#### 1. Struct trong C

##### 1.1. Khái niệm cấu trúc

* Cấu trúc (struct) là một kiểu dữ liệu người dùng tự định nghĩa cho phép lưu trữ các loại phần tử có kiểu dữ liệu khác nhau. Mỗi phần tử của một cấu trúc được gọi là một thành viên (member).
* Từ khoá struct được sử dụng để xác định cấu trúc.

##### 1.2. 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;  }; |

##### 1.3. Khai báo biến cấu trúc

Có 2 cách để khai báo biến cấu trúc:

**Cách 1:** Khai báo biến cấu trúc ngoài định nghĩa cấu trúc ( trong hàm main hoặc local)

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

**VD:** Định nghĩa một kiểu một kiểu dữ liệu cấu trúc có tên là SinhVien

|  |  |
| --- | --- |
| **Cách 1** | **Cách 2** |
| struct SinhVien {      int maSV;      char HoTen[50];      int NamSinh;  };  struct SinhVien SV1, SV2; | struct SinhVien {      int maSV;      char HoTen[50];      int NamSinh;  } SV1, SV2; |

***Sử dụng cách khai báo nào tốt hơn ?***

Khi số lượng biến không cố định, sử dụng phương pháp tiếp cận thứ nhất. Nó linh hoạt để khai báo biến cấu trúc nhiều lần. Khi số lượng biến được cố định, sử dụng phương pháp thứ 2.

##### 1.4. Định nghĩa cấu trúc dùng typedef

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;  } typeName; | typedef struct SinhVien {      int maSV;      char HoTen[50];      int NamSinh;  } 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;  } TypeSinhVien;    int main() {      TypeSinhVien SV1, SV2;      return 0;  } |

#### 2. Thao tác với cấu trúc

##### 2.1. 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 |

##### 2.2. 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 truy cập:

|  |
| --- |
| < 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;  };    int main() {      struct SinhVien SV = {17141068, "Nguyen Phi Hao", 1999};      printf("\nMa Sinh Vien: %i",SV.maSV);      printf("\nHo va Ten: %s",SV.HoTen);      printf("\nNam sinh: %i",SV.NamSinh);      return 0;  } |

**Kết quả:**

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

##### 2.3. Gán biến cấu trúc

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

**VD:** Gán biến SV1 = SV2

|  |
| --- |
| #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 |

##### 2.4. 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 |

#### 3. Mảng cấu trúc

Những cấu trúc có chứa mảng. Thật may mắn, chúng ta hoàn toàn có thể tạo các mảng cơ bản và mảng ký tự (string) trong cấu trúc.

Nào bây giờ giả sử chúng ta có một cấu trúc *Taikhoan* để lưu trữ thông tin của một người dùng:

**C code:**

|  |
| --- |
| struct Taikhoan {      char ten[100];      char ho[100];      char diachi[1000];      int tuoi;      int gioitinh; // Boolean : 1 = nam, 0 = nu  }; |

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. Riêng giới tính là một biến dạng boolean (bạn đã học ở các bài trước về boolean), biến này sẽ trả về 1 = đúng = giới tính là nam, trả về 0 = sai = giới tính là nữ, chúng ta tạm chia ra 2 giới tính thôi nhé ^^!

Bạn có thể ứng dụng cấu trúc này để tạo một chương trình lưu trữ danh sách người dùng. Dĩ nhiên là bạn có thêm một số biến khác để bổ sung những thông tin mà bạn muốn. Không có giới hạn số lượng biến trong một cấu trúc nên bạn đừng lo.

Sử dụng mảng cấu trúc trong C để lưu trữ nhiều thông tin của các loại dữ liệu khác nhau.

**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 = 0; i < 3; i++) {          printf("\nNhap ma SV[%i]: ", i);          scanf("%i", &SV[i].maSV);          printf("Nhap Ho va Ten SV[%i]: ", i);          fflush(stdin);          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 = 0; 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[0]: 86  Nhap Ho va Ten SV[0]: Nguyen Phi Hao  Nhap nam sinh SV[0]: 1999  Nhap ma SV[1]: 78  Nhap Ho va Ten SV[1]: Phan Thi My Hoa  Nhap nam sinh SV[1]: 2000  Nhap ma SV[2]: 86  Nhap Ho va Ten SV[2]: Nguyen Phu Hung  Nhap nam sinh SV[2]: 2022  Danh sach sinh vien:  Id: 86 - Ho Ten: Nguyen Phi Hao - Nam Sinh: 1999  Id: 78 - Ho Ten: Phan Thi My Hoa - Nam Sinh: 2000  Id: 86 - Ho Ten: Nguyen Phu Hung - Nam Sinh: 2022 |

**Lưu ý:**

Khi ta nhập dữ liệu có khoảng trắng cho các biến, ta sử dụng hàm fgets().

