I.Câu lệnh if else

1. Câu lệnh if

Câu lệnh if được sử dụng khi bạn cần kiểm tra một điều kiện trước khi thực hiện những câu lệnh khi điều kiện đó có giá trị đúng

Ví dụ : Nếu n là số chẵn sẽ in ra "28tech"

Cú pháp :

```
if(điều_kiện){
    //Các câu lệnh
}
```

Điều kiện ở trong if thường là các phép so sánh, biểu thức trả về giá trị đúng sai. Khi điều kiện này có giá trị đúng thì các câu lệnh bên trong ngoặc nhọn của if sẽ được thực hiện, ngược lại thì các câu lệnh này sẽ bị bỏ qua.

Ví dụ 1 : Kiểm tra n là 28 thì in ra 28tech

```
#include <stdio.h>

int main(){
    int n = 28;
    if(n == 28){
        printf("28tech\n");
    }
    return 0;
}
Output : 28tech
```

Ví dụ 2 : Kiểm tra n là số chẵn thì in ra "CHAN"

Để kiểm tra 1 số là số chẵn bạn tìm số dư của số đó với 2 và so sánh số dư đó với 0

```
#include <stdio.h>

int main(){
```

```
    int n = 28;
    if((n % 2) == 0){
        printf("CHAN\n");
    }
    if((n % 2) != 0){
        printf("LE\n");
    }
    return 0;
}
```

Output : CHAN

Ví dụ 3 : Kiểm tra n chia hết cho m

Để kiểm tra tính chia hết bạn tìm số dư của n với m và so sánh với 0

```
#include <stdio.h>

int main(){
    int n = 28, m = 4;
    if((n % m) == 0){
        printf("Chia het\n");
    }
    if((n % m) != 0){
        printf("Khong chia het\n");
    }
    return 0;
}
```

Output : Chia het

Ví dụ 4 : Kiểm tra n là một trong 4 số 2, 3, 5, 7 thì in ra YES

```
#include <stdio.h>

int main(){
```

```
    int n = 5;
    if((n == 2) || (n == 3) || (n == 5) || (n == 7)){
        printf("YES");
    }
    return 0;
}
```

Output : YES

Chú ý : Trong C các giá trị khác không được coi là đúng nêu bạn có thể sử dụng nó để làm điều kiện cho if. Nếu số mà bạn truyền vào làm điều kiện cho if khác 0 thì code trong if sẽ thực thi và ngược lại.

Ví dụ 5 :

```
#include <stdio.h>

int main(){
    int n = 28, m = 0;
    if(n){
        printf("%d khac 0\n", n);
    }
    if(m){
        printf("%d khac 0\n", m);
    }
    printf("END\n");
    return 0;
}
```

Output :

```
28 khac 0
END
```

2. If Else

Câu lệnh if sẽ giúp thực thi khi điều kiện trong if đúng, còn trong trường hợp sai thì khối lệnh bên trong else sẽ được thực hiện.

If có thể không cần đến else nhưng else thì cần if đi trước.

Cú pháp :

```
if(điều_kiện){
   //Code này sẽ được thực hiện
   //khi điều kiện đúng
}
else{
   //Code này sẽ thực hiện khi điều
   //kiện sai
}
```

Ví dụ 1 : Kiểm tra n là số chẵn nếu đúng in CHAN và 28tech, sai in LE và 28tech.com.vn

```
#include <stdio.h>

int main(){
   int n;
   scanf("%d", &n);
   if(n % 2 == 0){
      printf("CHAN\n");
      printf("28tech\n");
   }
   else{
      printf("LE\n");
      printf("28tech.com.vn\n");
   }
   return 0;
}
```

Ví dụ 2 : Kiểm tra nếu n là năm nhuận in ra YES, ngược lại in ra NO

Năm nhuận là năm chia hết cho 400 hoặc chia hết cho 4 và không chia chết cho 100

```
#include <stdio.h>

int main(){
   int n = 2020;
   if((n % 400 == 0) || (n % 4 == 0 && n % 100 != 0)){
      printf("YES\n");
   }
   else{
      printf("NO\n");
   }
}
```

3. If Else Lồng Nhau

If else có thể lồng vào nhau tức là bên trong khối if else cũng có thể có thêm nhiều câu lệnh if else khác.

Thông thường if else lồng nhau được sử dụng khi điều kiện của bài toán của bạn quá lớn và cần chia nhỏ ra làm nhiều phần để kiểm tra từng bước một.

