## STRUCT TRONG C

### Khái niệm cấu trúc

* Cấu trúc (struct) là một kiểu dữ liệu do người dùng tự định nghĩa, cho phép lưu trữ nhiều loại dữ liệu khác nhau trong một thực thể duy nhất.
* Mỗi phần tử trong cấu trúc được gọi là thành viên (member).
* Cấu trúc giúp quản lý dữ liệu tốt hơn khi làm việc với nhiều thuộc tính có liên quan.

### Định nghĩa cấu trúc

Cú pháp định nghĩa cấu trúc trong C:

|  |  |
| --- | --- |
| **Cú pháp** | **VD** |
| struct structName {  dataType member1;  dataType member2;  ...  dataType memberN;  }; | struct SinhVien {      int maSV;      char HoTen[50];      int NamSinh;      float diemTrungBinh;  }; |

### Khai báo biến cấu trúc

Có thể khai báo biến dựa trên **struct** theo các cách sau:

**Cách 1: Khai báo trực tiếp sau struct**

|  |
| --- |
| struct SinhVien {      int maSV;      char HoTen[50];      int NamSinh;  } SV1, SV2; // Tạo 2 biến kiểu SinhVien |

**Cách 2:** Khai báo biến riêng biệt

|  |
| --- |
| struct SinhVien {      int maSV;      char HoTen[50];      int NamSinh;  };  struct SinhVien SV1, SV2; |

**Cách 3:** Sử dụng typedef để đặt tên ngắn gọn

Bạn có thể sử dụng từ khóa typedef để tạo ra một tên thay thế cho kiểu dữ liệu đã có. Nó thường được sử dụng kiểu struct để đơn giản hóa cú pháp khai báo biến.

Cú pháp định nghĩa typedef:

|  |  |
| --- | --- |
| **Cú pháp** | **VD** |
| typedef struct structName {  dataType member1;  dataType member2;  ...  dataType memberN;  } StructNameNew; | typedef struct SinhVien {      int maSV;      char HoTen[50];      int NamSinh;      float diemTrungBinh;  } TypeSinhVien; |

Sau khi tạo ra một kiểu dữ liệu mới có tên là TypeSinhVien. Ta có thể sử dụng kiểu dữ liệu này để khai báo biến cấu trúc.

|  |
| --- |
| #include <stdio.h>  typedef struct SinhVien {      int maSV;      char HoTen[50];      int NamSinh;      float diemTrungBinh;  } TypeSinhVien;    int main() {      TypeSinhVien SV1, SV2;      return 0;  } |

### Thao tác với Cấu Trúc

#### Gán dữ liệu kiểu cấu trúc

Có 2 cách để gán dữ liệu:

Cách 1: <Biến cấu trúc đích> = <Biến cấu trúc nguồn>

Cách 2: < Biến cấu trúc đích> . <Tên thành viên> = <Giá trị>;

|  |  |
| --- | --- |
| **Cách 1** | **Cách 2** |
| #include <stdio.h>  struct SinhVien {      int maSV;      char HoTen[50];      int NamSinh;  };  int main() {      struct SinhVien SV = {17141068, "Nguyen Phi Hao", 1999};  } | #include <stdio.h>  #include <string.h>  struct SinhVien {      int maSV;      char HoTen[50];      int NamSinh;  };  int main() {      struct SinhVien SV;      SV.maSV = 17141068;      strcpy(SV.HoTen, "Nguyen Phi Hao");      SV.NamSinh = 1999;  } |

**Lưu ý:**

Ta không thể gán dữ liệu chuỗi cho kiểu cấu trúc. Mặc khác ta có thể dụng dụng hàm strcpy() trong thư viện string.h để copy dữ liệu gán vào biến.

|  |
| --- |
| SV.HoTen = "Nguyen Phi Hao"; // ERROR  strcpy(SV.HoTen, "Nguyen Phi Hao"); //OK |

#### Truy cập các thành viên kiểu cấu trúc

Ta không thể truy cập trực tiếp các thành viên của cấu trúc. Sử dụng toán tử chấm (.) để truy xuất tới thành viên của cấu trúc. Mỗi lần truy cập ta chỉ có thể truy cập đến một thành viên duy nhất.