**VD:** Nhập số lượng sinh viên và hiển thị thông tin của các sinh viên.

|  |
| --- |
| #include<stdio.h>  #include<string.h>    struct sinhVien {      int maSV;      char HoTen[50];      int NamSinh;  };    int main() {      int i, n;      printf("Nhap so luong sinh vien: ");      scanf("%d", &n);      struct sinhVien SV[n];      printf("\nNhap thong tin cho %i sinh vien:", n);      for (i = 0; i < n; i++) {          printf("\nNhap ma SV[%i]: ", i);          scanf("%i", &SV[i].maSV);          printf("Nhap Ho va Ten SV[%i]: ", i);          fflush(stdin);          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 = 0; i < n; 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 so luong sinh vien: 2  Nhap thong tin cho 2 sinh vien:  Nhap ma SV[0]: 86  Nhap Ho va Ten SV[0]: Nguyen Phi Hao  Nhap nam sinh SV[0]: 1999  Nhap ma SV[1]: 78  Nhap Ho va Ten SV[1]: Phan Thi My Hoa  Nhap nam sinh SV[1]: 2000  Danh sach sinh vien:  Id: 86 - Ho Ten: Nguyen Phi Hao - Nam Sinh: 1999  Id: 78 - Ho Ten: Phan Thi My Hoa - Nam Sinh: 2000 |

#### 4. Cấu trúc struct lồng nhau

struct trong C (cấu trúc trong C) là một kiểu dữ liệu người dùng tự định nghĩa. Nên hoàn toàn có thể sử dụng structure bên trong structure khác, nó được biết đến như structure lồng nhau trong C. Có 2 cách để định nghĩa cấu trúc lồng nhau trong C:

* Cách 1: Theo cấu trúc lồng riêng biệt.
* Cách 2: Theo cấu trúc nhúng.

##### 4.1. Cấu trúc lồng riêng biệt

Ta tạo 1 cấu trúc phụ bên ngoài và cấu trúc phụ thuộc được sử dụng bên trong cấu trúc chính như một thành viên.

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

|  |
| --- |
| struct Date {      int day;      int month;      int year;  };  struct Student {      int id;      char fullName[50];      struct Date birthday;  } SV; |

Ưu điểm của cấu trúc lồng riêng biệt là ta có thể sử dụng cấu trúc phụ lại nhiều lần.

##### 4.2. Cấu trúc nhúng.

Cấu trúc nhúng là viêc định nghĩa cấu trúc phụ bên trong một cấu trúc chính.

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

|  |
| --- |
| struct Student {      int id;      char fullName[50];      struct Date {          int day;          int month;          int year;      } birthday;  } SV; |

##### 4.3. Truy cập cấu trúc lồng nhau

Truy cập các thành viên của cấu trúc lồng nhau bởi toán tử dấu chấm.

**VD:** Truy cập các thành viên của cấu trúc lồng.

|  |
| --- |
| #include <stdio.h>  #include <string.h>    struct Student {      int id;      char fullName[50];      struct Date {          int day;          int month;          int year;      } birthday;  } SV;  int main() {      // Thông tin sinh viên      SV.id = 17141068;      strcpy(SV.fullName, "Nguyen Phi Hao");      SV.birthday.day = 25;      SV.birthday.month = 12;      SV.birthday.year = 1999;        //Truy cập và hiển thị thông tin sinh viên ra màn hình      printf("Ma so sinh vien: %i\n", SV.id);      printf("Ten sinh vien: %s\n", SV.fullName);      printf("Ngay: %i - Thang: %i - Nam: %i\n", SV.birthday.day,SV.birthday.month, SV.birthday.year);      return 0;  } |

**Kết quả**:

|  |
| --- |
| Ma so sinh vien: 17141068  Ten sinh vien: Nguyen Phi Hao  Ngay: 25 - Thang: 12 - Nam: 1999 |

#### 5. Mối liên hệ giữa struct và con trỏ

Tương tự như khai báo con trỏ với các kiểu dữ liệu có sẵn trong C. Chúng ta cũng có thể khai báo biến con trỏ, cấp phát động cho biến con trỏ kiểu struct.