Ví dụ 1 : Kiểm tra n nằm trong đoạn [20, 50] và chia hết cho ít nhất 1 trong 4 số 2, 3, 5, 7, nếu đúng in YES, ngược lại in NO

Code 1 : Sử dụng if else

```
#include <stdio.h>

int main(){
   int n = 20;
   if((n >= 20 && n <= 50) && (n % 2 == 0 || n % 3 == 0 || n % 5 == 0 || n % 7 == 0)){
      printf("YES\n");
   }
   else{
```

```
        printf("NO\n");
    }
    return 0;
}
```

Code 2 : Sử dụng if else lồng nhau

```
#include <stdio.h>

int main(){
    int n = 20;
    if((n >= 20 && n <= 50)){
        if(n % 2 == 0 || n % 3 == 0 || n % 5 == 0 || n % 7 == 0){
            printf("YES\n");
        }
        else{
            printf("NO\n");
        }
    }
    else{
        printf("NO\n");
    }
    return 0;
}
```

4. If Và Else If

Trong trường hợp bài toán của bạn có nhiều rẽ nhánh khác nhau với các điều kiện tương ứng, nếu chỉ dùng if else thì bạn phải sử dụng if else lồng nhau dẫn tới code trở nên dài dòng và phức tạp.

If và else if giúp bạn có thể kiểm tra nhiều điều kiện và nó chỉ thực hiện duy nhất 1 khối lệnh trong các rẽ nhánh đó.

Cú pháp :

```
if(điều_kiện_1){
   //code 1
}
else if(điều_kiện_2){
   //code 2
}
else if(điều_kiện_3){
   //code 3
}
...
else if(điều_kiện_n){
   //code n
}
else{
   //Code else
}
```

Lưu ý khi sử dụng if và else if :

- Bạn có thể kiểm tra bao nhiêu điều kiện tùy ý
- Khối lệnh else có thể có hoặc không
- Khi rẽ nhánh nào được thực hiện thì các nhánh khác sẽ không được thực hiện, cấu trúc sẽ kết thúc

II. Switch case

1. Switch Case

Switch case sử dụng tương đối giống if và else if, nó cũng giúp bạn có thể kiểm tra nhiều điều kiện để thực hiện các rẽ nhánh khác nhau.

Cú pháp :

```
switch(value){
   case c1:
      //code
      break;
   case c2:
      //code
      break;
   ....
   case cn:
      //code
      break;
   default:
      //code
}
```

Cách hoạt động của switch case đó là sẽ so sánh lần lượt giá trị của value bên trong switch với giá trị của các biến trong các case là c1, c2, ... cn. Nếu giá trị của value bằng giá trị của case nào thì khối lệnh bên trong case đó sẽ được thực hiện.

Nếu giá trị của value không bằng bất cứ giá trị nào trong các case thì khối lệnh trong default sẽ được thực hiện, default trong switch case tương tự như else trong if else bạn đã học ở bài trước.

Ví dụ 1:

```
#include <stdio.h>


int main(){
   int n = 3;
   switch(n){
      case 1:
```

```
            printf("ONE\n");
            break;
        case 2:
            printf("TWO\n");
            break;
        case 3:
            printf("THREE\n");
            break;
        case 4:
            printf("FOUR\n");
            break;
        //Neu n khong phai la 1, 2, 3, 4
        default:
            printf("DEFAULT\n");
    }
}
```

Output :

THREE

Chú ý : Các câu lệnh bên trong case sẽ được kết thúc bởi câu lệnh break. Nếu không có câu lệnh break thì khi code trong nhánh nào được thực hiện, switch case sẽ không kết thúc ngay như else if mà sẽ thực hiện luôn các câu lệnh trong các rẽ nhánh bên dưới.

Ví dụ 2 :

```
#include <stdio.h>

int main(){
    int n = 2;
    switch(n){
        case 1:
            printf("ONE\n");
        case 2:
            printf("TWO\n");
        case 3:
```

```
            printf("THREE\n");
        case 4:
            printf("FOUR\n");
        //Neu n khong phai la 1, 2, 3, 4
        default:
            printf("DEFAULT\n");
    }
}
```

Output :

```
TWO
THREE
FOUR
DEFAULT
```

Ví dụ 3 : Nhập vào toán tử +, -, *, / và in ra kết quả tương ứng với 2 số a, b

```
#include <stdio.h>

int main(){
    int a = 20, b = 5;
    char op;
    scanf("%c", &op);
    switch(op){
        case '+':
            printf("%d\n", a + b);
            break;
        case '-':
            printf("%d\n", a - b);
            break;
        case '*':
            printf("%d\n", a * b);
            break;
        case '/':
            printf("%d\n", a / b);
```

```
            break;
        default:
            printf("INVALID INPUT");
    }
}
```

Ví dụ 4 : Nhập tháng và năm in ra số ngày của tháng, chú ý tháng 2 của năm nhuận có 29 ngày

```
#include <stdio.h>

int main(){
    int m, y;
    printf("Nhap thang, nam : ");
    scanf("%d%d", &m, &y);
    switch(m){
        case 1: case 3: case 5: case 7: case 8: case 10 : case 12:
            printf("31\n");
            break;
        case 4: case 6 : case 9 : case 11:
            printf("30\n");
            break;
        case 2:
            if((y % 400 == 0) && (y % 4 == 0 && y % 100 != 0)){
                printf("29\n");
            }
            else{
                printf("28\n");
            }
            break;
        default:
            printf("Du lieu khong hop le !\n");
    }
}
```

III. For

1. Vòng Lặp For

Vòng lặp giúp bạn thực hiện các câu lệnh nhiều lần tùy theo ý bạn muốn. Ví dụ bạn muốn in ra 1000 dòng chữ "28tech.com.vn" thì thay vì bạn viết 1000 câu lệnh printf bạn có thể viết 1 câu lệnh printf và cho nó lặp 1000 lần.

