạo tất cả phần tử là 0
```

```
double a[5] = {5.1, 2.2, 5.3};
```

Tương đương với:

```
double a[5] = {5.1, 2.2, 5.3, 0, 0};
```

MẢNG

1. Mảng
   - Khái niệm
   - Khai báo
   - Khởi tạo
   - Truy xuất
2. Các thao tác trên mảng
3. Một số lưu ý khi làm việc với mảng
4. Truyền mảng cho hàm

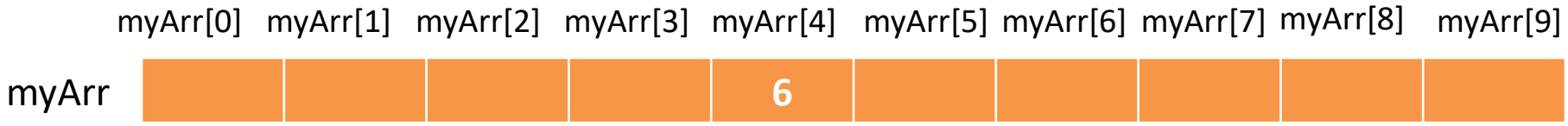
Truy xuất

Cú pháp gán dữ liệu cho một phần tử trong mảng:

```
arrayName [subscript] = expression;
```

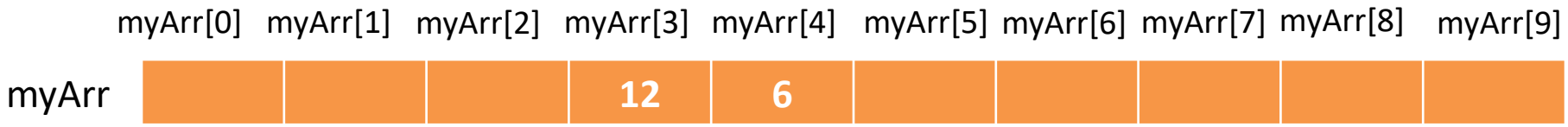
- subscript: chỉ số là số nguyên hoặc biểu thức có giá trị là số nguyên. (chỉ số có giá trị từ *0* đến *số lượng phần tử - 1*)
- expression: biểu thức có kiểu phù hợp với kiểu dữ liệu của mảng.

Ví dụ: `int myArr[10]; myArr[4] = 6;`



Truy xuất

```
int  i = 3;
myArr[i] = 12;
```




```
myArr[6] = myArr[3] + myArr[4];
```

MẢNG

1. Mảng
2. Các thao tác trên mảng
   - Nhập/xuất
   - Tính tổng và giá trị trung bình
   - Tìm kiếm
   - Tìm giá trị nhỏ nhất, giá trị lớn nhất
3. Một số lưu ý khi làm việc với mảng
4. Truyền mảng cho hàm

MẢNG

1. Mảng
2. Các thao tác trên mảng
   - Nhập/xuất
   - Tính tổng và giá trị trung bình
   - Tìm kiếm
   - Tìm giá trị nhỏ nhất, giá trị lớn nhất
3. Một số lưu ý khi làm việc với mảng
4. Truyền mảng cho hàm

Nhập/xuất

- Các kiểu nhập mảng:
  - Khởi tạo tất cả các phần tử của mảng đều là 0, hoặc khởi tạo bằng giá trị cụ thể.
  - Yêu cầu người dùng nhập vào đủ số lượng giá trị mà mảng có thể lưu trữ tối đa.
  - Yêu cầu người dùng nhập số lượng phần tử cần dùng. Sau đó kiểm tra xem có <= giá trị tối đa lưu trữ ?

Nhập/xuất

- Khởi tạo tất cả các phần tử của mảng đều là 0, hoặc khởi tạo bằng giá trị cụ thể.

```
const int MAXSIZE = 10;
double a[MAXSIZE];
for (int i = 0; i < MAXSIZE; i++)
      a[i] = 0;
```

// có thể thay thế bằng: double a[MAXSIZE] = {0};

Nhập/xuất

- Yêu cầu người dùng nhập vào đủ số lượng giá trị mà mảng có thể lưu trữ tối đa.

```
const int MAXSIZE = 10;
double a[MAXSIZE];
cout << "Nhap  " << MAXSIZE << " so double: ";
for (int i = 0; i < MAXSIZE; i++)
        cin >> a[i];
```

Nhập/xuất

- Yêu cầu người dùng nhập số lượng phần tử cần dùng. Sau đó kiểm tra xem có <= giá trị tối đa lưu trữ ?