Cú pháp:

|  |
| --- |
| < Biến cấu trúc > . <Tên thành viên> |

**VD:** Truy cập các thành viên của cấu trúc dụng toán tử dấu chấm.

|  |
| --- |
| #include <stdio.h>  struct SinhVien{      int maSV;      char HoTen[50];      int NamSinh;      float diemTB;  };    int main() {      struct SinhVien SV1;      // Nhập dữ liệu      printf("Nhap tin sinh vien:\n");      printf("Ma Sinh Vien: ");      scanf("%d", &SV1.maSV);      printf("Ho va Ten: ");      scanf("%s", SV1.HoTen);      printf("Nam sinh: ");      scanf("%d", &SV1.NamSinh);      printf("Nhap Diem TB: ");      scanf("%f", &SV1.diemTB);      // Xuất dữ liệu      printf("\nThong tin sinh vien:\n");      printf("Ma Sinh Vien: %d\n", SV1.maSV);      printf("Ho va Ten: %s\n", SV1.HoTen);      printf("Nam sinh: %d\n", SV1.NamSinh);      printf("Diem TB: %.2f\n", SV1.diemTB);      return 0;  } |

**Kết quả:**

|  |
| --- |
| Nhap tin sinh vien:  Ma Sinh Vien: 17141068  Ho va Ten: Nguyen Phi Hao  Nam sinh: 1999  Nhap Diem TB:8.5  Thong tin sinh vien:  Ma Sinh Vien: 17141068  Ho va Ten: Nguyen Phi Hao  Nam sinh: 1999  Diem TB:8.5 |

#### Gán biến cấu trúc

Toán tử gán = có thể dùng để gán biến cấu trúc.

**VD:** Gán biến SV2 = SV1

|  |
| --- |
| #include <stdio.h>  struct SinhVien{      int maSV;      char HoTen[50];      int NamSinh;  };  int main() {      struct SinhVien SV1 = { 17141068, "Nguyen Phi Hao", 1999};      printf("\nMa Sinh Vien 1: %i",SV1.maSV);      printf("\nHo va Ten 1: %s",SV1.HoTen);      printf("\nNam sinh 1: %i",SV1.NamSinh);      printf("\n");      struct SinhVien SV2;      SV2 = SV1;      printf("\nMa Sinh Vien 2: %i",SV2.maSV);      printf("\nHo va Ten 2: %s",SV2.HoTen);      printf("\nNam sinh 2: %i",SV2.NamSinh);      return 0;  } |

**Kết quả:**

|  |
| --- |
| Ma Sinh Vien 1: 17141068  Ho va Ten 1: Nguyen Phi Hao  Nam sinh 1: 1999  Ma Sinh Vien 2: 17141068  Ho va Ten 2: Nguyen Phi Hao  Nam sinh 2: 1999 |

Một số trường hợp ta không gán được biến cấu trúc thì ta sử dụng hàm memcpy() trong thư viện string.h

Cú pháp hàm memcpy():

|  |
| --- |
| memcpy(char\* destn, char &source, int nbytes); |

**VD:** hàm memcpy() trong thư viện string.h

|  |
| --- |
| memcpy(&SV2, &SV1, sizeof(struct SinhVien )); |

**VD:** Sử dụng hàm memcpy() để gán giá trị SV1 cho SV2.

|  |
| --- |
| #include <stdio.h>  #include <string.h>  struct SinhVien{      int maSV;      char HoTen[50];      int NamSinh;  };  int main() {      struct SinhVien SV1 = { 17141068, "Nguyen Phi Hao", 1999};      printf("\nMa Sinh Vien 1: %i",SV1.maSV);      printf("\nHo va Ten 1: %s",SV1.HoTen);      printf("\nNam sinh 1: %i",SV1.NamSinh);      printf("\n");      struct SinhVien SV2;      memcpy(&SV2, &SV1, sizeof(struct SinhVien));      printf("\nMa Sinh Vien 2: %i",SV2.maSV);      printf("\nHo va Ten 2: %s",SV2.HoTen);      printf("\nNam sinh 2: %i",SV2.NamSinh);      return 0;  } |

**Kết quả:**

|  |
| --- |
| Ma Sinh Vien 1: 17141068  Ho va Ten 1: Nguyen Phi Hao  Nam sinh 1: 1999  Ma Sinh Vien 2: 17141068  Ho va Ten 2: Nguyen Phi Hao  Nam sinh 2: 1999 |

#### Nhập dữ liệu và hiển thị cho cấu trúc

Ta có thể nhập dữ liệu và hiển thị cho một cấu trúc.