Cú pháp :

```
for(Câu_lệnh_khởi_tạo ; Điều_kiện_lặp; Câu_lệnh_cập_nhật){
   //code
}
```

Cách vòng lặp For hoạt động :

1. Câu lệnh khởi tạo được thực hiện 1 lần duy nhất khi bắt đầu vào vòng lặp
2. Sau đó điều kiện lặp được kiểm tra, nếu đúng thì sẽ tiến thành thực hiện các câu lệnh trong for, sai sẽ không thực hiện và kết thúc vòng lặp
3. Sau khi các câu lệnh trong vòng lặp được thực hiện xong thì tới phần câu lệnh cập nhật được thực thi
4. Câu lệnh cập nhật được thực hiện xong sẽ tiếp tục kiểm tra điều kiện lặp và nếu đúng tiếp tục thực hiện code trong vòng lặp, sai sẽ kết thúc vòng lặp

Ví dụ 1 : In ra 4 dòng "28tech.com.vn"

```
#include <stdio.h>


int main(){
   int i;
   for(i = 1; i <= 4; i++){
      printf("28tech.com.vn\n");
   }
   return 0;
}
```

Output :

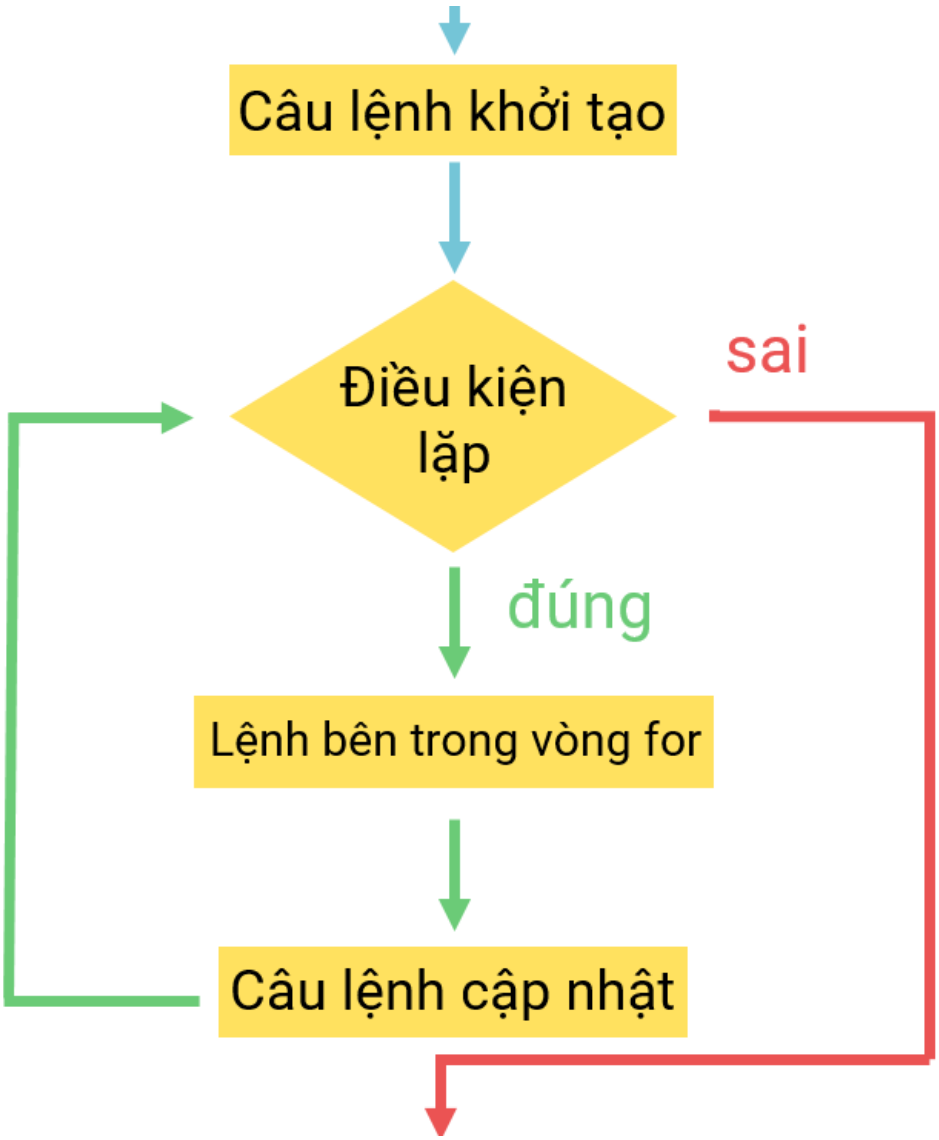
```
28tech.com.vn
28tech.com.vn
28tech.com.vn
28tech.com.vn
```

Giải thích :

