THUẬT TOÁN SINH

Nội dung:

1. Giới thiệu:

Thuật toán sinh là một phương pháp vét cạn được dùng với các bài toán liệt kê hoặc đếm cấu hình, thỏa các yêu cầu sau:

- Có thể xác định được cấu hình đầu tiên và cấu hình cuối cùng.
- Tìm được thuật toán để từ cấu hình hiện tại sinh ra cấu hình kế tiếp.

Các bài toán phổ biến sử dụng phương pháp sinh:

- Liệt kê xâu nhị phân
- Liệt kê hoán vị
- Liệt kê tập con
- Sinh phân hoạch

2. Mã giả:

Mã giả:

```
<Bước 1> : Khởi tạo
    <Khởi tạo cấu hình đầu tiên>
<Bước 2> : Lặp
while(<Chưa gặp cấu hình cuối cùng>){
   <Đưa ra cấu hình hiện tại>
   <Sinh ra cấu hình kế tiếp>
}
<Đưa ra cấu hình cuối cùng>
```

3. Sinh xâu nhị phân:

Phân tích bài toán

Đề bài: Liệt kê xâu nhị phân có độ dài N

OUTPUT (N=3)
000
001
010
011
100
101
110
111

Cấu hình đầu tiên: N bit 0

Cấu hình cuối cùng: N bit 1

3. Sinh xâu nhị phân:

Các biến toàn cục và hàm khởi tạo cấu hình đầu tiên

```
int n, X[100]; // lưu cấu hình
bool final = false; // check cấu hình cuối
void init(){
    for(int i = 1; i <= n; i++){
        X[i] = 0;
    }
}
```

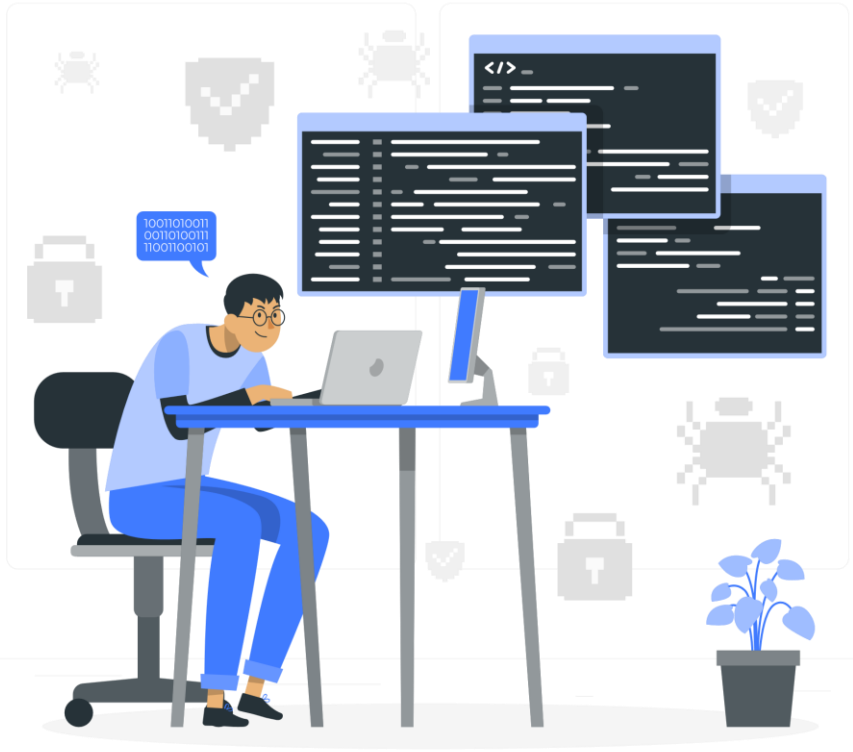
Hàm sinh cấu hình kế tiếp:

```
void sinh(){
    int i = n;
    while(i >= 1 && X[i] == 1){
        X[i] = 0;
        --i;
    }
    if(i == 0){
        final = true;
    }
    else{
        X[i] = 1;
    }
}
```

3. Sinh xâu nhị phân:

Hàm main:

```
int main(){
    n = 3;
    init();
    while(!final){
        for(int i = 1; i <= n; i++){
            cout << X[i];
        }
        cout << endl;
        sinh();
    }
}
```



4. Sinh tổ hợp chập K của N phần tử:

Phân tích bài toán

Đề bài: Cho các số tự nhiên từ 1 đến N, nhiệm vụ của bạn là liệt kê các tập con có K phần tử của tập N phần tử này theo thứ tự từ điển tăng dần.

