BAN HỌC TẬP

UIT
Trường Đại học
Công nghệ Thông tin

Khoa Khoa học
và Kỹ thuật Thông tin

Nhập môn lập trình

Trainer: - Phạm Quốc Cường (CNCL2020)
- Nguyễn Thiện Thuật (KHDL2020)
- Đặng Thị Thúy Hồng (KHDL 2020)

Nội dung Training

1. Thuật toán và lưu đồ
2. Kiểu dữ liệu cơ bản và phép toán
3. Cấu trúc điều khiển
   - **Cấu trúc rẽ nhánh**
   - **Câu lệnh lặp**
4. Hàm
5. Cấu trúc đề thi, giải đề thi
6. Một số lỗi thường gặp

BAN HỌC TẬP

Thuật toán và lưu đồ

Thuật toán

Thuật toán là gì?

Là tập hợp (dãy) hữu hạn các chỉ thị (hành động) được định nghĩa rõ ràng nhằm giải quyết một bài toán cụ thể nào đó.

Các phương pháp biểu diễn thuật toán

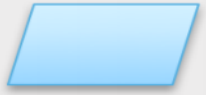
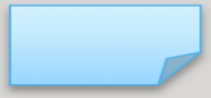
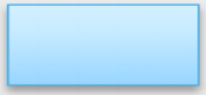
- Dùng ngôn ngữ tự nhiên
- Dùng lưu đồ - sơ đồ khối (flowchart)
- Dùng mã giả (pseudocode)

Thuật toán

Dùng lưu đồ - sơ đồ khối

- Là một công cụ trực quan để diễn đạt các thuật toán
- Biểu diễn thuật toán bằng lưu đồ sẽ giúp người đọc theo dõi được sự phân cấp các trường hợp và quá trình xử lý của thuật toán
- Phương pháp lưu đồ thường được dùng trong những thuật toán có tính rắc rối, khó theo dõi được quá trình xử lý

Thuật toán

STT	Ký hiệu	Ý nghĩa
1		Bắt đầu / Kết thúc chương trình
2		Điều kiện rẽ nhánh (lựa chọn)
3		Luồng xử lý
4		Nhập
5		Xuất
6		Xử lý / tính toán / gán
7		Trả về giá trị (return)
8		Điểm nối liên kết

BAN HỌC TẬP

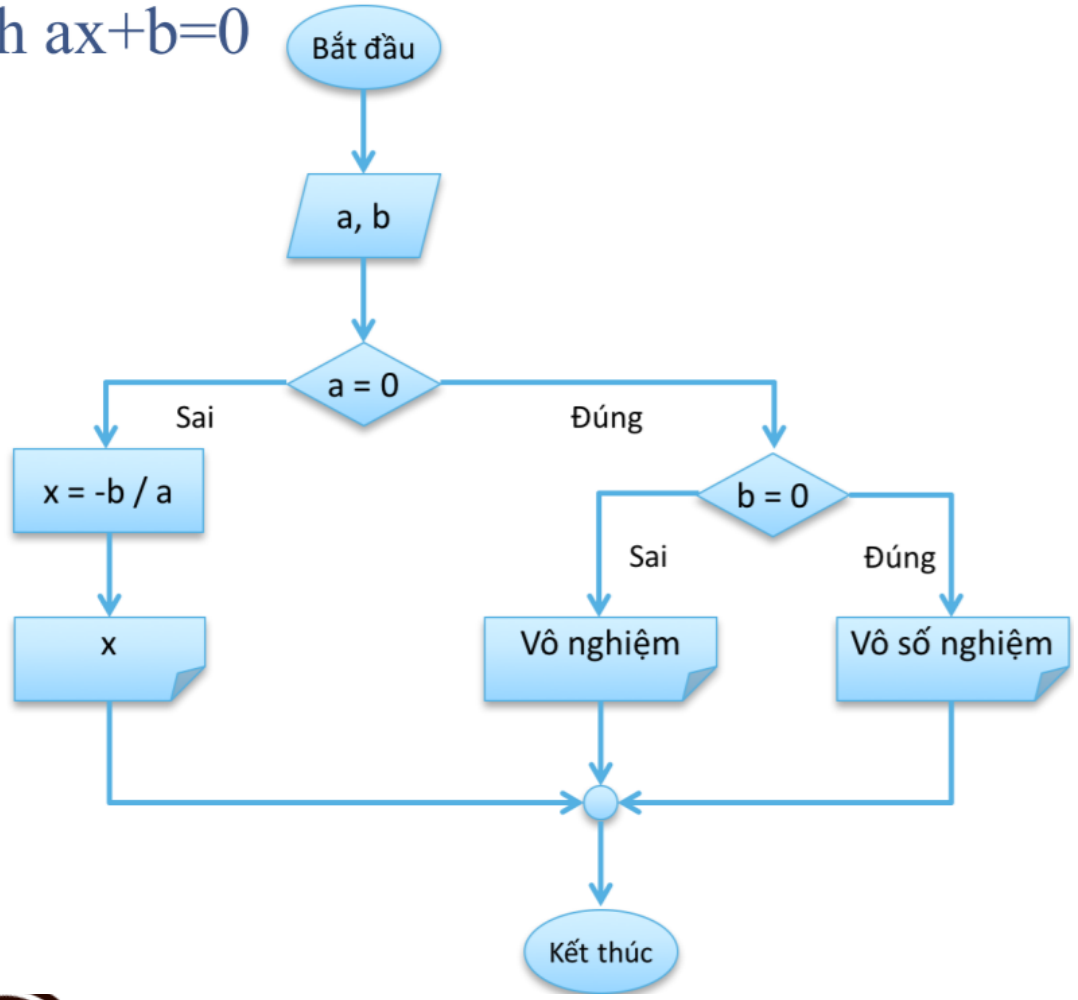
Thuật toán

Ví dụ: Thuật toán giải phương trình bậc nhất: ax + b = 0 (a, b là số thực)