1. Biến i được khởi tạo giá trị bằng 1, sau đó kiểm tra điều kiện i <= 4 sẽ có giá trị đúng. Vòng lặp thực hiện lần thứ 1 và in ra 28tech.com.vn
2. Biến i sau câu lệnh cập nhật i++ sẽ có giá trị 2, kiểm tra điều kiện i <= 4 sẽ có giá trị đúng. Vòng lặp thực hiện lần thứ 2 và in ra 28tech.com.vn
3. Biến i sau câu lệnh cập nhật i++ sẽ có giá trị 3, kiểm tra điều kiện i <= 4 sẽ có giá trị đúng. Vòng lặp thực hiện lần thứ 3 và in ra 28tech.com.vn
4. Biến i sau câu lệnh cập nhật i++ sẽ có giá trị 4, kiểm tra điều kiện i <= 4 sẽ có giá trị đúng. Vòng lặp thực hiện lần thứ 4 và in ra 28tech.com.vn
5. Biến i sau câu lệnh cập nhật i++ sẽ có giá trị 5, kiểm tra điều kiện i <= 5 sẽ có giá trị sai. Vòng lặp kết thúc

2. Sơ Đồ Khối Vòng Lặp For

Sơ đồ khối mô tả trực quan cách vòng for hoạt động



3. Ví Dụ

Các ví dụ dưới đây sẽ giúp các bạn làm quen với các vòng lặp thường xuyên được sử dụng, bạn cố gắng viết ra giấy từng bước của vòng lặp như phần giải thích của mình ở mục 1 để hiểu rõ hơn cách vòng for chạy.

Ví dụ 1 : In ra các số từ 1 đến n và từ n về 0

Thông thường nếu bạn muốn thực hiện các khối lệnh n lần thì bạn làm vòng for chạy từ 1 tới n.

```
#include <stdio.h>

int main(){
    int n = 10;
    for(int i = 1; i <= n; i++){
        printf("%d ", i);
    }
    printf("\n");
    for(int i = n; i >= 0; i--){
        printf("%d ", i);
    }
    return 0;
}
```

Output :

```
1 2 3 4 5 6 7 8 9 10
10 9 8 7 6 5 4 3 2 1 0
```

Ví dụ 2 : In ra các ước của N

```
#include <stdio.h>

int main(){
    int n = 100;
    for(int i = 1; i <= n; i++){
        if(n % i == 0){
            printf("%d ", i);
        }
```

```
    }
    return 0;
}
```

Output :

```
1 2 4 5 10 20 25 50 100
```

Ví dụ 3 : Tính tổng các ước của N

```
#include <stdio.h>

int main(){
    int n = 100;
    int tong = 0;
    for(int i = 1; i <= n; i++){
        if(n % i == 0){
            tong += i;
        }
    }
    printf("%d", tong);
    return 0;
}
```

Output :

```
217
```

Ví dụ 4 : Tính N giai thừa (N! = 1.2.3....N)

```
#include <stdio.h>

int main(){
    int n = 5;
    int giaithua = 1;
    for(int i = 1; i <= n; i++){
        giaithua *= i;
```

```
    }
    printf("%d", giaithua);
    return 0;
}
```

Output :

```
120
```

Ví dụ 5 : Tính tổng và in ra bội số của 3 trong đoạn [a, b]

```
#include <stdio.h>

int main(){
    int a = 10, b = 31;
    int tong = 0;
    for(int i = a; i <= b; i++){
        if(i % 3 == 0){
            printf("%d ", i);
            tong += i;
        }
    }
    printf("\nTong boi cua 3 la : %d\n", tong);
    return 0;
}
```

Output :

```
12 15 18 21 24 27 30
Tong boi cua 3 la : 147
```

4. Một Số Vòng Lặp For Đặc Biệt

Vòng lặp for thường có 3 phần và các câu lệnh, tuy nhiên đôi khi bạn có thể để khuyết các phần này.

Ví dụ 1 : Vòng lặp for không có điều kiện lặp sẽ lặp vĩnh viễn

```
#include <stdio.h>

int main(){
   for(int i = 1; ;i++){
      printf("28tech\n");
      printf("lap vinh vien\n");
      printf("lap trinh C\n");
   }
   return 0;
}
```

Ví dụ 2 : Vòng lặp không có điều kiện lặp và câu lệnh cập nhật cũng lặp vĩnh viễn

```
#include <stdio.h>

int main(){
   for(int i = 1; ;){
      printf("28tech\n");
      printf("lap vinh vien\n");
      printf("lap trinh C\n");
   }
   return 0;
}
```

Ví dụ 3 : Vòng lặp for có thể khuyết cả 3 phần, mục đích là tạo vòng lặp vĩnh viễn

Code sau sẽ yêu cầu người dùng nhập số cho tới khi nào nhập số 28 mới dừng vòng lặp.

```
#include <stdio.h>

int main(){
   int n;
```

```
    for(; ;){
        printf("Nhap n : ");
        scanf("%d", &n);
        if(n == 28){
            goto nhan;
        }
    }
    nhan:
    printf("n = %d\n", n);
    printf("Ket thuc !!!\n");
    return 0;
}
```

IV. While

1. Vòng Lặp While

Vòng lặp while là vòng lặp thông dụng thứ 2 sau vòng lặp for, cú pháp của while có phần dễ hiểu hơn so với vòng lặp for.

