

NHẬP MÔN LẬP TRÌNH

CHƯƠNG 4.2: CẤU TRÚC LẶP - ITERATION STRUCTURES

ThS. Nguyễn Thị Ngọc Diễm
diemntn@uit.edu.vn



Nội dung

1. Đặt vấn đề
2. Cấu trúc lặp for – the for loop
3. Cấu trúc lặp while - the while loop
4. Cấu trúc lặp (do-while) – the (do-while) loop
5. Vòng lặp cho các khoảng giá trị - Range-based for loop
6. Câu lệnh break, continue
7. Một số ví dụ minh họa



1. Đặt vấn đề

- Viết chương trình xuất các số từ **1** đến **10**
=> Sử dụng **10** câu lệnh cout
- Viết chương trình xuất các số từ **1** đến **1000**
=> Sử dụng **1000** câu lệnh cout !
- Giải pháp:
 - **Sử dụng cấu trúc lặp** nhằm lặp lại một hành động trong khi còn thỏa một điều kiện nào đó.
 - 3 lệnh lặp: **for**, **while**, **do... while**



2. Cấu trúc lặp for

- Cú pháp: **for** (*initialization*; *condition*; *step*)
statements

Bước 1: initialization: Biểu thức khởi đầu được thực thi

Bước 2: condition: Biểu thức điều kiện được kiểm tra. Nếu đúng thì Statement được gọi, ngược lại thì kết thúc for

Bước 3: step: sau khi được thực thi thì quay lại bước 2

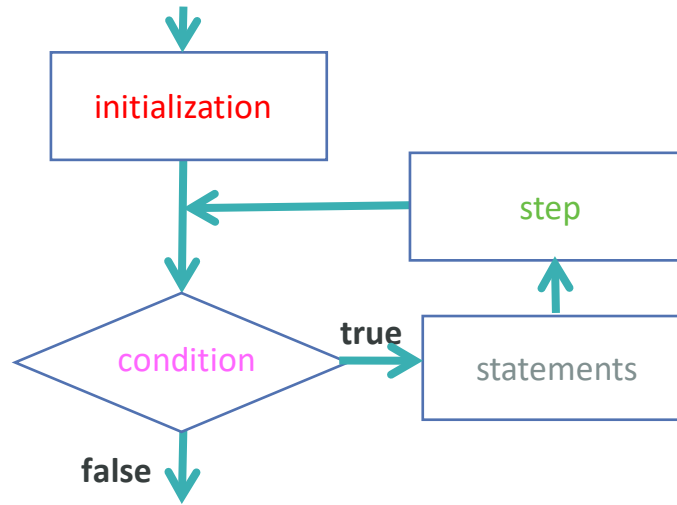
statements: câu lệnh đơn hoặc khối lệnh

- Ví dụ:

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}
```



2. Cấu trúc lặp for





VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
std::cout << "all done" << std::endl;
```

i 0



VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
std::cout << "all done" << std::endl;
```

i 0



VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}
```

```
std::cout << "all done" << std::endl;
```

i 0

```
i = 0
```




VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}
```

```
std::cout << "all done" << std::endl;
```

i 0

```
i = 0
```



VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}
```

```
std::cout << "all done" << std::endl;
```

i 1

```
i = 0
```



VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}
```

```
std::cout << "all done" << std::endl;
```

i 1

```
i = 0
```



VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
std::cout << "all done" << std::endl;
```

i 1

i = 0

i = 1



VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}
```

```
std::cout << "all done" << std::endl;
```

i 1

i = 0

i = 1



VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
std::cout << "all done" << std::endl;
```

i 2

i = 0

i = 1



VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
std::cout << "all done" << std::endl;
```

i 2

i = 0

i = 1



VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
std::cout << "all done" << std::endl;
```

i 2

i = 0

i = 1

i = 2



VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
std::cout << "all done" << std::endl;
```

i 2

i = 0

i = 1

i = 2



VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
std::cout << "all done" << std::endl;
```

i 3

i = 0

i = 1

i = 2



VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
std::cout << "all done" << std::endl;
```

i 3

i = 0

i = 1

i = 2



VD: Chạy từng bước

```
for (int i = 0; i < 3; ++i) {  
    std::cout << "i = " << i << std::endl;  
}  
std::cout << "all done" << std::endl;
```

i 3

i = 0

i = 1

i = 2

all done



Một số lưu ý for

- Câu lệnh **for** là một **câu lệnh đơn** và **có thể lồng nhau**.


```
if (n < 10 && m < 20) {  
    for (int i = 0; i < n; i++) {  
        for (int j = 0; j < m; j++) {  
            cout << i + j;  
            cout << "\n";  
        }  
    }  
}
```



Một số lưu ý for

- Trong câu lệnh for, có thể sẽ không có phần **initialization**.