**Lưu ý:**

Ta nên sử dụng hàm fgets() trong thư viện string.h để nhập dữ liệu chuỗi cho kiểu cấu trúc và sử dụng hàm fflush(stdin) trong thư viện stdio.h để xoá bộ đệm trước khi nhập chuỗi.

**VD:** Nhập dữ liệu và hiển thị cho cấu trúc, viết trong hàm main.

|  |
| --- |
| #include <stdio.h>  #include <string.h>  struct SinhVien{      int maSV;      char HoTen[50];      int NamSinh;  };  int main() {      struct SinhVien SV;      // Nhập dữ liệu cấu trúc      printf("Nhap MSSV: ");      scanf("%i", &SV.maSV);      printf("Nhap Ho va Ten: ");      fflush(stdin);      fgets(SV.HoTen, sizeof(SV.HoTen), stdin);      printf("Nhap nam sinh: ");      scanf("%i", &SV.NamSinh);      //Hiển thị dữ liệu cấu trúc      printf("\nMa Sinh Vien: %i",SV.maSV);      printf("\nHo va Ten: %s",SV.HoTen);      printf("Nam sinh: %i",SV.NamSinh);      return 0;  } |

**Kết quả:**

|  |
| --- |
| Nhap MSSV: 17141068  Nhap Ho va Ten: Nguyen Phi Hao  Nhap nam sinh: 1999  Ma Sinh Vien: 17141068  Ho va Ten: Nguyen Phi Hao  Nam sinh: 1999 |

### Mảng trong Cấu Trúc

Mảng cấu trúc là một mảng chứa các phần tử có kiểu là một cấu trúc (struct).

Dùng để lưu trữ nhiều đối tượng có cùng kiểu dữ liệu nhưng mỗi đối tượng có thể chứa nhiều thông tin khác nhau.

**C code:**

|  |
| --- |
| struct Taikhoan {      char Ten[100];      // Lưu tên      char Ho[100];       // Lưu họ      char DiaChi[100];   // Lưu địa chỉ      int Tuoi;           // Tuổi      int GioiTinh;       // Giới tính (1 = Nam, 0 = Nữ)  }; |

Cấu trúc Taikhoan chứa 5 biến thành phần, trong đó:

* 3 biến đầu tiên kiểu char lưu trữ các thông tin lần lượt là: Tên, Họ, Địa chỉ.
* 2 biến còn lại kiểu int lưu trữ các thông tin: Tuổi và Giới tính.

**VD:** Nhập thông tin cho 3 sinh viên và hiển thị thông tin của các sinh viên.

|  |
| --- |
| #include <stdio.h>  struct SinhVien {      int maSV;      char HoTen[50];      int NamSinh;  };  int main() {      int i;      struct SinhVien SV[3];      printf("Nhap thong tin cho 3 sinh vien:\n");      for (i = 1; i <= 3; i++) {          printf("Nhap ma SV[%i]: ", i);          scanf("%i", &SV[i].maSV);          getchar();  // Xóa bộ đệm phím Enter          printf("Nhap Ho va Ten SV[%i]: ", i);          fgets(SV[i].HoTen, sizeof(SV[i].HoTen), stdin);          printf("Nhap nam sinh SV[%i]: ", i);          scanf("%i", &SV[i].NamSinh);      }      printf("\nDanh sach sinh vien:\n");      for (i = 1; i <= 3; i++) {          printf("ID: %d - Ho Ten: %s - Nam Sinh: %i\n", SV[i].maSV, SV[i].HoTen, SV[i].NamSinh);      }      return 0;  } |