Đầu vào: a,b thuộc R
Đầu ra: nghiệm phương trình ax + b = 0

Thuật toán

Giải phương trình ax+b=0



BAN HỌC TẬP

Thuật toán

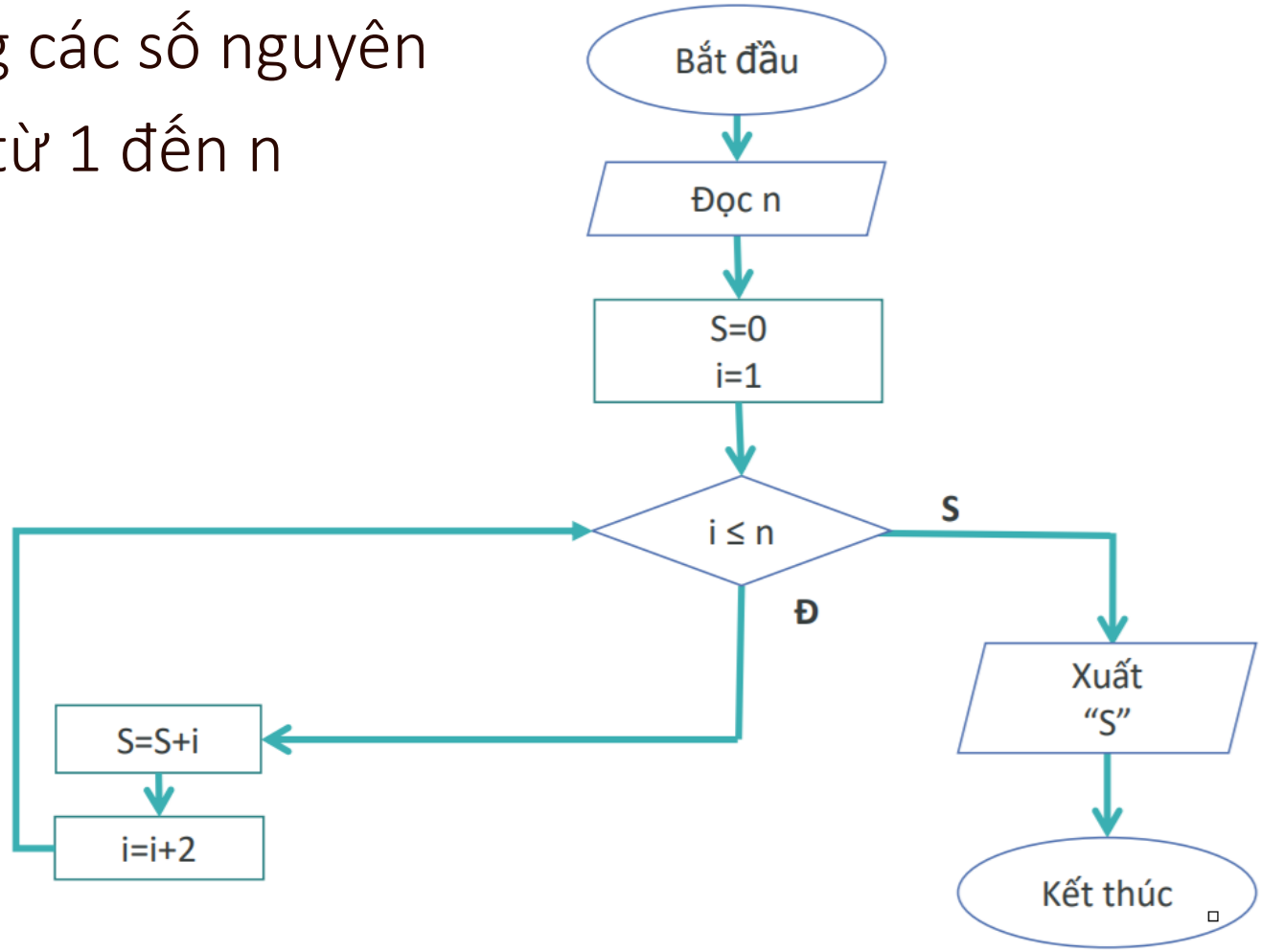
Một số ví dụ về thuật toán

- VD1: Vẽ lưu đồ thuật toán Tính tổng các số nguyên dương lẻ từ 1 đến n
- VD2: Vẽ lưu đồ thuật toán Liệt kê tất cả các ước số của số nguyên dương n
- Extra: Vẽ lưu đồ thuật toán Tìm nghiệm của phương trình bậc hai một ẩn

BAN HỌC TẬP

Thuật toán

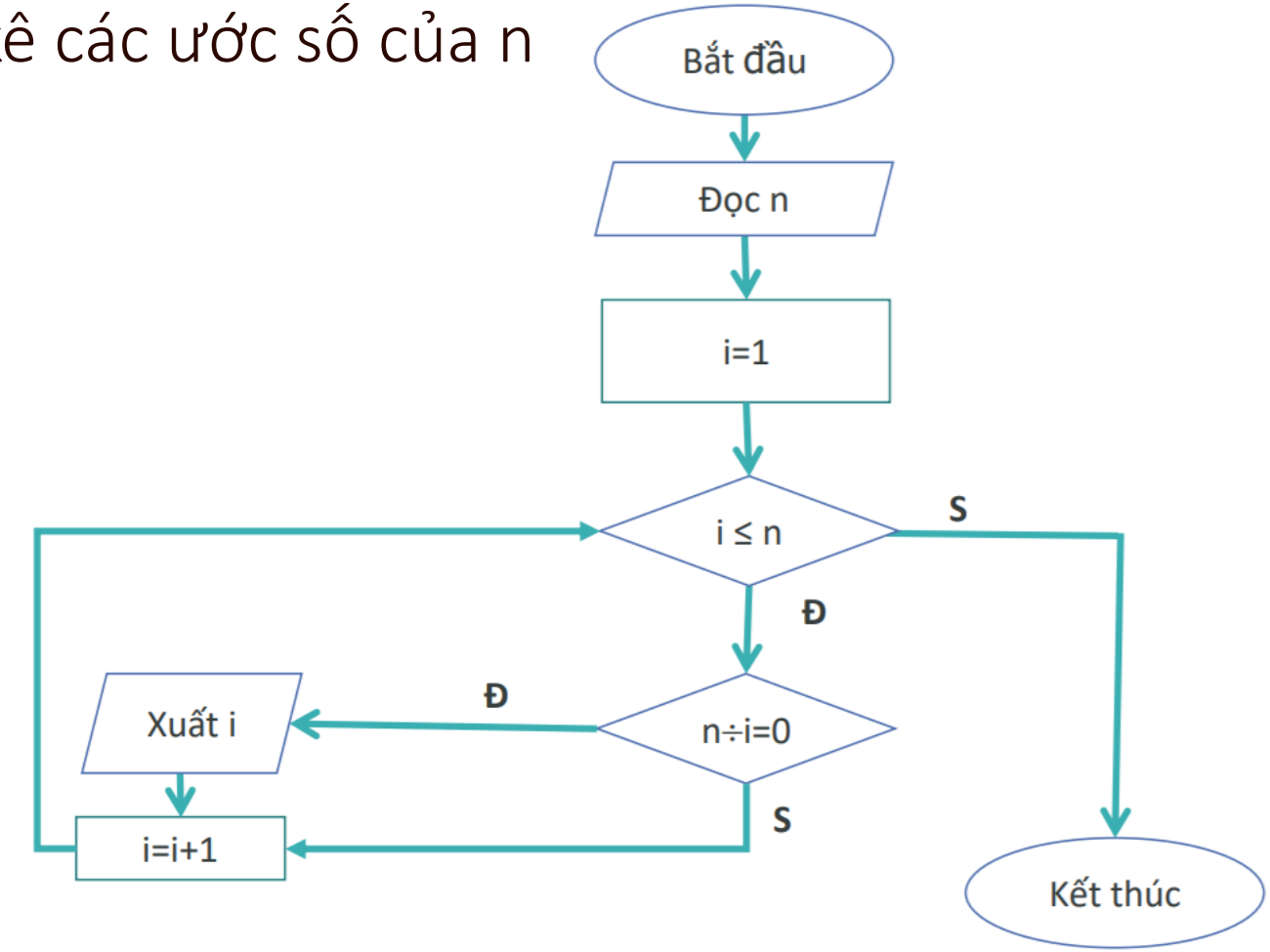
VD1: Tổng các số nguyên dương lẻ từ 1 đến n