```
const int MAXSIZE = 10;
double a[MAXSIZE];
int n;
cout << "Ban can nhap bao nhieu phan tu: ";
cin >> n;
if ( n  <= MAXSIZE)
{
      for (int i = 0; i < n; i++)
      {
          cout << "\nNhap gia tri phan tu thu " << i << ": ";
          cin >> a[i];
      }
}
else
      cout << "\nMang chi co the luu tru toi da " << MAXSIZE << "
phan tu." << endl;
```

Nhập/xuất

– Xuất mảng:

```
cout << "Cac phan tu trong mang hien  tai la: ";
for (int i = 0; i < n; i++)
      cout << a[i] << " ";
```

Lưu ý với số lượng phần tử

- Nếu dùng mảng kiểu khởi tạo giá trị ban đầu thì có thể tính số lượng phần tử n của mảng a:

```
int n = sizeof(a)/sizeof(a[0]);
```

Ví dụ:

```
double a[] = {0,5,6, 8,9};
int x = sizeof(a)/sizeof(a[0]);
cout << x << endl;
```

//kết quả: 5

MẢNG

1. Mảng
2. Các thao tác trên mảng
   - Nhập/xuất
   - Tính tổng và giá trị trung bình
   - Tìm kiếm
   - Tìm giá trị nhỏ nhất, giá trị lớn nhất
3. Một số lưu ý khi làm việc với mảng
4. Truyền mảng cho hàm

Tính tổng

- Ý tưởng tính tổng các phần tử trong mảng: duyệt từ đầu đến cuối mảng (i = 0 đến số phần tử - 1), mỗi lần duyệt cộng dồn giá trị của phần tử thứ i trong mảng a vào giá trị tổng.

```
double a[MAXSIZE];
double tong = 0;
//Nhap mang voi so luong pt la n
for ( int i = 0; i < n ; i++)
     tong += a[i];
cout << "\nTong cac phan  tu la = " << tong << endl;
```

Tính giá trị trung bình

- Ý tưởng tính trung bình các phần tử trong mảng: duyệt từ đầu đến cuối mảng (i = 0 đến số phần tử - 1), mỗi lần duyệt cộng đồn giá trị của phần tử thứ i trong mảng a vào giá trị tổng. Sau đó lấy giá trị tổng vừa tìm được chia cho số lượng phần tử của mảng.