**Kết quả**:

|  |
| --- |
| Nhap thong tin cho 3 sinh vien:  Nhap ma SV[1]: 0001  Nhap Ho va Ten SV[1]: Nguyen Phi Hao  Nhap nam sinh SV[1]: 1999  Nhap ma SV[2]: 0002  Nhap Ho va Ten SV[2]: Phan Thi My Hoa  Nhap nam sinh SV[2]: 2000  Nhap ma SV[3]: 0003  Nhap Ho va Ten SV[3]: Nguyen Phu Hung  Nhap nam sinh SV[3]: 2022  Danh sach sinh vien:  ID: 0001 - Ho Ten: Nguyen Phi Hao - Nam Sinh: 1999  ID: 0002 - Ho Ten: Phan Thi My Hoa - Nam Sinh: 2000  ID: 0003 - Ho Ten: Nguyen Phu Hung - Nam Sinh: 2022 |

**Lưu ý:**

Dùng fgets() để nhập chuỗi có khoảng trắng.

Dùng getchar() để loại bỏ ký tự Enter sau scanf().

### Cấu Trúc lồng nhau trong C

#### Khái Niệm

struct trong C là kiểu dữ liệu do người dùng tự định nghĩa.

Có thể sử dụng một struct bên trong một struct khác, gọi là cấu trúc lồng nhau (Nested Struct).

Có 2 cách chính để lồng struct:

1. **Cấu trúc lồng riêng biệt** (Tạo cấu trúc bên ngoài, rồi sử dụng lại bên trong struct chính).
2. **Cấu trúc nhúng** (Định nghĩa struct con trực tiếp bên trong struct chính).

#### Cấu trúc lồng riêng biệt

Cấu trúc phụ (struct con) được khai báo riêng rồi sử dụng bên trong struct chính. Có thể tái sử dụng struct con nhiều lần.

**VD:** Cấu trúc birthday được sử dụng bên ngoài và như một thành viên của cấu trúc Student

|  |
| --- |
| #include <stdio.h>  // Cấu trúc phụ  struct Date {      int day;      int month;      int year;  };  // Cấu trúc chính  struct Student {      int id;      char fullName[50];      struct Date birthday;  // Dùng cấu trúc Date  };  int main() {      struct Student sv1;      // Nhập dữ liệu      printf("Nhap ID: ");      scanf("%d", &sv1.id);      getchar();      printf("Nhap ho va ten: ");      fgets(sv1.fullName, sizeof(sv1.fullName), stdin);      printf("Nhap ngay sinh (dd mm yyyy): ");      scanf("%d %d %d", &sv1.birthday.day, &sv1.birthday.month, &sv1.birthday.year);      // Hiển thị dữ liệu      printf("\nID: %d\n", sv1.id);      printf("Ho Ten: %s", sv1.fullName);      printf("Ngay sinh: %02d/%02d/%d\n", sv1.birthday.day, sv1.birthday.month, sv1.birthday.year);      return 0;  } |

**Ưu điểm của cấu trúc lồng riêng biệt**

* **Tái sử dụng** struct Date cho nhiều cấu trúc khác như Employee, Book,...
* **Dễ bảo trì**: Nếu cần sửa struct Date, chỉ cần thay đổi một nơi.

#### Cấu Trúc nhúng

struct con được định nghĩa trực tiếp bên trong struct chính. Không thể tái sử dụng struct con ngoài struct chính.

**VD:** Cấu trúc birthday được nhúng trong cấu trúc Student

|  |
| --- |
| #include <stdio.h>  struct Student {      int id;      char fullName[50];        // Cấu trúc nhúng trực tiếp bên trong      struct {          int day;          int month;          int year;      } birthday;  };  int main() {      struct Student sv1;      // Nhập dữ liệu      printf("Nhap ID: ");      scanf("%d", &sv1.id);      getchar();      printf("Nhap ho va ten: ");      fgets(sv1.fullName, sizeof(sv1.fullName), stdin);      printf("Nhap ngay sinh (dd mm yyyy): ");      scanf("%d %d %d", &sv1.birthday.day, &sv1.birthday.month, &sv1.birthday.year);      // Hiển thị dữ liệu      printf("\nID: %d\n", sv1.id);      printf("Ho Ten: %s", sv1.fullName);      printf("Ngay sinh: %02d/%02d/%d\n", sv1.birthday.day, sv1.birthday.month, sv1.birthday.year);      return 0;  } |

**Ưu điểm của cấu trúc nhúng**

* **Đơn giản hơn** nếu struct con chỉ được dùng trong struct chính.
* **Giảm số lượng khai** báo không cần thiết.