Cấu hình đầu tiên: 123...K

OUTPUT (N=5, K=3)

123	145
124	234
125	235
134	245
135	345

Cấu hình cuối cùng: N-K+1, N-K+2,...N

4. Sinh tổ hợp chập K của N phần tử:

Các biến toàn cục và hàm khởi tạo cấu hình đầu tiên

```
int n, X[100]; // lưu cấu hình
bool final = false; // check cấu hình cuối
void init(){
    for(int i = 1; i <= k; i++){
        X[i] = i;
    }
}
```

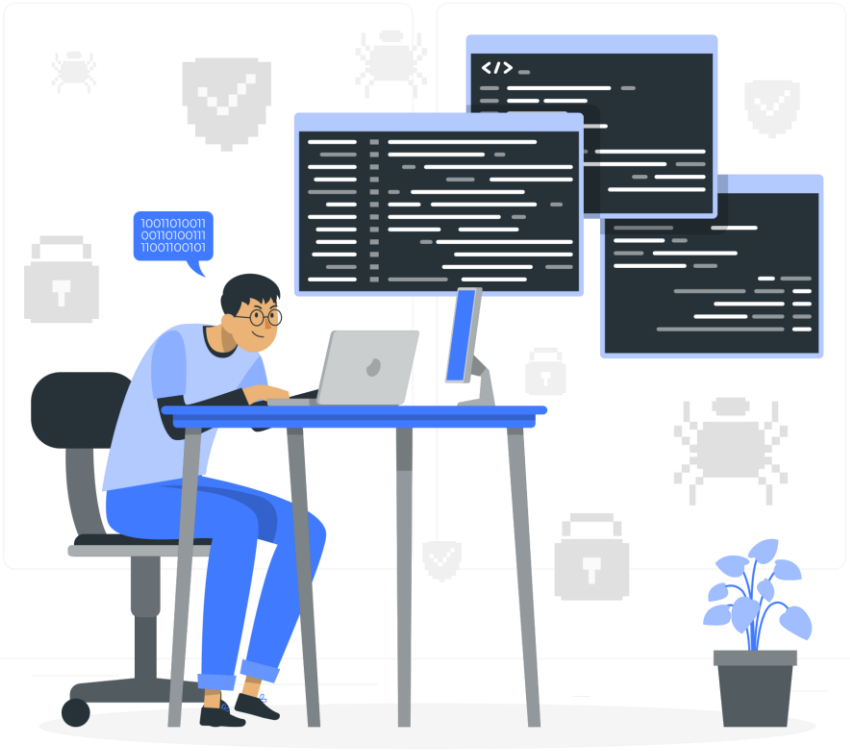
Hàm sinh cấu hình kế tiếp:

```
void sinh(){
    int i = k;
    while(i >= 1 && X[i] == n - k + i){
        --i;
    }
    if(i == 0){
        final = true;
    }
    else{
        X[i]++;
        for(int j = i + 1; j <= k; j++){
            X[j] = X[j - 1] + 1;
        }
    }
}
```

4. Sinh tổ hợp chập K của N phần tử:

Hàm main:

```
int main(){
    n = 5, k = 3;
    init();
    while(!final){
        for(int i = 1; i <= k; i++){
            cout << X[i];
        }
        cout << endl;
        sinh();
    }
}
```



5. Sinh hoán vị:

Phân tích bài toán

Đề bài: Sinh ra các hoán vị của các số tự nhiên từ 1 đến N

OUTPUT (N=3)
123
132
213
231
312
321

Cấu hình đầu tiên: 123...N

Cấu hình cuối cùng: N,N-1,N-2...1

5. Sinh hoán vị:

Các biến toàn cục và hàm khởi tạo cấu hình đầu tiên

```
int n, X[100]; // lưu cấu hình
bool final = false; // check cấu hình cuối
void init(){
    for(int i = 1; i <= n; i++){
        X[i] = i;
    }
}
```