BAN HỌC TẬP

Thuật toán

VD2: Liệt kê các ước số của n



BAN HỌC TẬP

Kiểu dữ liệu cơ bản và phép toán

Các kiểu dữ liệu cơ bản

Kiểu số nguyên

Kiểu dữ liệu	Kích thước	Phạm vi
short	2 bytes	[-32.768, 32.767]
unsigned short	2 bytes	[0, 65.535]
int	4 bytes	[-2.147.483.648, 2.147.483.647]
unsigned	4 bytes	[0, 4.294.967.295]
long	4 bytes	[-2.147.483.648, 2.147.483.647]
unsigned long	4 bytes	[0, 4.294.967.295]
long long	8 bytes	[-9.223.372.036.854.775.807, 9.223.372.036.854.775.807]
unsigned long long	8 bytes	[0, 18.446.744.073.709.551.615]

BAN HỌC TẬP

Các kiểu dữ liệu cơ bản

Kiểu số thực

Kiểu dữ liệu	Kích thước	Phạm vi
float	4 bytes	[3.4E-38, 3.4E+38](~7 chữ số)
double	8 bytes	[1.7E-308,1.7E+308](~15 chữ số)
long double	8 bytes	[1.7E-308,1.7E+308](~15 chữ số)

Kiểu kí tự

Kiểu dữ liệu	Kích thước	Giá trị
char	1 byte	[-128, 127] hoặc [0, 255]
unsigned char	1 byte	[0, 255]

Kiểu logic

Kiểu dữ liệu	Kích thước	Giá trị
bool	1 bytes	**false**: giá trị 0 **true**: giá trị khác 0

Các phép toán

- Toán tử gán
- Toán tử toán học
- Toán tử tăng giảm
- Toán tử toán học và gán
- Toán tử quan hệ
- Toán tử luận lý

Các phép toán

- Toán tử gán
  - Dùng để gán giá trị cho 1 biến
  - Ví dụ:

```
int x = 10;
int y = x;
```

→ Gán 10 cho biến x
Gán giá trị của biến x cho biến y →

```
x = 10
y = 10
```

- Toán tử toán học

Toán tử	Miêu tả	Ví dụ
+	Cộng hai toán hạng	10 + 20 kết quả là 30
-	Trừ hai toán hạng	8 - 2 kết quả là 5
*	Nhân hai toán hạng	7 * 5 kết quả là 35
/	Phép chia	15 / 3 kết quả là 5
%	Phép chia lấy số dư	27 % 5 kết quả là 2

Các phép toán

- Toán tử tăng giảm
  - Dùng để tăng ++ hoặc giảm – – 1 đơn vị
  - Ví dụ: để tăng giá trị biến a lên 1 đơn vị ta có thể dùng câu lệnh sau

```
a = a + 1;      a += 1;      a++;
```

  - Sự khác biệt giữa ++x và x++?

```
int x = 5;
int y = ++x;
// x = 5, y = 6
```

1. ++x → x = 6
2. y = x → y = 6

```
int x = 5;
int y = x++;
// x = 6, y = 6
```

1. y = x → x = 5, y = 5
2. x++ → x = 6, y = 5

Các phép toán

- Toán tử quan hệ

Toán tử	Miêu tả	Ví dụ
==	Kiểm tra nếu 2 toán hạng bằng nhau hay không. Nếu bằng thì kết quả là true.	10 == 20 là false
!=	Kiểm tra 2 toán hạng có giá trị khác nhau hay không. Nếu không bằng thì kết quả là true.	10 != 20 là true
>	Kiểm tra toán hạng bên trái có giá trị lớn hơn toán hạng bên phải hay không. Nếu lớn hơn thì kết quả là true.	10 > 20 là false
<	Kiểm tra toán hạng bên trái nhỏ hơn toán hạng bên phải hay không. Nếu nhỏ hơn thì kết quả là true.	10 < 20 là true
>=	Kiểm tra toán hạng bên trái lớn hơn hoặc bằng toán hạng bên phải hay không. Nếu đúng thì kết quả là true.	10 >= 20 là false
<=	Kiểm tra toán hạng bên trái nhỏ hơn hoặc bằng toán hạng bên phải hay không. Nếu đúng thì kết quả là true.	10 <= 20 là true

BAN HỌC TẬP

Các phép toán

- Toán tử luận lý

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

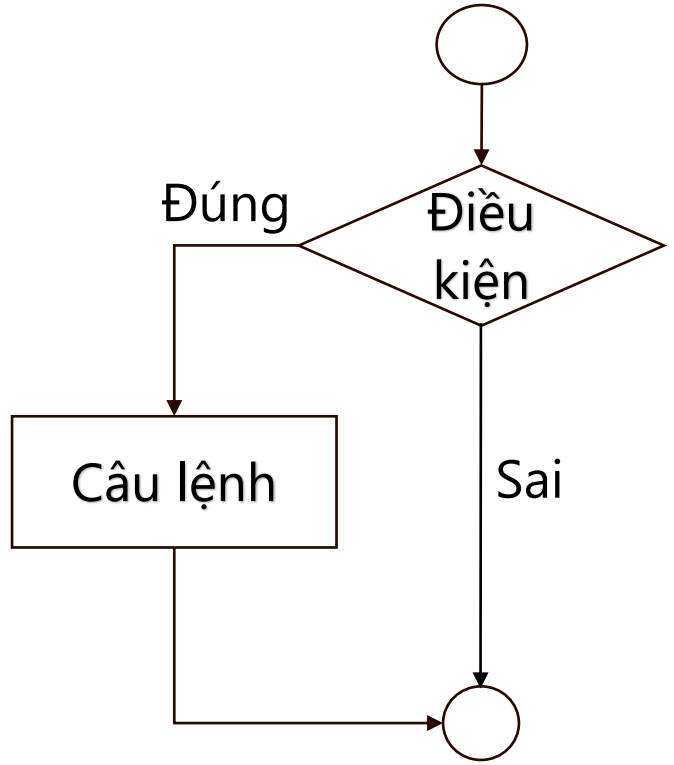
Cấu trúc điều khiển

1. Cấu trúc rẽ nhánh if
2. Cấu trúc rẽ nhánh if-else
3. Cấu trúc rẽ nhánh switch-case

BAN HỌC TẬP

Cấu trúc rẽ nhánh

- Cấu trúc rẽ nhánh if
  - Cú pháp: `if (<điều kiện>)  <câu lệnh>;`
  - Lưu đồ:



BAN HỌC TẬP

Cấu trúc rẽ nhánh

- Cấu trúc rẽ nhánh if
  - VD1: Viết chương trình tìm giá trị lớn nhất trong ba số a,b,c cho trước
    - **Ý tưởng:** gán giá trị a vào m → so sánh m lần lượt với b và c. Nếu m nhỏ hơn thì gán giá trị đó vào m → Xuất m
    - **Code:**

```
#include<iostream>
using namespace std;
int main() {
    int m,a,b,c;
    cout<<"Nhap cac gia tri a, b, c "; cin>>a>>b>>c;
    m=a;
    if (m<b) m=b;
    if (m<c) m=c;
    cout<<"Gia tri lon nhat la: "<<m;
    return 0;
}
```

Cấu trúc rẽ nhánh

- Cấu trúc rẽ nhánh if
  - VD2: Viết chương trình kiểm tra tính chẵn lẻ của một số nguyên n
    - **Ý tưởng:** chia lấy phần dư n cho 2. Nếu bằng 0 là số chẵn, khác 0 là số lẻ
    - **Code:**

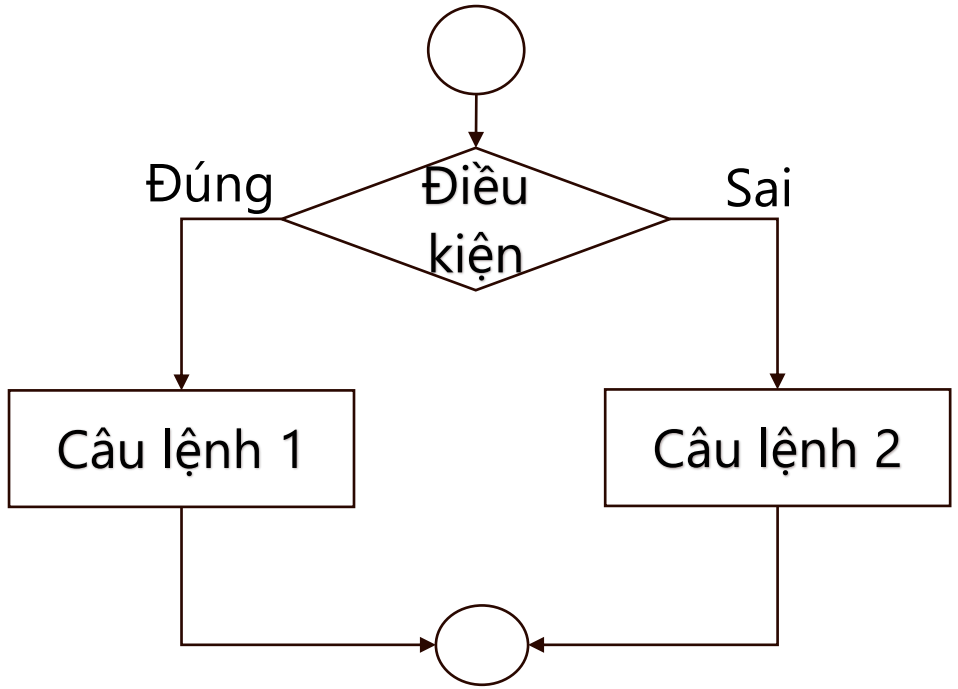
```
#include<iostream>
using namespace std;

int main()
{
    int n;

    cout<<"Nhap so nguyen n: "; cin>>n;
    if (n%2==0) cout<<n<<" la so chan"<<endl;
    if (n%2!=0) cout<<n<<" la so le";

    return 0;
}
```

Cấu trúc rẽ nhánh

- Cấu trúc rẽ nhánh if-else
  - Cú pháp:

```
if (<điều kiện>)   <câu lệnh 1>;
else   <câu lệnh 2>;
```

  - Lưu đồ:



BAN HỌC TẬP

Cấu trúc rẽ nhánh

- Cấu trúc rẽ nhánh if-else
  - VD: Nhập điểm ba môn học. Tính điểm trung bình và kiểm tra đậu hay không đậu.
    - **Ý tưởng:** Tính đtb → Nếu nhỏ hơn 5 → không đâu, ngược lại → đậu
    - **Code:**

```
#include<iostream>
using namespace std;
int main()
{
    int a,b,c,dtb;
    cout<<"Nhap diem 3 mon hoc: ";
    cin>>a>>b>>c;
    dtb=(a+b+c)/3;
    if (dtb<5) cout<<"Bai thi khong dau";
    else cout<<"Bai thi dau";
    return 0;
}
```

Cấu trúc rẽ nhánh

- Các câu lệnh if lồng nhau
  - VD: Viết chương trình nhập vào 3 số nguyên, kiểm tra xem 3 số đó có tạo thành ba cạnh của một tam giác vuông hay không
    - **Ý tưởng:**

      Nhập 3 số nguyên

      Kiểm tra 3 số nguyên có tạo thành một tam giác hay không

      (Tổng 2 cạnh bất kì lớn hơn cạnh còn lại)

      Nếu tạo thành một tam giác thì 3 số nguyên có tạo thành một tam giác vuông không

      (Tổng bình phương 2 cạnh bất kì bằng bình phương cạnh còn lại)

Cấu trúc rẽ nhánh

- Các câu lệnh if lồng nhau
  - VD: Viết chương trình nhập vào 3 số nguyên, kiểm tra xem 3 số đó có tạo thành ba cạnh của một tam giác vuông hay không
    - **Code:**

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

int main()
{
    double a,b,c;
    cout<<"Nhap 3 so: "; cin>>a>>b>>c;
    if ((a+b>c)&&(a+c>b)&&(b+c>a))
        if ((a*a+b*b==c*c)||(a*a+c*c==b*b)||(b*b+c*c==a*a))
            cout<<"3 so tao thanh tam giac vuong";
        else cout<<"3 so khong tao thanh tam giac vuong";
    else cout<<"3 so khong tao thanh tam giac";
    return 0;
}
```

Cấu trúc rẽ nhánh

- Cấu trúc switch case
  - Cú pháp:

```
switch (<biểu thức điều kiện>)
{
    case <giá trị 1>: <câu lệnh 1>; break;
    case <giá trị 2>: <câu lệnh 2>; break;
    ……
    case <giá trị n>: <câu lệnh n>; break;
    default:
        [câu lệnh mặc định];
}
```

Cấu trúc rẽ nhánh

- Cấu trúc switch case
  - VD: Nhập số nguyên. Hiển thị cách đọc số đó có giá trị từ 0 đến 9. Nếu không thì xuất "Không đọc được".
    - **Code:**