### Mối liên hệ giữa Cấu Trúc và Con trỏ

Tương tự như các kiểu dữ liệu khác trong C, chúng ta có thể khai báo **con trỏ kiểu struct** và sử dụng cấp phát động để quản lý bộ nhớ.

#### Khai báo biến con trỏ kiểu Cấu Trúc

Ta có 2 cách để khai báo biến cấu trúc:

**Cách 1:** Sử dụng từ khoá struct trong hàm main().

**Cách 2:** Khai báo biến cấu trúc tại thời điểm định nghĩa cấu trúc.

**VD:** Cách 1

|  |
| --- |
| struct SinhVien {      int maSV;      char HoTen[50];      int NamSinh;  };  int main() {      struct SinhVien \*SV1, SV2;  // SV1 là con trỏ, SV2 là biến thông thường  } |

**VD:** Cách 2

|  |
| --- |
| struct SinhVien {      int maSV;      char HoTen[50];      int NamSinh;  } \*SV1, SV2;  // SV1 là con trỏ, SV2 là biến thông thường |

Cách này giúp **viết gọn hơn**, nhưng không linh hoạt bằng cách 1.

#### **Truy cập các thành viên kiểu Cấu Trúc bằng con trỏ**

Để truy cập vào các thành viên sử dụng biến con trỏ của struct trong C, ta sử dụng toán tử ->

**VD:**

|  |
| --- |
| #include <stdio.h>  struct Person {     int age;     float weight;  };  int main() {      struct Person \*personPtr, person1;      // Trỏ con trỏ vào biến person1      personPtr = &person1;      // Nhập dữ liệu      printf("Enter age: ");      scanf("%i", &personPtr->age);      printf("Enter weight: ");      scanf("%f", &personPtr->weight);      // Hiển thị dữ liệu      printf("\nDisplaying:\n");      printf("Age: %i\n", personPtr->age);      printf("Weight: %.2f\n", personPtr->weight);      return 0;  } |

***Giải thích:***

* personPtr->age tương đương với (\*personPtr).age
* personPtr->weight tương đương với (\*personPtr).weight
* Dùng -> giúp **code ngắn gọn và dễ đọc hơn**.

### Mối liên hệ giữa struct và hàm

struct có thể được **truyền vào hàm** theo hai cách:

1. **Truyền bằng giá trị** (tạo bản sao, tốn bộ nhớ).
2. **Truyền bằng con trỏ** (tham chiếu trực tiếp, tối ưu hơn).

#### ****Truyền Cấu Trúc vào Hàm****

Trường hợp nếu ta muốn tách thành các hàm nhập liệu và hiển thị riêng thì ta làm như sau:

**VD:** Nhập dữ liệu và hiển thị cho cấu trúc, sử dụng các hàm con.

|  |
| --- |
| #include <stdio.h>  struct SinhVien {      char MaSV[50];      char HoTen[50];      int NamSinh;  };  // Nhập dữ liệu  void nhapLieu(struct SinhVien \*SV) {      printf("Nhap MSSV: ");      fgets(SV->MaSV, sizeof(SV->MaSV), stdin);      printf("Nhap Ho va Ten: ");      fgets(SV->HoTen, sizeof(SV->HoTen), stdin);      printf("Nhap nam sinh: ");      scanf("%i", &SV->NamSinh);      getchar(); // Xóa bộ đệm  }  // Hiển thị dữ liệu  void hienThi(struct SinhVien \*SV) {      printf("\nThong tin sinh vien:\n");      printf("MSSV: %s", SV->MaSV);      printf("Ho va Ten: %s", SV->HoTen);      printf("Nam Sinh: %i\n", SV->NamSinh);  }  int main() {      struct SinhVien SV;      nhapLieu(&SV);      hienThi(&SV);      return 0;  } |