```
int i;  
for (i = 0; i < 10; i++)  
    cout << i;
```



```
int i = 0;  
for (; i < 10; i++)  
    cout << i;
```

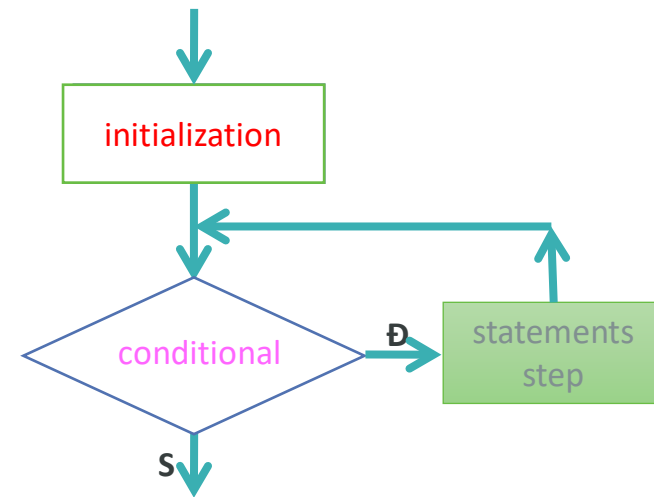


Một số lưu ý for

- Trong câu lệnh for, có thể sẽ không có phần **step**.

```
int i;  
for (i = 0; i < 10; i++)  
    cout << i << endl;
```

```
for (i = 0; i < 10; ) {  
    cout << i << "\n";  
    i++;  
}
```





Một số lưu ý for

- Trong câu lệnh for, có thể sẽ không có phần **condition**.
- Nếu không có **condition** thì câu lệnh for mặc định điều kiện luôn đúng => Cần dùng break, return, continue.

```
int i;  
for (i = 0; i < 10; i++)  
    cout << i << "\n";
```



```
for (i = 0; ; i++) {  
    if (i >= 10)  
        break;  
    cout << i << "\n";  
}
```

```
for (i = 0; ; i++)  
    cout << i << "\n";
```

? Điều kiện dừng



Một số lưu ý for

- Không được thêm **;** ngay sau lệnh `for`.
=> Tương đương câu lệnh rỗng.

```
for (i = 0; i < 10; i++); {  
    cout << i;  
    cout << "\n";  
}
```

```
for (i = 0; i < 10; i++) {  
};  
{  
    cout << "%d", i);  
    cout << "\n");  
}
```



Một số lưu ý for

- Các thành phần **initialization**, **condition**, **step** cách nhau bằng dấu **;**
- Nếu có nhiều thành phần trong mỗi phần thì được cách nhau bằng dấu **,**

```
for (int i = 1, j = 2; i + j < 10; i++, j += 2)  
    cout << i + j <<endl;
```



3. Cấu trúc lặp while

Biểu thức logic xác định khi nào thì Action sẽ được thực thi

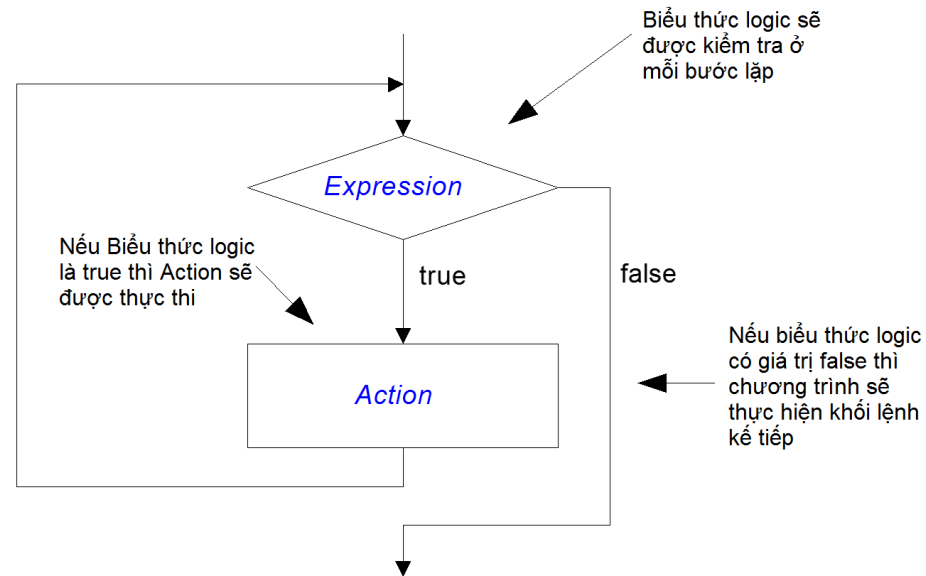
Action sẽ được thực thi cho đến khi biểu thức logic nhận giá trị false

while (*Expression*) *Action*

Action có thể là lệnh đơn hoặc khối lệnh



3. Cấu trúc lặp while





Ví dụ minh họa: Tính trung bình

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số cần nhập vào: 1 5 3 1

n 3



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	0



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	0
sum	0



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	0
sum	0



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	0
sum	0
value	...



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	0
sum	0
value	1



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	0
sum	1
value	1



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	1
sum	1
value	1



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	1
sum	1
value	1



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	1
sum	1
value	...



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	1
sum	1
value	5



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	1
sum	6
value	5



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	2
sum	6
value	5



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	2
sum	6
value	5



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	2
sum	6
value	...



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	2
sum	6
value	3



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	2
sum	9
value	3



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	3
sum	9
value	3



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

Dừng

n	3
count	3
sum	9
value	3



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	3
sum	9
average	3



Chạy từng bước