```
#include<iostream>
using namespace std;
int main()
{
    int n;
    cout<<"Nhap so nguyen n: ";cin>>n;
    switch (n)
    {
        case 0: cout<<" Khong";break;
        case 1: cout<<" Mot";break;
        case 2: cout<<" Hai";break;
        case 3: cout<<" Ba";break;
        case 4: cout<<" Bon";break;
        case 5: cout<<" Nam";break;
        case 6: cout<<" Sau";break;
        case 7: cout<<" Bay";break;
        case 8: cout<<" Tam";break;
        case 9: cout<<" Chin";break;
        default: cout<<"Khong doc duoc";
    }
    return 0;
}
```

Cấu trúc lặp

Cấu trúc lặp

1. Vòng lặp for

Cú pháp:

```
for ([khởi tạo]; [điều kiện]; [bước nhảy]) <câu lệnh>;
```

Trong đó:

- [khởi tạo] là phần để khởi tạo biến đếm. Biến đếm có thể khai báo ngay trong phần khởi tạo. Phần này chỉ chạy một lần duy nhất khi bắt đầu vòng lặp.
- [điều kiện] là điều kiện của vòng lặp. Khi điều kiện sai thì kết thúc vòng lặp.
- [bước nhảy] phần này được thực thi ở cuối vòng lặp. Thường dùng để tăng hoặc giảm giá trị biến đếm.
- <câu lệnh> được thực hiện khi điều kiện đúng.

Các phần [khởi tạo], [điều kiện], [bước nhảy] có thể có nhiều biến, nhiều biểu thức. Những phần này có thể có hoặc không.

Cấu trúc lặp

2. Vòng lặp while

Cú pháp:

```
while (<điều kiện>)
 <câu lệnh>;
```

Trong đó:

- <điều kiện> thường là biểu thức quan hệ, trả về kết quả **true** hoặc **false**.
- <câu lệnh> là câu lệnh được lặp lại cho đến khi điều kiện **sai (false)**.

Cấu trúc lặp

3. Vòng lặp do while

Cú pháp:

```
do {
    <câu lệnh>;
}
while (<điều kiện>);
```

Trong đó:

- <câu lệnh> là lệnh được thực thi trong vòng lặp.
- <điều kiện> là điều kiện của vòng lặp.

Cấu trúc lặp

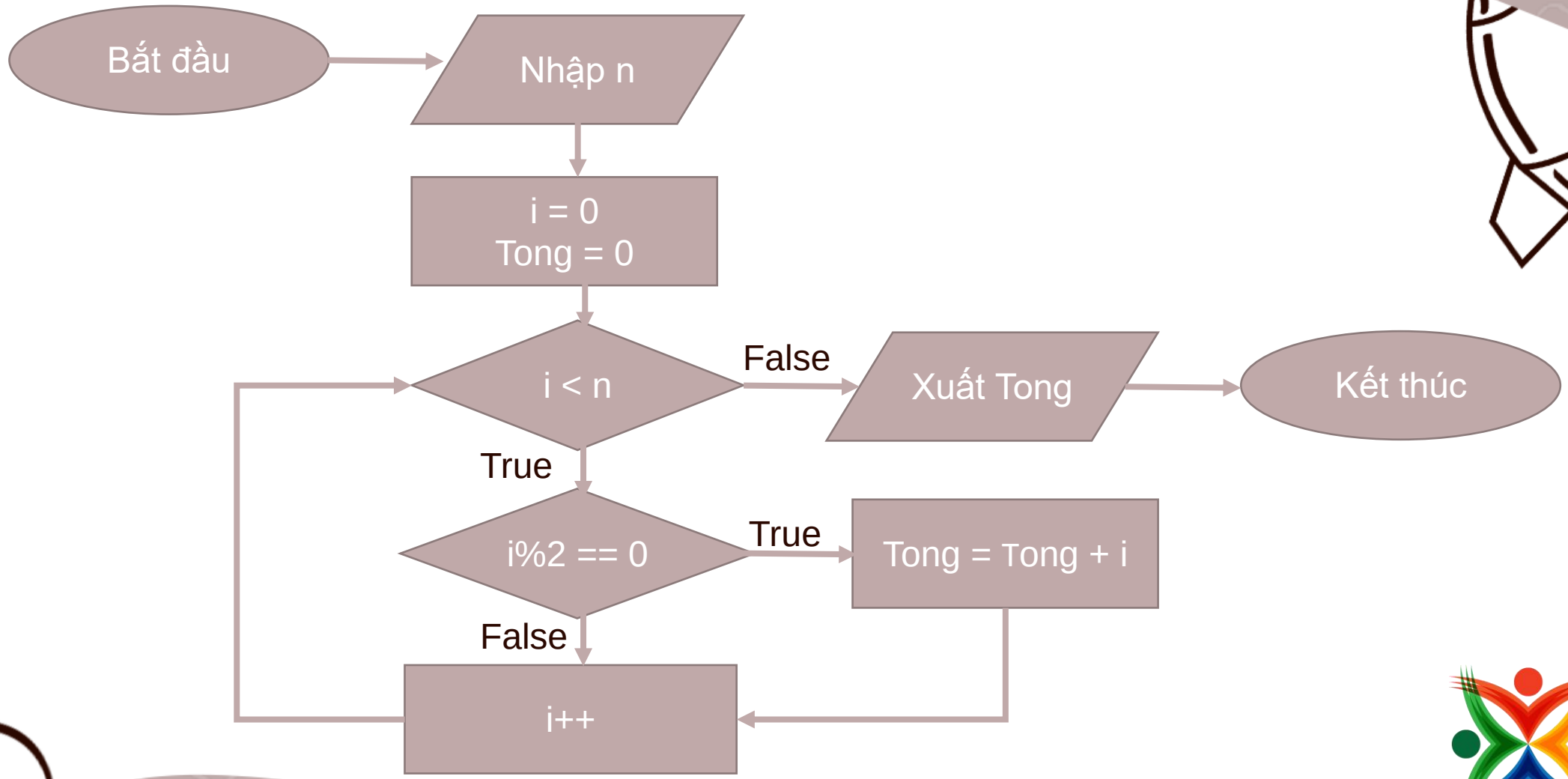
Ví dụ 1: Viết chương trình tính tổng các số nguyên dương chẵn nhỏ hơn N. Với N nhập từ bàn phím.

Ý tưởng:

- Tìm các số dương nhỏ hơn n
- Các số này chia hết cho 2 không?

>> Tính tổng nếu có

Cấu trúc lặp



BAN HỌC TẬP

Cấu trúc lặp

```
#include <iostream>
using namespace std;
int main () {
    int n; cin >> n;
    int sum = 0;
    for (int i = 1; i < n; i++) {
        if (i % 2 == 0)
            sum = sum + i;
    };
    cout << sum;
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main () {
    int N; cin >> N;
    int sum = 0;
    for (int i = 2; i < N; i+=2)
        sum += i;
    cout << sum << endl;
    return 0;
}
```


BAN HỌC TẬP

4. Hàm

Hàm

Không sử dụng hàm

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

do {
    cout << "Nhap mot so nguyen duong";
    cin >> a;
} while (a <= 0);
do {
    cout << "Nhap mot so nguyen duong";
    cin >> b;
} while (b <= 0);
do {
    cout << "Nhap mot so nguyen duong";
    cin >> c;
} while (c <= 0);
do {
    cout << "Nhap mot so nguyen duong";
    cin >> d;
} while (d <= 0);
```

Có sử dụng hàm

```
int nhap_so_duong(){
    int n;
    do {
        cout << "Nhap mot so nguyen duong";
        cin >> n;
    } while (n <= 0);
    return n;
}
```

```
a = nhap_so_duong();
b = nhap_so_duong();
c = nhap_so_duong();
d = nhap_so_duong();
```


BAN HỌC TẬP

Khái niệm hàm

Một đoạn chương trình có tên, đầu vào và đầu ra.