```
double a[MAXSIZE];
double tong = 0;
//Nhap mang voi so luong pt la n
for ( int i = 0; i < n ; i++)
     tong += a[i];
cout << "\nTrung binh cong cac phan  tu la = " << tong/n
<< endl;
```

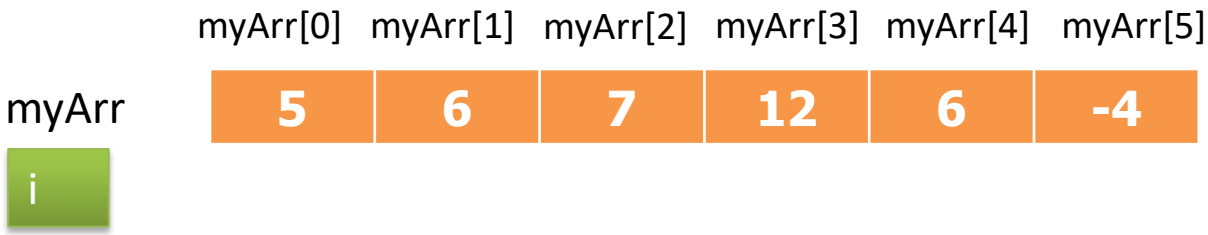
MẢNG

1. Mảng
2. Các thao tác trên mảng
   - Nhập/xuất
   - Tính tổng và giá trị trung bình
   - Tìm kiếm
   - Tìm giá trị nhỏ nhất, giá trị lớn nhất
3. Một số lưu ý khi làm việc với mảng
4. Truyền mảng cho hàm

Tìm kiếm

- **Bài toán tìm kiếm: cho mảng gồm n phần tử. Mỗi phần tử đã được nhận giá trị cụ thể. Tìm xem trong mảng đó xem giá trị x có tồn tại không?**
- Ý tưởng:
  - Duyệt từ đầu đến cuối mảng (chỉ số i = 0 đến số phần tử - 1)
  - Kiểm tra xem giá trị của mảng a tại từng vị trí thứ i có bằng với x hay không?
    - Nếu có: tìm kiếm thành công. Có thể trả về vị trí tồn tại. (dừng tìm)
    - Nếu không: tìm kiếm không thành công.
  - Dựa trên giá trị tồn tại, kiểm tra xem giá trị biến i sau khi kết thúc vòng lặp có <= số phần tử - 1 hay không?
    - Nếu có: kết luận x có tồn tại trong mảng.
    - Nếu không (đã đi đến hết vòng lặp vẫn không tìm thấy a[i] == x) thì kết luận x không tồn tại trong mảng.

Tìm kiếm

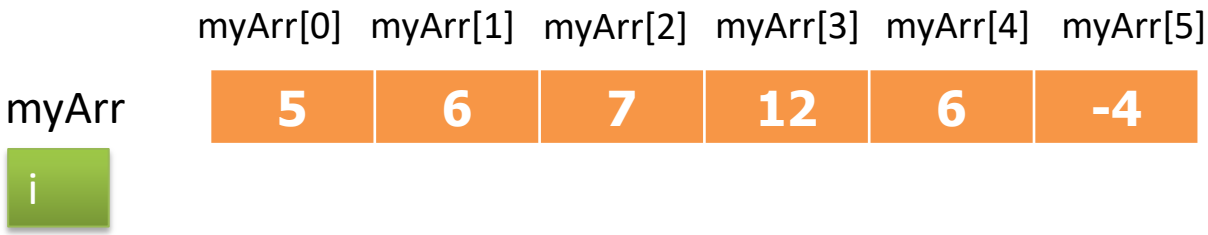


x = 13

i = 0 (i < 6) → myArr[0] != x → i++
i = 1 (i < 6) → myArr[1] != x → i++
i = 2 (i < 6) → myArr[2] != x → i++
i = 3 (i < 6) → myArr[3] != x → i++
i = 4 (i < 6) → myArr[4] != x → i++
i = 5 (i < 6) → myArr[5] != x → i++
i = 6 (i == 6) thoát khỏi vòng lặp

Kết thúc i == n nên x không tồn tại trong mảng

Tìm kiếm



i = 0 (i < 6) → myArr[0] != x → i++
i = 1 (i < 6) → myArr[1] != x → i++
i = 2 (i < 6) → myArr[2] == x → thoát khỏi vòng lặp

Kết thúc i < n nên x tồn tại trong mảng

Tìm kiếm

```
int a[MAXSIZE];
int i, int x;
//Nhap mang voi so luong pt la n
//Nhap gia tri can tim kiem x
for (i = 0; i < n ; i++)
     if ( a[i] == x)
          break;
if (i < n)
     cout << x << " co ton tai" << endl;
else
     cout << x << " khong ton tai" << endl;
```

Tìm kiếm

Vấn đề cần giải quyết:

- **Nếu cần kiểm tra tần số xuất hiện của x trong mảng nếu có tồn tại ???**

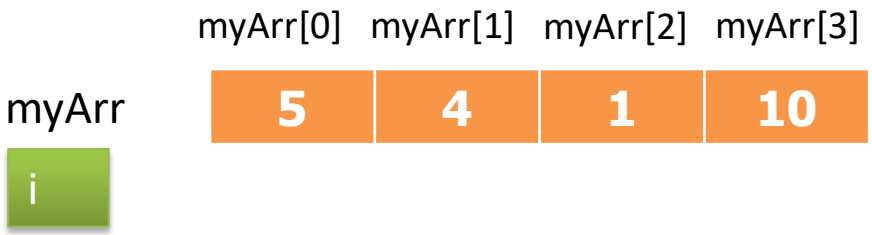
MẢNG

1. Mảng
2. Các thao tác trên mảng
   - Nhập/xuất
   - Tính tổng và giá trị trung bình
   - Tìm kiếm
   - Tìm giá trị nhỏ nhất, giá trị lớn nhất
3. Một số lưu ý khi làm việc với mảng
4. Truyền mảng cho hàm

Giá trị nhỏ nhất

- Ý tưởng:
  - Gán giá trị min = giá trị phần tử đầu tiên của mảng.
  - Duyệt từ đầu đến cuối mảng. Mỗi lần duyệt kiểm tra xem *min có lớn hơn giá trị phần tử đang xét* không?
    - Nếu *có*: cập nhật min là giá trị phần tử đang xét.
    - Nếu *không*: xét phần tử tiếp theo.
  - Kết thúc việc duyệt: xuất giá trị min

Tìm kiếm



min = 5

i = 1 (i < 4) → min > myArr[1] → min = 4, i++
i = 2 (i < 4) → min > myArr[2] → min = 1, i++
i = 3 (i < 4) → min < myArr[3] → i++
I = 4 (i==4) → kết thúc vòng lặp

Kết thúc min = 1

Giá trị nhỏ nhất