Cú pháp :

```
while(condition){
   //code
}
```

Cách hoạt động của vòng while :

1. Vòng lặp while vào kiểm tra điều kiện condition bên trong ngoặc tròn
2. Nếu điều kiện có giá trị sai, vòng lặp sẽ không thực hiện code mà kết thúc ngay
3. Nếu điều kiện này có giá trị đúng, vòng lặp sẽ tiến hành thực hiện code bên trong vòng lặp, sau khi thực hiện sau các câu lệnh code vòng lặp while sẽ quay lại kiểm tra điều kiện của condition
4. Bước 3 được lặp đi lặp lại cho tới khi điều kiện trong while bị sai thì vòng lặp kết thúc, nếu điều kiện này không sai vòng while sẽ bị lặp vĩnh viễn

Ví dụ 1 : In ra các số từ 1 đến n sử dụng vòng lặp while

```
#include <stdio.h>


int main(){
   int n = 4;
   int i = 1;
   while(i <= n){
      printf("%d ", i);
      ++i;
   }
   return 0;
}
```

Output :

```
1 2 3 4
```

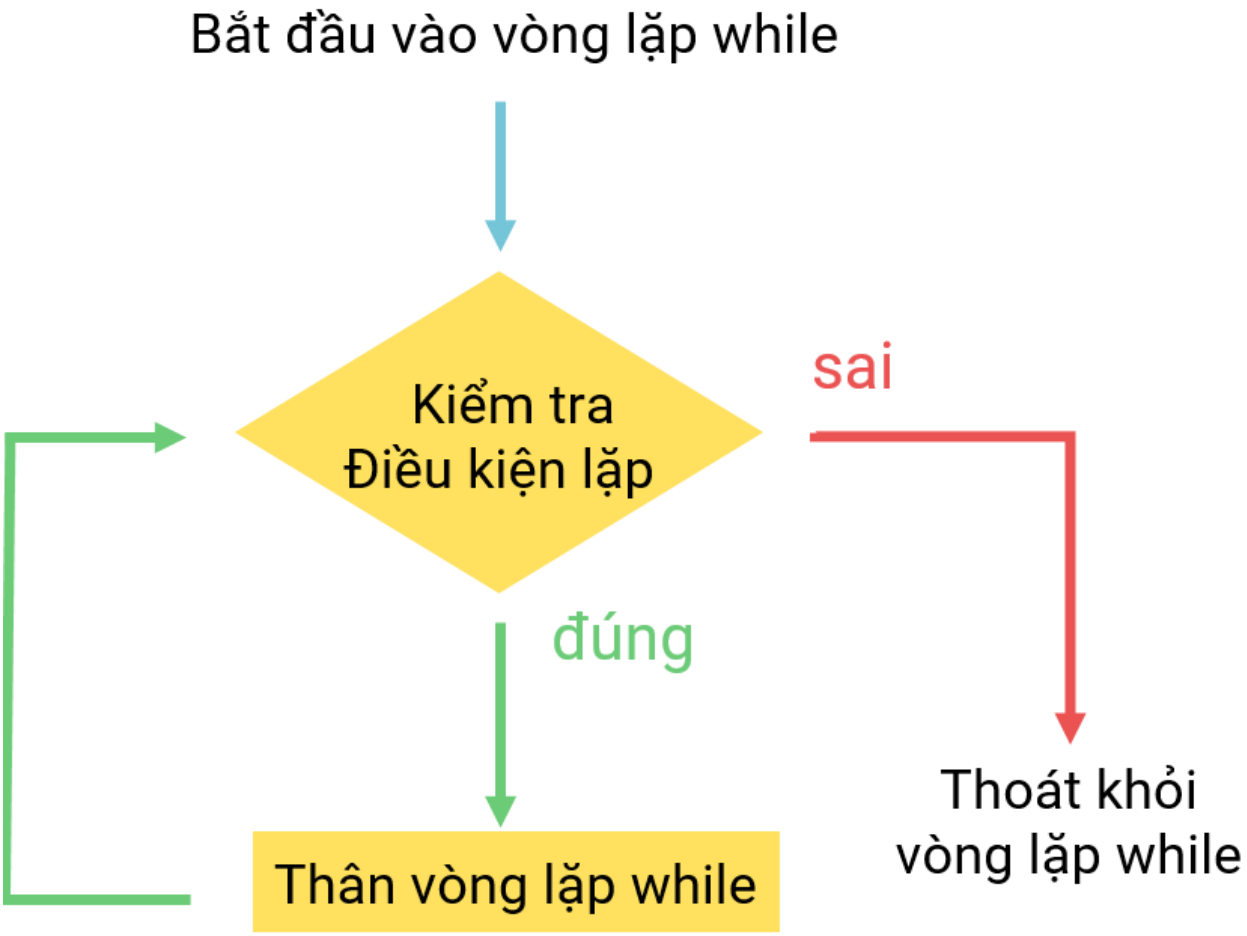
Giải thích :