Có chức năng giải quyết một số vấn đề chuyên biệt cho chương trình chính.

Có thể được gọi nhiều lần với các đối số khác nhau.

Được sử dụng khi có nhu cầu:

- Tái sử dụng.
- Sửa lỗi và cải tiến.

Cú pháp của hàm

Cú Pháp: <kiểu_trả_về> <tên_hàm>([danh sách tham số]){

<các câu lệnh>

return <giá_trị_trả_về>; }

Trong đó

- <kiểu trả về> : kiểu dữ liệu bất kỳ của C++ (char, int, long, float,...). Đặc biệt nếu hàm không trả về thì dùng kiểu void.
- <tên hàm>: theo quy tắc đặt tên định danh.
- <danh sách tham số> : tham số hình thức đầu vào giống khai báo biến, cách nhau bằng dấu ",".
- <giá trị> : trả về cho hàm qua lệnh return. Nếu kiểu trả về là void thì không có lệnh này.

Hàm

```
int nhap_so_duong(){
    int n;
    do {
        cout << "Nhap mot so nguyen duong";
        cin >> n;
    } while (n <= 0);
    return n;
}
```

<int> : Kiểu dữ liệu trả về

<nhap_so_duong> : Tên hàm

<return n;> : Giá trị trả về qua biến n;

Tham số hàm

Tham chiếu

- Khai báo hàm sử dụng tham số dưới dạng tham chiếu bằng cách thêm dấu & vào trước tham số đó.
- Khi một biến a được truyền vào lời gọi hàm f(int &x) làm tham số dưới dạng tham chiếu, thì biến x của hàm f(int &x) và biến a thực chất là một. Do đó, nếu hàm thay đổi giá trị của x trong hàm thì đồng nghĩa biến a cũng bị thay đổi theo.

Tham trị

- Khi một biến a được truyền vào lời gọi hàm f(int x) làm tham số dưới dạng tham trị, thì biến x của hàm f(int x) và biến a là hai biến độc lập. Bởi vì khi tham số của hàm là tham trị, hàm này sẽ tạo ra một biến mới và sao chép giá trị của a vào. Do đó, nếu hàm mà thay đổi giá trị của x trong hàm thì không tác động gì tới giá trị của biến a.

BAN HỌC TẬP

Tham số hàm

Tham trị

Tham chiếu

```
bool phep_chia(int x, int y, double& thuong){
    if (y != 0) {
        thuong = double(x)/y;
        return true;
    } else {
        return false;
    }
}
int main(){
    double thuong;
    if (phep_chia(5, 3, thuong)){
        cout << "Thuong so la " << thuong;
    } else {
        cout << "Khong the chia duoc ";
    }
}
```


BAN HỌC TẬP

Bài tập về hàm

- Kiểm tra xem chương trình này có lỗi không ? Hãy sửa lại, nếu có.
- Kết quả sau khi thực thi là gì?

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

void find(int a, int& b, int& c)
{
    int temp;
    c = a + b;
    temp = a;
    a = b;
    b = 2 * temp;
}
int main()
{
    int one, two, three;
    one = 5;
    two = 10;
    three = 15;

    find(three, two, one);
    cout << one << ", " << two << ", " << three << endl;

    return 0;
}
```


BAN HỌC TẬP

Bài tập về hàm

Đoạn chương trình bên có đúng không?
Nếu sai, thì phải sửa lại ở những điểm nào?

```
#include <iostream>
int n;
using namespace std;
void kiem_tra(int n) {
  for (int i = 2; i <= sqrt(n); i++)
        if (n % i = 0) cout << false;
  cout << true;
}
int main() {
  cin >> n;
  cout << kiem_tra(n);
  return 0;
}
```


BAN HỌC TẬP

Bài tập về hàm

```
#include <iostream>
int n;
using namespace std;
void kiem_tra(int n) {
  for (int i = 2; i <= sqrt(n); i++)
        if (n % i = 0) cout << false;
  cout << true;
}
int main() {
  cin >> n;
  cout << kiem_tra(n);
  return 0;
}
```

```
#include <iostream>
#include <cmath>
int n;
using namespace std;
bool kiem_tra(int n) {
  for (int i = 2; i <= sqrt(n); i++)
        if (n % i == 0) return false;
  return true;
}
int main() {
  cin >> n;
  cout << kiem_tra(n);
  return 0;
}
```


BAN HỌC TẬP

Giải đề thi minh họa

Câu 1.

Bạn hãy cho biết giá trị của 「x」 và 「y」 sau khi thực thi đoạn chương trình này.

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

void hamf(int x, int &y)
{
    x = x*y;
    y = x / y;
    x = x / y;
}
int main()
{
    int x = 2, y = 1;

    hamf(x, y);
    cout << "x = " << x << "\t";
    cout << "y = " << y;

    return 0;
}
```


BAN HỌC TẬP

Giải đề thi minh họa

Câu 2.

Bạn hãy cho biết giá trị của 「tong」 sau khi thực thi đoạn chương trình này.