```
int n = 3;
int count = 0;
double sum = 0;
while (count < n) {
    double value;
    std::cin >> value;
    sum += value;
    count++;
}
double average = sum / count ;
cout << "Average: " << average << std::endl;
```

Các số nhập vào: 1 5 3 1

n	3
count	3
sum	9
average	3



Một số lưu ý

- Câu lệnh **while** có thể không thực hiện lần nào do điều kiện lặp ngay từ lần đầu đã không thỏa.

```
int main() {  
    int n = 1;  
    while (n > 10) {  
        cout << n << endl;  
        n--;  
    }  
    ...  
}
```

```
int main() {  
    int n = 1;  
    do {  
        cout << n << endl;  
        n--;  
    } while (n > 10);  
    ...  
}
```



Một số lưu ý

- Cấu trúc for có thể được viết lại sử dụng cấu trúc while như sau:

```
for (initialization; condition; step)  
    statements
```

```
initialization;  
while (condition) {  
    statements  
    step;  
}
```



Một số lưu ý

- Không được thêm `;` ngay sau lệnh `while`.

```
int n = 0;
while (n < 10);
{
    cout << n << "\n";
    n++;
}
? Kết quả
```

```
int n = 0;
while (n < 10){
};
{
    cout << n << "\n";
    n++;
}
? Kết quả
```



Một số lưu ý

- Câu lệnh **while** có thể bị lặp vô tận (**infinite loop**).

```
int n = 1;
while (n < 10){
    cout << n << endl;
    n--;
}
```



```
int n = 1;
while (n < 10)
    cout << n << endl;
```



4. Cấu trúc lặp do..while

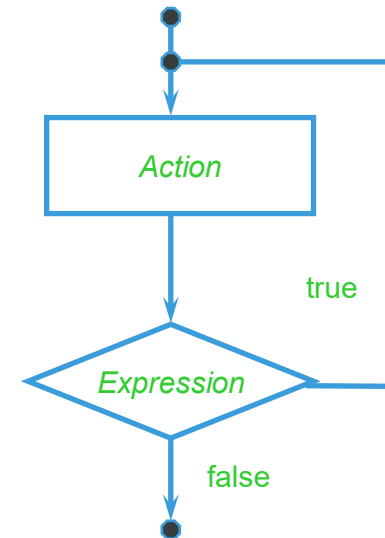
- Cú pháp

do *Action*

while (*Expression*)

- Thực thi

- Thực thi *Action*
- Nếu *Expression* = true thực thi *Action*
- Lặp cho đến khi nào *Expression* = false
- *Action* có thể là lệnh đơn hoặc là khối lệnh





Ví dụ

```
char Reply;  
do {  
    std::cout << "Selection (y, n): ";  
    if (std::cin >> Reply)  
        Reply = tolower(Reply);  
    else  
        Reply = 'n';  
} while ((Reply != 'y') && (Reply != 'n'));
```




Một số lưu ý

- Câu lệnh **do... while** là một **câu lệnh đơn** và **có thể lồng nhau**.

```
int a = 1, b;  
do {  
    b = 1;  
    do {  
        cout << "%d\n", a + b);  
        b = b + 2;  
    } while (b < 20);  
    a++;  
} while (a < 20);
```



Một số lưu ý

- Câu lệnh **do... while** sẽ được thực hiện ít nhất 1 lần do điều kiện lặp được kiểm tra ở cuối.

