



Kiểu Cấu Trúc (STRUCT)



Giới thiệu về struct:

 Cấu trúc hay struct trong ngôn ngữ lập trình C++ cho phép bạn tự định nghĩa kiểu dữ liệu mới cho mình, nhằm giải quyết các bài toán khi mà thông tin đối tượng bạn cần lưu trữ không thể chỉ sử dụng kiểu dữ liệu cơ bản.

 Để lưu thông tin của một sinh viên với tên, lớp, điểm gpa... thì việc xây dựng một kiểu dữ liệu có thể lưu toàn bộ thông tin của sinh viên này là cần thiết.



1. Khai báo struct trong C:

CÚ PHÁP:

```
struct struct_name{  
    //Danh sách các thuộc tính  
};
```

- Danh sách các thuộc tính ở đây chính là các trường dữ liệu để lưu thông tin cho struct. Bạn phải chỉ ra kiểu dữ liệu của các trường dữ liệu này.

VD: Cấu trúc sinh viên có tên, lớp, điểm gpa

```
struct SinhVien{  
    string ten;  
    string lop;  
    double gpa;  
};
```

1. Khai báo struct trong C++:



Trong ngôn ngữ lập trình C++ thì struct được sử dụng để khai báo như một kiểu dữ liệu thông thường. Sau khi xây dựng struct thì bạn có thể khai báo nó như một kiểu dữ liệu thông thường.

Ví dụ:

```
#include <bits/stdc++.h>
using namespace std;

struct SinhVien{
    string ten;
    string lop;
    double gpa;
};

int main(){
    SinhVien x;
    return 0;
}
```



2. Truy cập vào các thành phần của struct:

```
#include <bits/stdc++.h>
using namespace std;
```

```
struct SinhVien{
    string ten;
    string lop;
    double gpa;
};
```

```
int main(){
    SinhVien x;
    getline(cin, x.ten);
    getline(cin, x.lop);
    cin >> x.gpa;
    cout << x.ten << ' ' << x.lop << ' ' << fixed << setprecision(2) << x.gpa;
    return 0;
}
```



Để truy cập vào các trường dữ liệu của struct ta dùng **toán tử '.'**



3. Gán 2 biến struct:

```
#include <bits/stdc++.h>
using namespace std;
```

```
struct SinhVien{
    string ten;
    string lop;
    double gpa;
};
```

```
int main(){
    SinhVien x;
    getline(cin, x.ten);
    getline(cin, x.lop);
    cin >> x.gpa;
    SinhVien y = x;
    cout << y.ten << ' ' << y.lop << ' ' << fixed << setprecision(2) << y.gpa;
    return 0;
}
```



Khi bạn gán 2 biến struct cho nhau, mọi trường dữ liệu của 2 biến này có nội dung giống nhau.



4. Mảng struct:



Sau khi xây dựng struct, bạn có thể sử dụng kiểu struct như một kiểu dữ liệu nguyên thủy.



Ví dụ: Để quản lý 1 danh sách sinh viên có không quá 1000 người bạn có thể khai báo 1 mảng SinhVien có tối đa 1000 phần tử.



4. Mảng struct:

Mã nguồn nhập và in thông tin N sinh viên từ bàn phím:

Hàm nhập thông tin

```
void nhap(SinhVien &x){  
    getline(cin, x.ten);  
    getline(cin, x.lop);  
    cin >> x.gpa;  
}
```

Hàm xuất thông tin

```
void in(SinhVien x){  
    cout << x.ten << ' ' << x.lop << ' ' << fixed << setprecision(2) << x.gpa << endl;  
}
```


4. Mảng struct:

Mã nguồn nhập và in thông tin N sinh viên từ bàn phím:

Hàm main sử dụng mảng 1 chiều:

```
int main(){
    int n; cin >> n;
    SinhVien a[1000];
    for(int i = 0; i < n; i++){
        nhap(a[i]);
    }
    for(int i = 0; i < n; i++){
        in(a[i]);
    }
    return 0;
}
```

Hàm main sử dụng vector:

```
int main(){
    int n; cin >> n;
    vector<SinhVien> a;
    for(int i = 0; i < n; i++){
        SinhVien x;
        nhap(x);
        a.push_back(x);
    }
    for(int i = 0; i < n; i++){
        in(a[i]);
    }
    return 0;
}
```

Chú ý : Các bạn không thể sử dụng map hay set đối với struct SinhVien

5. Con trỏ kiểu struct:

Tương tự như kiểu dữ liệu thông thường, bạn có thể **khai báo con trỏ struct**. Nhưng **khi truy cập vào các trường dữ liệu của biến con trỏ struct ta dùng -> thay vì dùng toán tử '.'**