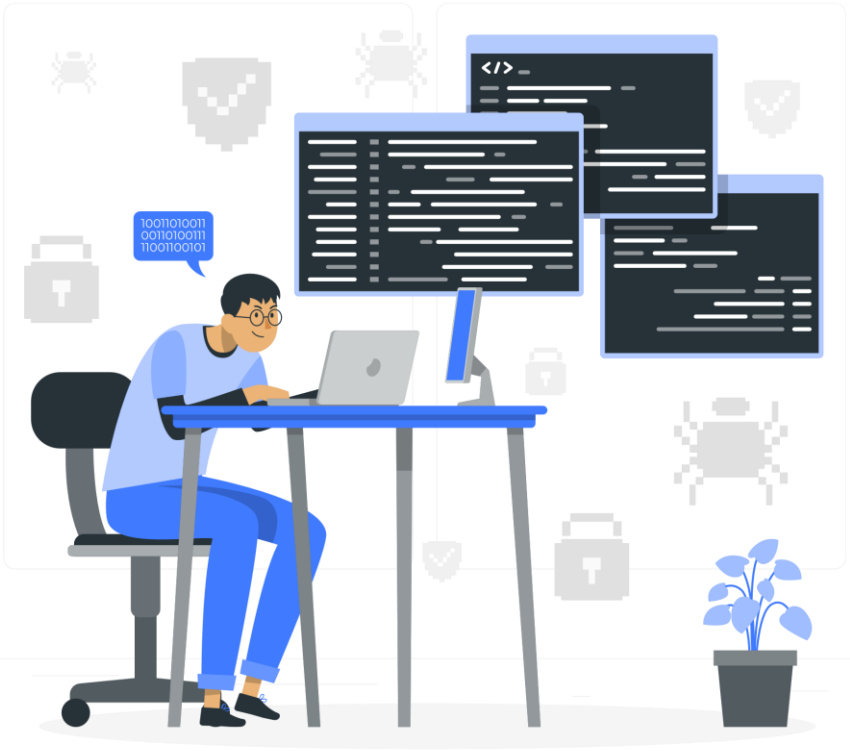
Hàm sinh cấu hình kế tiếp:

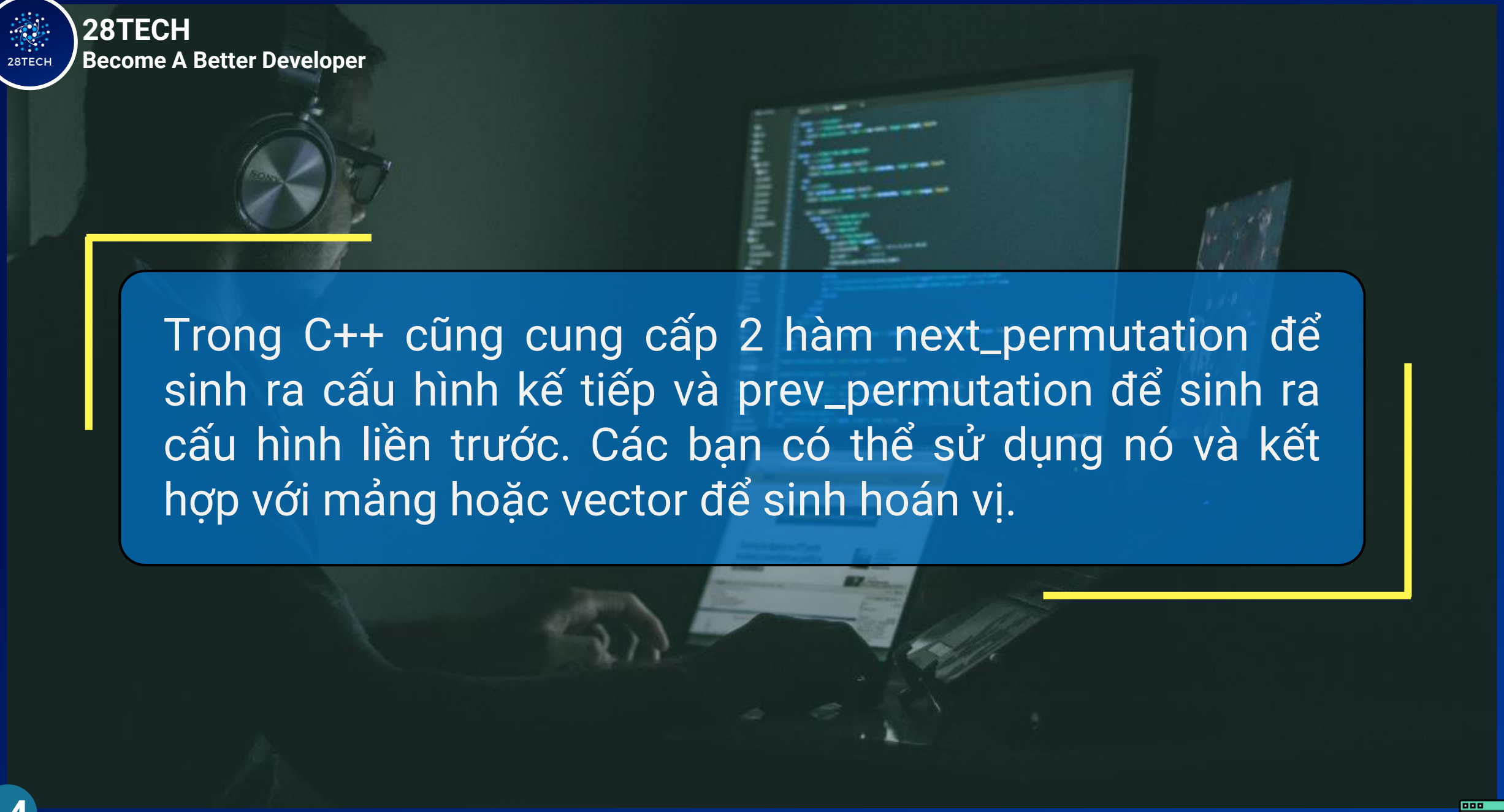
```
void sinh(){
    int i = n;
    while(i >= 1 && X[i] > X[i + 1]){
        --i;
    }
    if(i == 0){
        final = true;
    }
    else{
        int j = n;
        while(X[i] > X[j]) --j;
        swap(X[i], X[j]);
        reverse(X + i + 1, X + n + 1);
    }
}
```

5. Sinh hoán vị:

Hàm main:

```
int main(){
    n = 4;
    init();
    while(!final){
        for(int i = 1; i <= n; i++){
            cout << X[i];
        }
        cout << endl;
        sinh();
    }
}
```





Trong C++ cũng cung cấp 2 hàm next_permutation để sinh ra cấu hình kế tiếp và prev_permutation để sinh ra cấu hình liền trước. Các bạn có thể sử dụng nó và kết hợp với mảng hoặc vector để sinh hoán vị.

5. Sinh hoán vị:

Hàm next_permutation áp dụng với mảng, đối với vector và string các bạn thay bằng iterator begin và end.

Ví dụ:

```
int main(){
    int a[] = {1, 2, 3, 4};
    do{
        for(int i = 0; i < 4; i++){
            cout << a[i];
        }
        cout << endl;
    }while(next_permutation(a, a + 4));
}
```

5. Sinh hoán vị:

Hàm prev_permutation để sinh hoán vị theo thứ tự ngược

Ví dụ:

```
int main(){
    int a[] = {4, 3, 2, 1};
    do{
        for(int i = 0; i < 4; i++){
            cout << a[i];
        }
        cout << endl;
    }while(prev_permutation(a, a + 4));
}
```

6. Sinh phân hoạch:

Phân tích bài toán

Đề bài: In ra các cách phân tích N dưới dạng tổng các số tự nhiên nhỏ hơn hoặc bằng N không xét đến thứ tự.