```
int a[100]; int min;
//Nhap mang voi so luong pt la n
min = a[0];
for ( int i = 1; i < n ; i++)
    if ( min > a[i])
        min = a[i];
cout << "\Gia tri nho nhat la = " << min <<
endl;
```

Giá trị lớn nhất

- Ý tưởng:
  - Gán giá trị max = giá trị phần tử đầu tiên của mảng.
  - Duyệt từ đầu đến cuối mảng. Mỗi lần duyệt kiểm tra xem *max có nhỏ hơn giá trị phần tử đang xét* không?
    - Nếu *có*: cập nhật max là giá trị phần tử đang xét.
    - Nếu *không*: xét phần tử tiếp theo.
  - Kết thúc việc duyệt: xuất giá trị max

Giá trị lớn nhất

```
int a[100]; int max;
//Nhap mang voi so luong pt la n
max = a[0];
for ( int i = 1; i < n ; i++)
    if ( max < a[i])
        max = a[i];
cout << "\Gia tri lon nhat la = " << max
<< endl;
```

MẢNG

1. Mảng
2. Các thao tác trên mảng
3. Một số lưu ý khi làm việc với mảng
4. Truyền mảng cho hàm

Một số lưu ý khi làm việc với mảng

- Không truy xuất phần tử ngoài mảng.
- Không dùng phép gán 1 mảng cho 1 mảng khác.

```
int a[5] = {0}; int  b[5]; b = a; //lỗi
```

- Không dùng cin/cout với biến mảng.

```
int a[5]; cin >> a; //lỗi
```

- Không dùng phép so sánh với biến mảng.

```
int a[5] = {0}; int  b[5]; if (b < a) …; //lỗi
```

- Dùng vòng lặp để sao chép các giá trị của mảng.

```
for (int i = 0; i < n; i++)
        b[i] = a[i];
```

- Dùng vòng lặp để nhập/xuất dữ liệu cho mảng

Bài tập

1. Viết chương trình nhập vào một mảng số nguyên tối đa 50 phần tử. Đếm xem trong mảng có bao nhiêu phần tử có giá trị là số chẳn.
2. Viết chương trình nhập vào một mảng số nguyên tối đa 30 phần tử. Cho phép người dùng nhập vào vị trí 2 phần tử muốn hoán đổi giá trị với nhau. Thực hiện hoán đổi và xuất lại mảng sau khi đã hoán đổi xong.

MẢNG

1. Mảng
2. Các thao tác trên mảng
3. Một số lưu ý khi làm việc với mảng
4. Truyền mảng cho hàm

Truyền mảng cho hàm

- Biến mảng truyền cho hàm bằng **tham chiếu**.
  - *Không* dùng kí tự *&* khi *khai báo* tham số là *kiểu mảng*.
- Khi truyền tham số là mảng 1 chiều:
  - *Không* cần khai báo số phần tử tối đa của mảng.
  - *Phải* có tham số cho biết số phần tử hiện tại của mảng.
- Khi truyền tham chiếu, hàm có thể thay đổi giá trị của tham số.
- Nên truyền tham chiếu hằng để tránh bị thay đổi giá trị tham số mảng.
- C++ không cho phép hàm trả về kiểu giá trị có kiểu mảng.

Truyền mảng cho hàm

- Hàm khởi tạo giá trị ban đầu cho mảng:

```
void khoiTao (int a[], int n)
{
      for (int i = 0; i < n; i++)
            a[i] = 0;
}
```

Truyền mảng cho hàm

- Hàm xuất các phần tử của mảng truyền giá trị tham số mảng kiểu tham chiếu hằng:

```
void xuat (const int a[], int n)
{
	for (int i = 0; i < n; i++)
		cout << a[i] << " ";
	cout << endl;
}
```

Truyền mảng cho hàm

- Hàm tính toán trên mảng:

```
double tinhtrungbinh (int a[], int n)
{
	int tong = 0;
	for (int i = 0; i < n; i++)
		tong += a[i];
	return tong*1.0/n;
}
```

Truyền mảng cho hàm

- Hàm tính toán trên mảng:

```
int main()
{
        //khai báo, nhập mảng arr với z phần tử
        cout << "Trung binh = " << tinhtrungbinh(arr, z)
<< endl;
}
```