Ví dụ:

```
#include <bits/stdc++.h>
using namespace std;

struct SinhVien{
    string ten;
    string lop;
    double gpa;
};

int main(){
    SinhVien x;
    SinhVien *ptr = &x;
    getline(cin, ptr->ten);
    getline(cin, ptr->lop);
    cin >> ptr->gpa;
    cout << ptr->ten << ' ' << ptr->lop << ' ' << ptr->gp;
    return 0;
}
```



6. Struct lồng nhau:

Struct sau khi xây dựng sẽ là tương tự **như một kiểu dữ liệu** vì thế bạn cũng có thể **sử dụng nó để làm thành phần cho struct khác**. Ví dụ struct SinhVien có chứa struct Date như sau:

Ví dụ:

```
#include <bits/stdc++.h>
using namespace std;

struct date{
    int day, month, year;
};

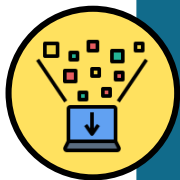
struct SinhVien{
    string ten;
    string lop;
    date ngaysinh;
    double gpa;
};

int main(){
    SinhVien x;
    getline(x.ten);
    getline(x.lop);
    cin >> x.ngaysinh.day > x.ngaysinh.month >> x.ngaysinh.year;
    cin >> x.gpa;
}
```

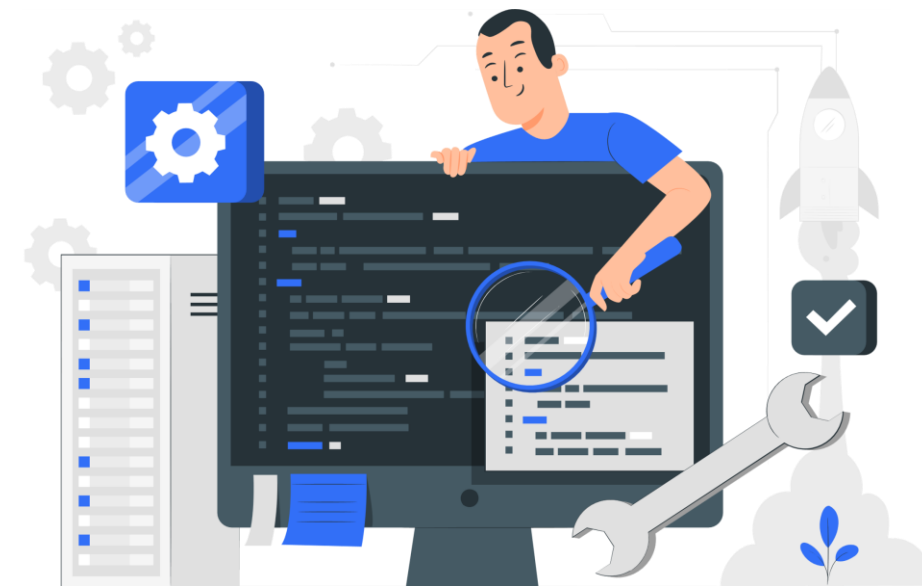
7. Hàm tạo và các hàm thành viên:



Struct có thể chứa các trường dữ liệu để mô tả thông tin của struct đó hoặc cũng có thể chứa các hàm thành phần thực hiện các chức năng của struct này.



Hàm tạo là một hàm đặc biệt trong struct, nó không có kiểu trả về và có tên trùng với tên struct. Hàm tạo được gọi khi bạn khai báo mới 1 biến struct.



7. Hàm tạo và các hàm thành viên:

a) Hàm tạo mặc định:

Trong ví dụ bên, khi bạn khai báo **x** thì **một hàm constructor mặc định** sẽ được gọi để tạo nên biến struct x.

Ví dụ:

```
#include <bits/stdc++.h>
using namespace std;

struct SinhVien{
    //data
    string ten;
    string lop;
    double gpa;
    //method
    SinhVien(){
        cout << "Day la ham khoi tao mac dinh !\n";
    }
};

int main(){
    SinhVien x;
}
```

OUTPUT

Day la ham khoi tao mac dinh !



7. Hàm tạo và các hàm thành viên:

b) Hàm tạo có đầy đủ tham số:

Bạn có thể **xây dựng hàm tạo có đầy đủ tham số** để có thể **nhANH chóng gán thông tin cho các trường dữ liệu của struct**. Dựa vào **cách khai báo biến struct** của bạn mà hàm khởi tạo mặc định hoặc khởi tạo có đầy đủ tham số sẽ được gọi.