1. Vòng lặp while kiểm tra i <= n tương đương 1 <= 4 có giá trị đúng, câu lệnh printf in ra 1, sau đó ++i, giá trị của i lên 2

2. Vòng lặp quay lại kiểm tra kiểm tra i <= n tương đương 2 <= 4 có giá trị đúng, câu lệnh printf in ra 2, sau đó ++i, giá trị của i lên 3
3. Vòng lặp quay lại kiểm tra kiểm tra i <= n tương đương 3 <= 4 có giá trị đúng, câu lệnh printf in ra 3, sau đó ++i, giá trị của i lên 4
4. Vòng lặp quay lại kiểm tra kiểm tra i <= n tương đương 4 <= 4 có giá trị đúng, câu lệnh printf in ra 4, sau đó ++i, giá trị của i lên 5
5. Vòng lặp quay lại kiểm tra kiểm tra i <= n tương đương 5 <= 4 có giá trị sai nên vòng lặp kết thúc

2. Sơ Đồ Khối Vòng Lặp While

Các bạn cần lưu ý rằng vòng lặp while có thể không thực hiện bất cứ lần lặp nào nếu điều kiện của nó có giá trị sai



3. Các Bài Toán Với Vòng Lặp While

Khi bạn đã thành thạo cả 2 vòng lặp for và while thì việc sử dụng chúng tương đương nhau, tuy nhiên sẽ có những bài toán sẽ thuận tiện hơn khi sử dụng for hay sử dụng while

Bài toán 1. Đếm chữ số của số tự nhiên N

Ý tưởng : Mỗi lần đếm 1 chữ số hàng đơn vị của n sau đó dùng phép chia nguyên cho 10 để làm mất đi chữ số đó.

```
#include <stdio.h>


int main(){
   int n = 12345;
   int dem = 0;
   while(n != 0){
      ++dem;
      n /= 10;
   }
   printf("So luong chu so cua n : %d\n", dem);
   return 0;
}
```

Output :

```
So luong chu so cua n : 5
```

Giải thích :

1. While kiểm tra điều kiện n != 0 <=> 12345 != 0 là đúng, ++dem thì dem = 1, n /= 10 thì n = 1234
2. While kiểm tra điều kiện n != 0 <=> 1234 != 0 là đúng, ++dem thì dem = 2, n /= 10 thì n = 123
3. While kiểm tra điều kiện n != 0 <=> 123 != 0 là đúng, ++dem thì dem = 3, n /= 10 thì n = 12
4. While kiểm tra điều kiện n != 0 <=> 12 != 0 là đúng, ++dem thì dem = 4, n /= 10 thì n = 1
5. While kiểm tra điều kiện n != 0 <=> 1 != 0 là đúng, ++dem thì dem = 5, n /= 10 thì n = 0
6. While kiểm tra điều kiện n != 0 <=> 0 != 0 là sai, vòng lặp while kết thúc, bạn in ra dem sẽ là số chữ số của n

Bài toán 2 : Tính tổng chữ số của N

Ý tưởng : Mỗi lần tính tổng chữ số hàng đơn vị của n sau đó dùng phép chia nguyên cho 10 để làm mất đi chữ số đó.

```c
#include <stdio.h>


int main(){
   int n = 12345;
   int tong = 0;
   while(n != 0){
      tong += n % 10;
      n /= 10;
   }
   printf("Tong chu so cua n : %d\n", tong);
   return 0;
}
```

Output :

```
Tong chu so cua n : 15
```

Giải thích :

1. While kiểm tra điều kiện n != 0 <=> 12345 != 0 là đúng, tong += n % 10 -> tong = 5, n /= 10 thì n = 1234
2. While kiểm tra điều kiện n != 0 <=> 1234 != 0 là đúng, tong += n % 10 -> tong = 9, n /= 10 thì n = 123
3. While kiểm tra điều kiện n != 0 <=> 123 != 0 là đúng, tong += n % 10 -> tong = 12, n /= 10 thì n = 12
4. While kiểm tra điều kiện n != 0 <=> 12 != 0 là đúng, tong += n % 10 -> tong = 14, n /= 10 thì n = 1
5. While kiểm tra điều kiện n != 0 <=> 1 != 0 là đúng, tong += n % 10 -> tong = 15, n /= 10 thì n = 0
6. While kiểm tra điều kiện n != 0 <=> 0 != 0 là sai, vòng lặp while kết thúc, bạn in ra tong sẽ là tổng chữ số của n

Bài toán 3 : Nhập số từ bàn phím cho tới khi nhập số 28 thì dừng

Ý tưởng : Tạo một vòng lặp while lặp vĩnh viễn, mỗi lần lặp sẽ nhập 1 số và kiểm tra số vừa nhập, nếu n = 28 bạn sẽ cho kết thúc vòng lặp bằng câu lệnh break hoặc goto, nhưng mình chưa giới thiệu câu lệnh break nên mình sẽ dùng câu lệnh goto

```c
#include <stdio.h>


int main(){
```

```
    int n;
    while(1){
        printf("Nhap n : ");
        scanf("%d", &n);
        if(n == 28){
            goto end;
        }
        else{
            printf("Nhap lai\n");
        }
    }
    end:
    return 0;
}
```

Giải thích : Vòng lặp while có điều kiện là 1 sẽ lặp vĩnh viễn, mỗi lần bạn nhập n nếu n là 28 thì câu lệnh goto end được thực thi sẽ thoát vòng lặp while tới câu lệnh bên dưới nhãn end để thực hiện tiếp.