**Kết quả:**

|  |
| --- |
| Nhap thong tin sinh vien.  Nhap MSSV: 17141068  Nhap Ho va Ten: Nguyen Phi Hao  Nhap nam sinh: 1999  Hien thi thong tin sinh vien.  Ma Sinh Vien: 17141068  Ho va Ten: Nguyen Phi Hao  Nam sinh: 1999 |

***Giải Thích***

* Dùng con trỏ (struct SinhVien \*SV) giúp tránh tạo bản sao, tiết kiệm bộ nhớ.
* getchar() dùng để xóa bộ đệm, tránh lỗi nhập dữ liệu.

#### ****Trả về Cấu Trúc từ Hàm****

Chúng ta có thể **trả về một struct từ một hàm**.

|  |
| --- |
| #include <stdio.h>  struct SinhVien {      char HoTen[50];      int NamSinh;  };  // Hàm trả về một struct SinhVien  struct SinhVien taoSinhVien() {      struct SinhVien sv;      printf("Nhap Ho va Ten: ");      fgets(sv.HoTen, sizeof(sv.HoTen), stdin);      printf("Nhap nam sinh: ");      scanf("%i", &sv.NamSinh);      return sv;  }  int main() {      struct SinhVien SV = taoSinhVien();      printf("\nHo Ten: %s", SV.HoTen);      printf("Nam Sinh: %i\n", SV.NamSinh);      return 0;  } |

⮕ Hàm taoSinhVien() tạo và **trả về một struct**, giúp chương trình gọn hơn.

### Cấp phát bộ nhớ động trong Cấu Trúc

Trước khi bạn đọc phần này, mình hi vọng các bạn đã có kiến thức về cấp phát động trong C.

Đôi khi, số lượng biến struct trong C mà chúng ta cần có thể lớn. Khi đó có thể bạn sẽ cần tới cấp phát động trong quá trình chương trình thực thi.

#### Cấp phát động cho Cấu Trúc

Dùng **malloc()** để cấp phát bộ nhớ cho struct.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  struct Person {     int age;     float weight;     char name[30];  };  int main() {     struct Person \*ptr;     // Cấp phát động cho 1 struct     ptr = (struct Person\*) malloc(sizeof(struct Person));     printf("Nhap ten: ");     scanf("%s", ptr->name);     printf("Nhap tuoi: ");     scanf("%i", &ptr->age);     printf("\nThong tin:\n");     printf("Ten: %s - Tuoi: %i\n", ptr->name, ptr->age);     free(ptr); // Giải phóng bộ nhớ     return 0;  } |

#### Cấp phát động cho mảng Cấu Trúc

Dùng malloc() để cấp phát cho **mảng** struct.

**VD: Nhập và hiển thị danh sách n người**

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  struct SinhVien {      char HoTen[50];      int NamSinh;  };  int main() {      int n;      printf("Nhap so luong sinh vien: ");      scanf("%d", &n);      getchar(); // Xóa bộ đệm      // Cấp phát động cho mảng struct      struct SinhVien \*SV = (struct SinhVien\*) malloc(n \* sizeof(struct SinhVien));      // Nhập thông tin      for(int i = 0; i < n; i++) {          printf("\nNhap ten SV %d: ", i+1);          fgets((SV+i)->HoTen, sizeof((SV+i)->HoTen), stdin);          printf("Nhap nam sinh: ");          scanf("%d", &(SV+i)->NamSinh);          getchar(); // Xóa bộ đệm      }      // Hiển thị danh sách      printf("\nDanh sach sinh vien:\n");      for(int i = 0; i < n; i++) {          printf("Ho Ten: %s", (SV+i)->HoTen);          printf("Nam Sinh: %d\n", (SV+i)->NamSinh);      }      free(SV); // Giải phóng bộ nhớ      return 0;  } |

***Giải Thích***

* malloc(n \* sizeof(struct SinhVien)) cấp phát động cho **n sinh viên**.
* (SV+i)->HoTen tương đương với SV[i].HoTen.