Ví dụ:

```
#include <bits/stdc++.h>
using namespace std;
```

```
struct SinhVien{
    string ten;
    string lop;
    double gpa;
    SinhVien(){
        cout << "Day la ham khoi tao mac dinh !\n";
    }
    SinhVien(string _ten, string _lop, double _gpa){
        cout << "Day la ham khoi tao co tham so !\n";
        ten = _ten;
        lop = _lop;
        gpa = _gpa;
    }
};

int main(){
    SinhVien x; // ham tao mac dinh
    SinhVien y("28tech", "CNTT", 2.1);
    cout << y.ten << endl;
}
```

OUTPUT

```
Day la ham khoi tao mac dinh !
Day la ham khoi tao co tham so !
28tech
```



7. Hàm tạo và các hàm thành viên:

c) Các hàm hỗ trợ:

Các bạn có thể xây dựng các hàm chức năng như in thông tin, nhập thông tin, trả về dữ liệu, gán dữ liệu...

Hàm nhập và xuất cho struct:

```
struct SinhVien{  
    //data  
    string ten;  
    string lop;  
    double gpa;  
    //method  
    void nhap(){  
        getline(ten);  
        getline(lop);  
        cin >> gpa;  
    }  
    void in(){  
        cout << ten << ' ' << lop << ' ' << gpa << endl;  
    }  
};
```

Các hàm lấy thông tin (get) và sửa thông tin (set):

```
struct SinhVien{  
    //data  
    string ten;  
    string lop;  
    double gpa;  
    //method  
    string getTen(){  
        return ten;  
    }  
    void setTen(string name){  
        ten = name;  
    }  
};
```



8. Nạp chồng toán tử:

Đối với struct các bạn chỉ có thể sử dụng toán tử gán để gán 2 struct, các toán tử khác muốn sử dụng các bạn phải nạp chồng nó. Ví dụ bạn muốn nạp chồng toán tử > để so sánh 2 sinh viên theo GPA.

Sau khi nạp chồng toán tử này các bạn có thể sort mảng struct mà không cần comparator, tự động nó sẽ được sort theo gpa tăng dần.

Ví dụ:

```
#include <bits/stdc++.h>
using namespace std;
```

```
struct SinhVien{
    //data
    string ten;
    string lop;
    double gpa;
    //operator overloading
    bool operator > (SinhVien khac){
        return gpa > khac.gpa;
    }
};

int main(){
    SinhVien x{"28tech", "CNTT1", 2.5};
    SinhVien y{"dev", "DTVT2", 2.1};
    if(x > y){ // ok
        cout << "X co gpa cao hon Y" << endl;
    }
}
```

OUTPUT

X co gpa cao hon Y



8. Nạp chồng toán tử:

Nạp chồng toán tử nhập xuất

```
#include <bits/stdc++.h>
using namespace std;

struct SinhVien{
    //data
    string ten;
    string lop;
    double gpa;
    //operator overloading
    friend istream& operator >> (istream &in, SinhVien &x){
        getline(in, x.ten);
        getline(in, x.lop);
        in >> x.gpa;
        return in;
    }
    friend ostream& operator << (ostream &out, SinhVien x){
        out << x.ten << ' ' << x.lop << ' ' << x.gpa << endl;
        return out;
    }
};
```

Hàm main

```
int main(){
    SinhVien x;
    cin >> x;
    cout << x;
}
```



8. Nạp chồng toán tử:

Nạp chồng toán tử + cho struct số phức

```
#include <bits/stdc++.h>
using namespace std;

struct Comp{
    int a, b; // a + bi
    Comp operator + (Comp khac){
        Comp tong;
        tong.a = a + khac.a;
        tong.b = b + khac.b;
        return tong;
    }
};

int main(){
    Comp s1{5, 3};
    Comp s2{3, 4};
    s1 = s1 + s2;
    cout << s1.a << ' ' << s1.b << 'i' << endl;
}
```

OUTPUT

8 7i



8. Nạp chồng toán tử:

Nạp chồng toán tử + cho struct số phức

```
#include <bits/stdc++.h>
using namespace std;

struct Comp{
    int a, b; // a + bi
    friend Comp operator + (Comp u, Comp v){
        Comp tong;
        tong.a = u.a + v.a;
        tong.b = u.b + v.b;
        return tong;
    }
};

int main(){
    Comp s1{5, 3};
    Comp s2{3, 4};
    s1 = s1 + s2;
    cout << s1.a << ' ' << s1.b << 'i' << endl;
}
```

OUTPUT

8 7i