V. Do - While

1. Vòng Lặp Do-While

Nếu bạn đã hiểu rõ cách hoạt động của vòng lặp for và while thì vòng lặp do-while bạn cũng sẽ thấy sự tương đồng của nó với vòng lặp while.

Chỉ có 1 chút khác biệt đó là vòng lặp do-while thực hiện câu lệnh trước rồi mới kiểm tra điều kiện sau

Cú pháp :

```
do{
   //Code
}while(condition);
```

Cách vòng lặp do-while hoạt động :

1. Code bên trong thân vòng lặp được thực hiện lần thứ nhất, sau đó điều kiện (condition) được kiểm tra
2. Nếu điều kiện trong while có giá trị sai vòng lặp do-while sẽ kết thúc
3. Nếu điều kiện trong while đúng, khối lệnh trong do được thực hiện thêm 1 lần nữa
4. Bước 3 được lặp đi lặp lại cho tới khi điều kiện trong while bị sai, nếu điều kiện này luôn đúng vòng lặp sẽ lặp vĩnh viễn

Ví dụ 1: In ra các số từ 1 đến n bằng do-while

```
#include <stdio.h>


int main(){
   int i = 1, n = 4;
   do{
      printf("%d ", i);
      ++i;
   }while(i <= n);
   return 0;
}
```

Output :

```
1 2 3 4
```

Giải thích :

1. Vòng lặp thực hiện in ra i là 1, sau đó ++i thì i lên 2
2. Vòng lặp kiểm tra điều kiện lặp : i <= n tương tương 2 <= 4 có giá trị đúng nên tiếp tục in ra i là 2, ++i thì i tăng lên 3
3. Vòng lặp kiểm tra điều kiện lặp : i <= n tương tương 3 <= 4 có giá trị đúng nên tiếp tục in ra i là 3, ++i thì i tăng lên 4
4. Vòng lặp kiểm tra điều kiện lặp : i <= n tương tương 4 <= 4 có giá trị đúng nên tiếp tục in ra i là 4, ++i thì i tăng lên 5
5. Vòng lặp kiểm tra điều kiện lặp : i <= n tương tương 5 <= 4 có giá trị sai nên vòng lặp kết thúc

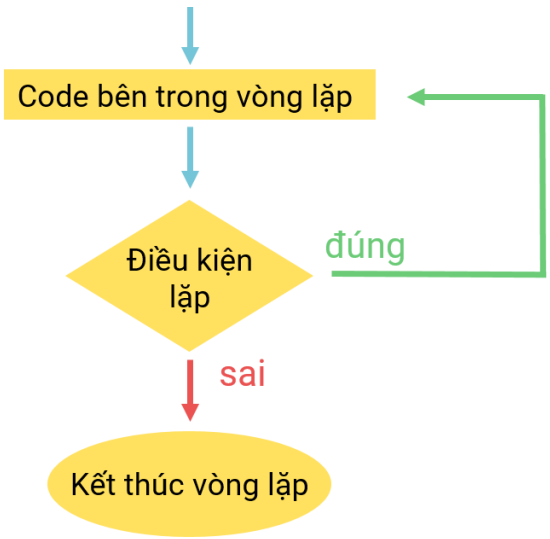
Ví dụ 2 : Nhập 1 số nguyên từ bàn phím nếu nhập số âm thì yêu cầu nhập lại, nhập số không âm thì cho dừng

```c
#include <stdio.h>


int main(){
   int n;
   do{
      printf("Nhap n : ");
      scanf("%d", &n);
   }while(n < 0);
   return 0;
}
```

2. Sơ Đồ Khối

Các bạn lưu ý rằng vòng lặp do-while sẽ luôn thực hiện khối lệnh 1 lần đầu tiên trước khi kiểm tra điều kiện lặp.



VI. Break và Continue

1.Câu Lệnh Break

Break được sử dụng khi bạn muốn dừng vòng lặp (for, while, do-while) ngay lập tức, thông thường câu lệnh break thường đi kèm với 1 điều kiện kích hoạt, ví dụ như sau khi tìm được đáp án trong vòng lặp.

Khi câu lệnh break được thực thi thì các câu lệnh bên dưới break trong vòng lặp sẽ không được thực hiện nốt, vòng lặp sẽ kết thúc ngay tại ví trí của câu lệnh break

Với vòng lặp lồng nhau thì câu lệnh break có ý nghĩa với vòng lặp gần nhất chứa nó.