Cấu hình đầu tiên: N

Cấu hình cuối cùng: N số 1

OUTPUT (N=5)
5
4 + 1
3 + 2
2 + 2 + 1
2 + 1 + 1 + 1
1 + 1 + 1 + 1 + 1

6. Sinh phân hoạch:

Các biến toàn cục và hàm khởi tạo cấu hình đầu tiên

```
int n, X[100]; // lưu cấu hình
int cnt; //Lưu sl số hạng trong phân tích
bool final = false; // check cấu hình cuối
void init(){
    cnt = 1;
    X[1] = n;
}
```

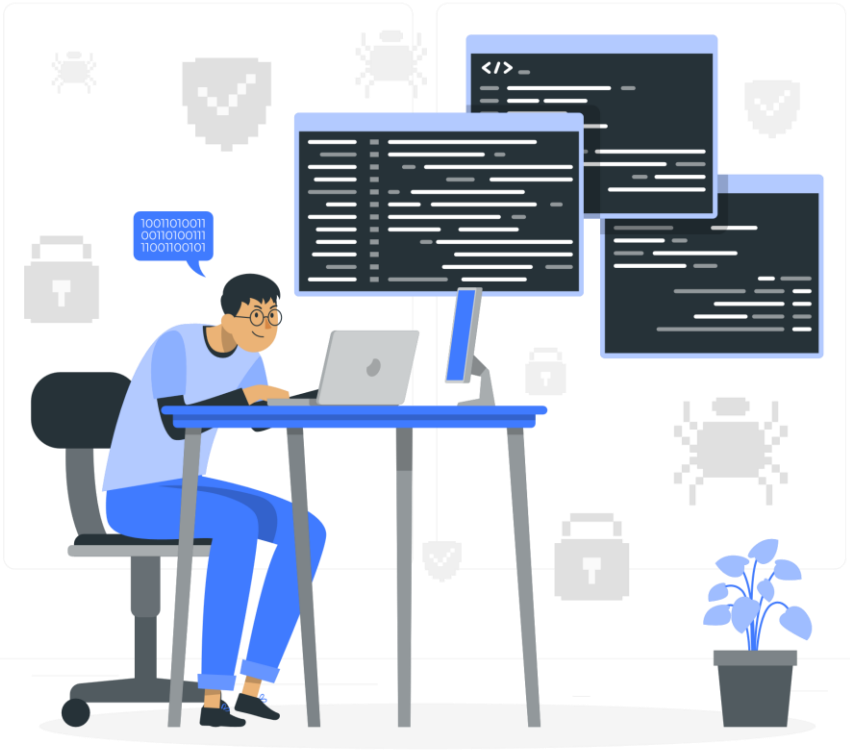
Hàm sinh cấu hình kế tiếp:

```
void sinh(){
    int i = cnt;
    while(i >= 1 && X[i] == 1){
        --i;
    }
    if(i == 0){
        final = true;
    }
    else{
        int tmp = cnt - i + 1;
        --X[i];
        cnt = i;
        int q = tmp / X[i];
        int r = tmp % X[i];
        if(q != 0){
            for(int j = 1; j <= q; j++){
                X[i + j] = X[i];
            }
            cnt += q;
        }
        if(r != 0){
            ++cnt;
            X[cnt] = r;
        }
    }
}
```

6. Sinh phân hoạch:

Hàm main:

```
int main(){
    n = 5;
    init();
    while(!final){
        for(int i = 1; i <= cnt; i++){
            cout << X[i] << ' ';
        }
        cout << endl;
        sinh();
    }
}
```



7. Sinh tập con bằng toán tử bit

Để sinh tập con bạn có thể sử dụng sinh nhị phân, mỗi cấu hình nhị phân tương ứng với 1 tập con của tập N phần tử. Ví dụ tập {1, 2, 3}

Cấu hình	Tập con
000	Tập rỗng
001	{3}
010	{2}
011	{2, 3}
100	{1}
101	{1,3}
110	{1,2}
111	{1,2,3}

7. Sinh tập con bằng toán tử bit

Sử dụng toán tử bit:

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int a[] = {1, 2, 3};
    int n = 3;
    for(int i = 0; i < (1 << 3); i++){
        for(int j = 0; j < 3; j++){
            if(i & (1 << j)){
                cout << a[j] << ' ';
            }
        }
        cout << endl;
    }
}
```