```
#include<iostream>
int main() {
    int n;
    do {
        std::cout << "Nhap n: ";
        std::cin >> n;
    } while (n < 1 || n > 100);
}
```



Một số lưu ý

- Câu lệnh **do... while** có thể bị lặp vô tận (**infinite loop**)

```
int n = 1;  
do {  
    printf("%d", n);  
    n--;  
} while (n < 10);
```



```
int n = 1;  
do  
    cout << n;  
while (n < 10);
```



Một số lưu ý

- Vòng lặp phải có **điểm dừng**
- Mục đích sử dụng vòng lặp phải rõ ràng
 - Chú thích lại mục đích sử dụng vòng lặp là gì
 - Chú thích cách thực thi vòng lặp để thực hiện được mục đích trên.



6. Câu lệnh break, continue, goto

- Lệnh **break**:
 - Cho phép ra khỏi for, while, do while, switch
 - Dùng để thoát ra khỏi (chấm dứt) các câu lệnh cấu trúc, chương trình sẽ tiếp tục thực hiện các câu lệnh tiếp sau câu lệnh vừa thoát.
- Lệnh **continue** :
 - Lệnh dùng để quay lại đầu vòng lặp mà không chờ thực hiện hết các lệnh trong khối lệnh lặp.
 - Lệnh for: chuyển đến xét step tiếp theo.
 - while/do while:: chuyển tới xác định biểu thức điều kiện và kiểm tra điều kiện kết thúc chu trình.



6. Câu lệnh break, continue, goto

- Ví dụ:

```
for (i = 0; i < 10; i++) {  
    if (i % 2 == 0)  
        break;  
    printf("%d\n", i);  
}
```

```
for (i = 0; i < 10; i++) {  
    if (i % 2 == 0)  
        continue;  
    printf("%d\n", i);  
}
```



6. Câu lệnh break, continue, goto

Lệnh nhảy goto cho phép chương trình chuyển đến thực hiện một đoạn lệnh khác bắt đầu từ một điểm được đánh dấu bởi một nhãn trong chương trình. Nhãn là một tên gọi do NSD tự đặt theo các qui tắc đặt tên gọi. Lệnh goto thường được sử dụng để tạo vòng lặp. Tuy nhiên việc xuất hiện nhiều lệnh goto dẫn đến việc khó theo dõi trình tự thực hiện chương trình, vì vậy lệnh này thường được sử dụng rất hạn chế.

```
#include <iostream>

using namespace std;

int main () {
    int n=10;
    mylabel:
    cout << n << ", ";
    n--;
    if (n>0) goto mylabel;
    cout << "liftoff!\n";
}
```



```
#include <stdio.h>
int main() {
    int control = 5;
    if (control == 1) goto LABEL_1;
    else if (control == 2) goto LABEL_2;
    else if (control == 3) goto LABEL_3;
    else if (control == 4) goto LABEL_4;
    else if (control == 5) goto LABEL_5;
    else goto DEFAULT;
    {
        DEFAULT: printf("Hello world!!!\n");
                printf("Hello world!!!\n");
                printf("Hello world!!!\n");
        LABEL_5: printf("Hello world!!!\n");
        LABEL_4: printf("Hello world!!!\n");
        LABEL_3: printf("Hello world!!!\n");
        LABEL_2: printf("Hello world!!!\n");
        LABEL_1: printf("Hello world!!!\n");
    }
}
```



```
#include <stdio.h>
int main() {
    int control = 5;
    switch (control) {
        default:
            printf("Hello world!!!\n");
            printf("Hello world!!!\n");
            printf("Hello world!!!\n");
        case 5: printf("Hello world!!!\n");
        case 4: printf("Hello world!!!\n");
        case 3: printf("Hello world!!!\n");
        case 2: printf("Hello world!!!\n");
        case 1: printf("Hello world!!!\n");
    }
}
```




7. Một số ví dụ

- Ví dụ 1: Viết chương trình Nhập một số nguyên dương n (có kiểm tra điều kiện nhập) và tính tổng $S=1+2+\dots+n$
- Ví dụ 2: Viết chương trình Liệt kê tất cả các ước số của số nguyên dương n
- Ví dụ 3: Viết chương trình Đếm số lượng chữ số của số nguyên dương n
- Ví dụ 4: Viết chương trình Kiểm tra số nguyên tố (có dùng break)
- Ví dụ 5: Viết chương trình In tất cả các số lẻ nhỏ hơn 50 trừ các số 3,9,31 (có dùng continue)



Bài tập bắt buộc

1. Viết chương trình nhập vào số nguyên dương n . Tính tổng:
2. Viết chương trình nhập vào số nguyên dương n . Tính tổng: $S = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots + \frac{1}{2^n}$.
3. Viết chương trình liệt kê tất cả các số nguyên tố nhỏ hơn giá trị N nhập từ bàn phím ($N < 100$).
4. Viết chương trình tính tổng các chữ số trong 1 số Ví dụ: số 1234 có tổng $S = 1 + 2 + 3 + 4 = 10$.
5. Tìm ước số chung lớn nhất của 2 số nguyên dương a và b .



Chúc các em học tốt!