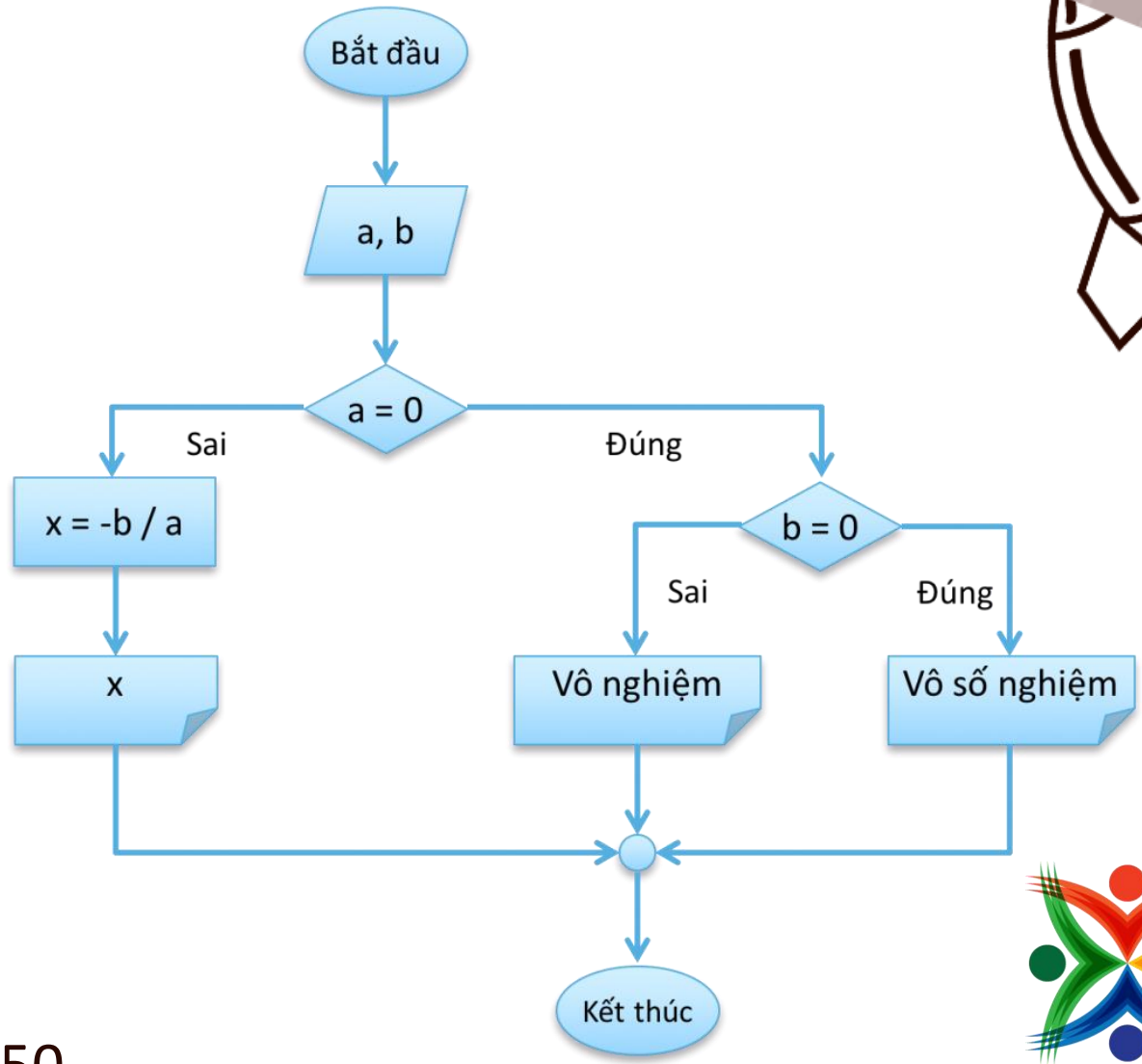
```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
  int n, x; cin >> n >> x;
  double tong = 0;

  for (int i = 0; i <= 2*n; i++)
      if (i % 2 == 0) tong = tong + pow(x, i);

  cout << tong;
  return 0;
}
```


BAN HỌC TẬP

Giải đề thi minh họa

Câu 3.

- Theo bạn, lưu đồ này hoàn chỉnh chưa?
- Với 「a=24, b=16」 bạn hãy cho biết kết quả sau khi thực hiện lưu đồ trên
- Với 「a=0, b=-6 」 bạn hãy cho biết kết quả sau khi thực hiện lưu đồ trên

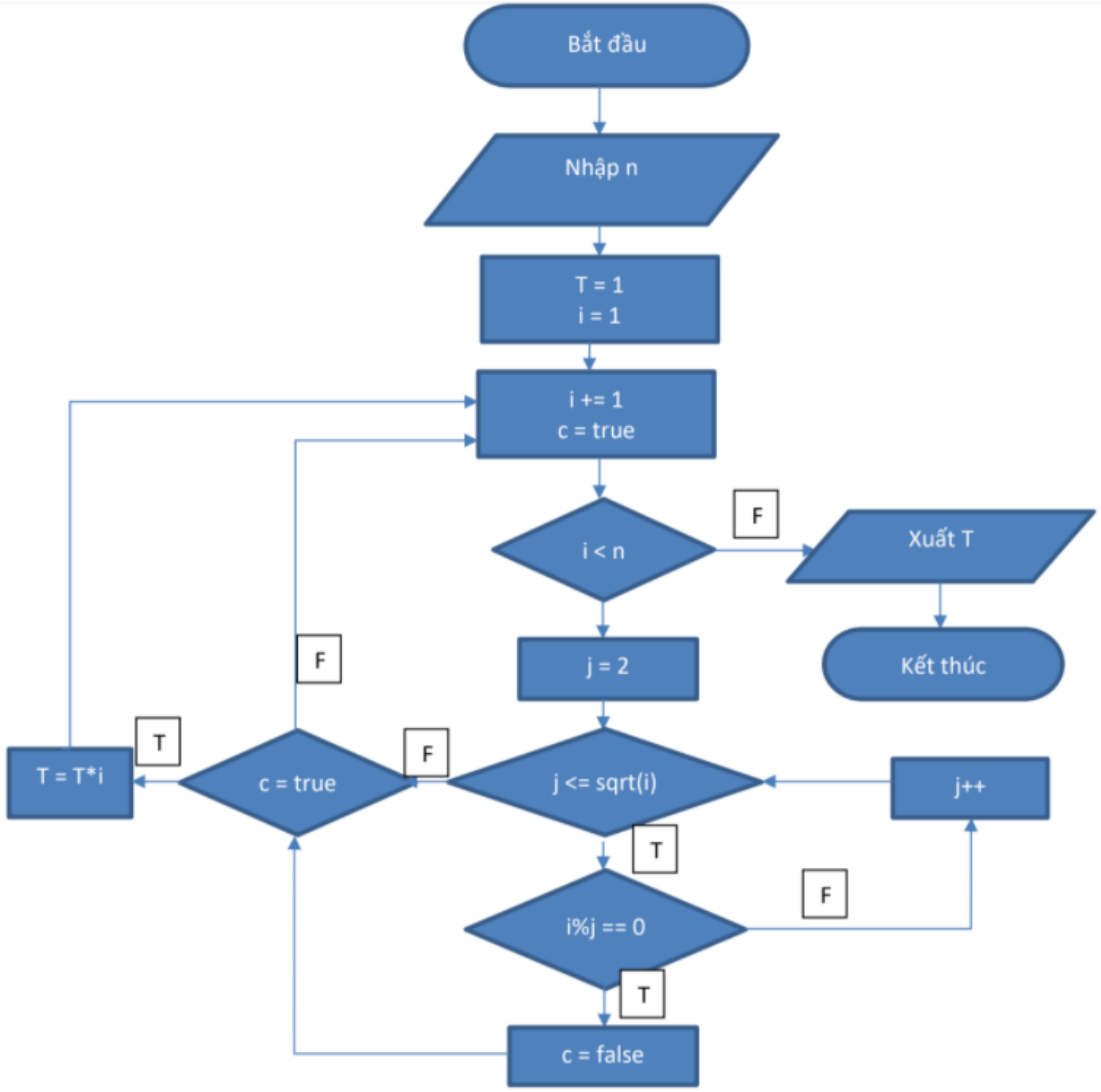


BAN HỌC TẬP

Giải đề thi minh họa

Câu 4.