Xem xét ví dụ sau :

```
#include <stdio.h>


int main(){
   int n = 5;
   for(int i = 1; i <= 5; i++){
      printf("%d 28tech\n", i);
      if(i == 3){
         break;
      }
      printf("C programming !\n");
   }
   return 0;
}
```

Output :

```
1 28tech
C programming !
2 28tech
C programming !
3 28tech
```

Giải thích :

1. Vòng lặp 1 : i = 1, in ra 1 28tech, break chưa thực hiện nên tiếp tục in ra C programming !
2. Vòng lặp 2 : i = 2, in ra 2 28tech, break chưa thực hiện nên tiếp tục in ra C programming !

3. Vòng lặp 3 : i = 3, in ra 3 28tech, do if đúng nên câu lệnh break được thực thi, vòng lặp kết thúc ngay mà không thực hiện câu lệnh in ra C programming !

Ví dụ 1 : Tìm Ước Chung Lớn Nhất của 2 số a và b

Ý tưởng : Duyệt từ số nhỏ hơn trong 2 số a và b về 1, nếu gặp số đầu tiên cả a và b chia hết thì in ra và dừng vòng lặp

```
#include <stdio.h>

int main(){
   int a = 18, b = 12;
   int min = a < b ? a : b;
   for(int i = min; i >= 1; i--){
      if((a % i == 0) && (b % i == 0)){
         printf("%d\n", i);
         break;
      }
   }
   return 0;
}
```

Output :

```
6
```

Ví dụ 2 : In ra chữ số chẵn đầu tiên tính từ bên phải của số tự nhiên n

Ý tưởng : Tách từng chữ số của n bằng vòng lặp while, khi gặp chữ số chẵn thì dừng vòng lặp

```
#include <stdio.h>

int main(){
   int n = 12689791;
   while(n != 0){
      if(n % 2 == 0){ // n % 10 % 2 cũng được
         printf("%d\n", n % 10);
```

```
            break;
        }
        n /= 10;
    }
    return 0;
}
```

Output :

```
8
```

Ví dụ 3 : Nhập 1 số nguyên từ bàn phím nếu nhập số âm thì yêu cầu nhập lại, nhập số không âm thì cho dừng

Ý tưởng : Dùng một vòng lặp lặp vĩnh viễn và kích hoạt câu lệnh break khi người dùng nhập số không âm

```
#include <stdio.h>

int main(){
    int n;
    while(1){
        printf("Nhap n : ");
        scanf("%d", &n);
        if(n >= 0){
            break;
        }
        else{
            printf("Nhap lai\n");
        }
    }
    return 0;
}
```

Ví dụ 4 : Câu lệnh break với vòng for lồng nhau

```
#include <stdio.h>
```

```
int main(){
   for(int i = 1; i <= 2; i++){
      for(int j = 1; j <= 5; j++){
         printf("28tech\n");
         if(j == 2){
            break; // có tác dụng dừng vòng for j
         }
         printf("C programming !\n");
      }
   }
   return 0;
}
```

Output :

```
28tech
C programming !
28tech
28tech
C programming !
28tech
```

2.Câu Lệnh Continue

Câu lệnh continue khi được thực hiện nó sẽ bỏ qua các câu lệnh bên dưới nó trong vòng lặp và quay trở lại vòng lặp mới ngay.

Với vòng lặp lồng nhau thì câu lệnh continue có ý nghĩa với vòng lặp gần nhất chứa nó.

Xem xét ví dụ sau :

```
#include <stdio.h>

int main(){
   for(int i = 1; i <= 4; i++){
      printf("28tech\n");
      if(i % 2 == 0){
         continue;
```

```
        }
        printf("C\n");
    }
    return 0;
}
```

Output :

```
28tech
C
28tech
28tech
C
28tech
```

Giải thích :

1. Vòng lặp i = 1, in ra 28tech, kiểm tra if không đúng nên tiếp tục thực hiện câu lệnh in ra C
2. Vòng lặp i = 2, in ra 28tech, kiểm tra if đúng nên bỏ qua câu lệnh in ra C bên dưới mà quay trở lại luôn cập nhật i
3. Vòng lặp i = 3, in ra 28tech, kiểm tra if không đúng nên tiếp tục thực hiện câu lệnh in ra C
4. Vòng lặp i = 4, in ra 28tech, kiểm tra if đúng nên bỏ qua câu lệnh in ra C bên dưới mà quay trở lại luôn cập nhật i

Ví dụ 1 : Nhập vào các số tự nguyên, kết thúc nhập bằng cách nhập số 0 và chỉ tính tổng các số nguyên dương được nhập

```
#include <stdio.h>


int main(){
    int tong = 0, n;
    while(1){
        printf("Nhap n : ");
        scanf("%d", &n);
        if(n < 0){
            continue; // bỏ qua các lệnh bên dưới
        }
        else if(n == 0){
            break;
        }
```

```
        tong += n;
    }
    printf("%d\n", tong);
    return 0;
}
```