- Với 「n = 4」, bạn hãy cho biết kết quả sau khi thực hiện lưu đồ trên
- Với 「n = 7」, bạn hãy cho biết kết quả sau khi thực hiện lưu đồ trên



BAN HỌC TẬP

Giải đề thi minh họa

Câu 5. Cho bài toán: Tìm ước số của n nhưng không phải ước số của m.

5a. Vẽ lưu đồ

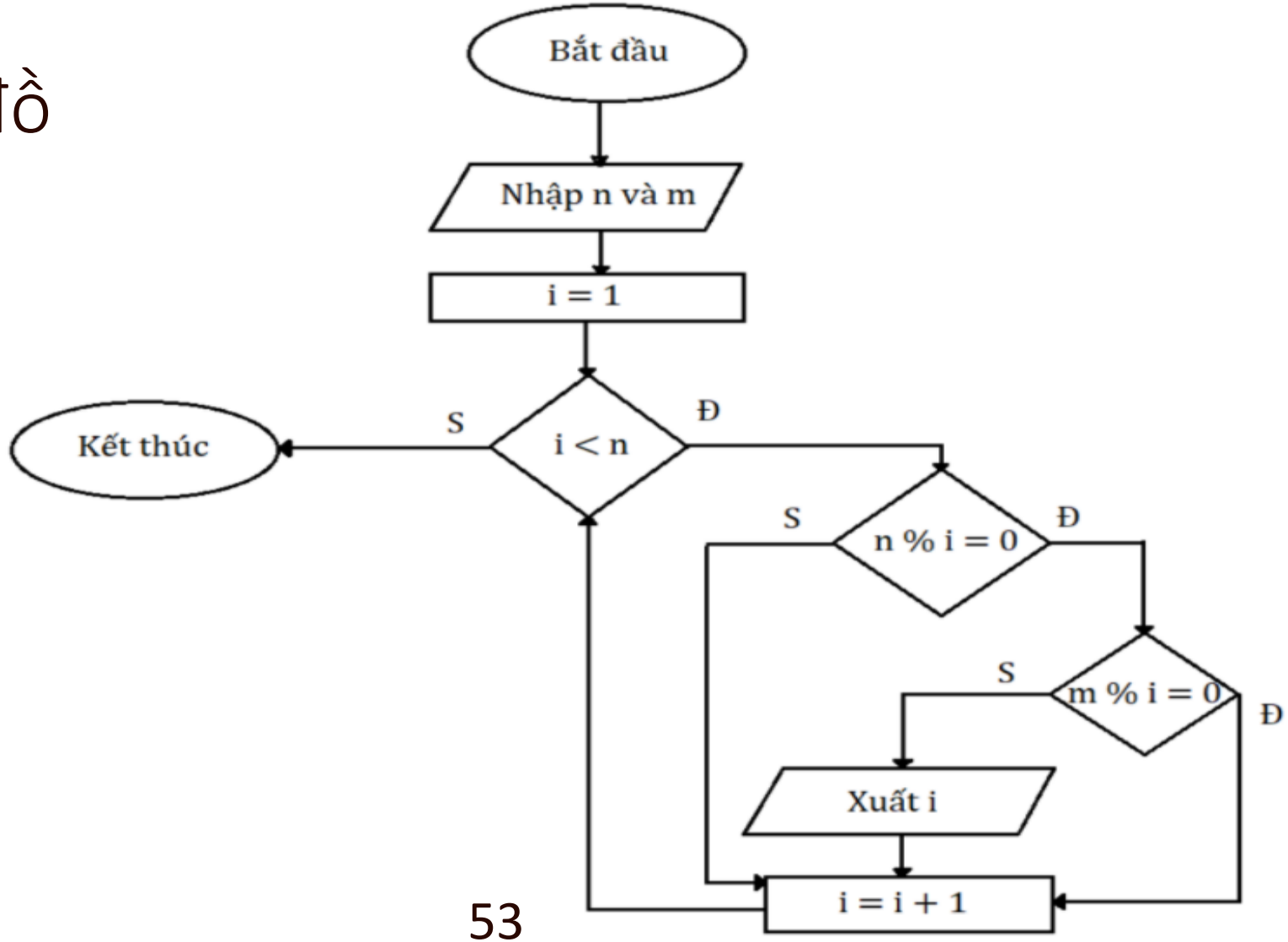
5b. Viết chương trình thực hiện

Input	Output
n = 12, m = 4	3 6 12

BAN HỌC TẬP

Giải đề thi minh hoa

5a. Vẽ lưu đồ



BAN HỌC TẬP

Giải đề thi minh họa

5b. Viết code thực hiện bài toán

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

int main()
{
    int n, m;
    cin >> n >> m;

    for (int i = 2; i <= n; i++) {
        if ((n % i == 0) && (m % i != 0))
            cout << i << "\t";
    };
}
```


BAN HỌC TẬP

6. Các lỗi thường gặp

Lỗi thường gặp khi code

1. Chưa khởi tạo giá trị cho biến

```
1.  int main() {
2.      int count;
3.      while (count < 100) {
4.        cout << count;
5.        count++;
6.      }
7.      return 0;
8.  }
```

2. Sử dụng dấu chấm phẩy

```
1.  int main() {
2.      for (int i = 1; i <= 10; i++);
3.              cout << "Hello\n";
4.  }
```

Các lưu ý khi code giấy

1. Thiếu dấu chấm phẩy 「 ; 」
2. Thiếu dấu ngoặc nhọn 「 { } 」
3. Sử dụng biến sai chữ hoa chữ thường (ví dụ khai báo biến tên Tong nhưng khi sử dụng lại viết là tong)
4. Quên không sử dụng kiểu dữ liệu (ví dụ cần khai báo biến Tong = 0, chính xác phải là int Tong = 0 nhưng lại viết là Tong = 0)
5. Sử dụng tên biến trùng với từ khóa
6. Nhầm lẫn các dấu:
   + Dấu 「=」 và dấu 「==」
   + Dấu 「&」, dấu 「&&」, dấu 「 || 」

BAN HỌC TẬP

Các lưu ý khi code giấy

1. Ép kiểu dữ liệu (ví dụ cần tính chính xác 1/2 thì cần viết (float)(1)/2, nếu chỉ viết 1/2 thì kết quả = 0)
2. Không khai báo hàm ở phần khai báo
3. Vòng lặp vô hạn (vòng lặp while, gọi hàm thành vòng tròn, ...)
4. Không khai báo hàm ở phần khai báo
5. Chú ý về tham trị và tham chiếu

BAN HỌC TẬP

Điểm danh

Link điểm danh:
https://bit.ly/DiemDanh_NMLT

Các bạn vui lòng sử dụng mail trường để điểm danh.
Cảm ơn các bạn rất nhiều!

BAN HỌC TẬP