##### 5.1. Khai báo biến con trỏ kiểu cấu trúc

Tương tự như biến thông thường, ta cũng 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:** Định nghĩa một kiểu một kiểu dữ liệu cấu trúc có tên là SinhVien

|  |  |
| --- | --- |
| **Cách 1** | **Cách 2** |
| struct SinhVien {      int maSV;      char HoTen[50];      int NamSinh;  };    int main() {      struct SinhVien \*SV1, SV2;  } | struct SinhVien {      int maSV;      char HoTen[50];      int NamSinh;  } \*SV1, SV2; |

Trong đó SV1 là con trỏ kiểu SinhVien, còn SV2 là biến kiểu SinhVien.

##### 5.2. 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;      personPtr = &person1;      printf("Enter age: ");      scanf("%i", &personPtr->age);      printf("Enter weight: ");      scanf("%f", &personPtr->weight);      printf("Displaying:\n");      printf("Age: %i\n", personPtr->age);      printf("weight: %f", personPtr->weight);      return 0;  } |

Trong ví dụ này, địa chỉ của biến person1 được lưu giữ bởi con trỏ personPtr. Và bạn có thể thao tác với biến con trỏ giống như chúng ta đã học ở bài [Con trỏ trong C](https://nguyenvanhieu.vn/con-tro-trong-c/).

Ta có:

personPtr->age cho kết quả giống với (\*personPtr).age

personPtr->weight cho kết quả giống với (\*personPtr).weight

#### 6. Mối liên hệ giữa struct và hàm

##### **6.1. 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 cấu trúc  void nhapLieu(struct SinhVien \*SV){      printf("Nhap thong tin sinh vien.\n");      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);      fflush(stdin);  }  //Hiển thị dữ liệu cấu trúc  void hienThi(struct SinhVien \*SV){      printf("\nHien thi thong tin sinh vien.\n");      printf("Ma Sinh Vien: %s",SV->MaSV);      printf("Ho va Ten: %s",SV->HoTen);      printf("Nam sinh: %i",SV->NamSinh);  }    int main() {      struct SinhVien SV;      nhapLieu(&SV);      hienThi(&SV);      return 0;  } |

**Kết quả:**

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

##### **6.2. Lấy lại cấu trúc từ một hàm**

##### **6.3. Truyền cấu trúc bằng cách tham chiếu**

Bạn cũng có thể truyền cấu trúc bằng cách tham chiếu (tương tự như cách mà bạn truyền biến thuộc kiểu xây dựng sẵn bằng cách tham chiếu). Tôi khuyên bạn nên xem lại phần truyền bằng cách tham chiếu trước khi xem những thông tin tiếp theo.

Trong quá trình truyền bằng cách tham chiếu, địa chỉ bộ nhớ của các biến struct trong C được truyền vào hàm.

#### 8. Cấp phát bộ nhớ động trong cấu trúc --- chưa làm

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.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  struct person {     int age;     float weight;     char name[30];  };  int main()  {     struct person \*ptr;     int i, n;     printf("Enter the number of persons: ");     scanf("%i", &n);     // allocating memory for n numbers of struct person     ptr = (struct person\*) malloc(n \* sizeof(struct person));     for(i = 0; i < n; ++i)     {         printf("Enter first name and age respectively: ");         // To access members of 1st struct person,         // ptr->name and ptr->age is used         // To access members of 2nd struct person,         // (ptr+1)->name and (ptr+1)->age is used         scanf("%s %i", (ptr+i)->name, &(ptr+i)->age);     }     printf("Displaying Information:\n");     for(i = 0; i < n; ++i)         printf("Name: %s\tAge: %i\n", (ptr+i)->name, (ptr+i)->age);     return 0;  } |

**Kết quả:**

|  |
| --- |
| Enter the number of persons:  2  Enter first name and age respectively:  Harry 24  Enter first name and age respectively:  Gary 32  Displaying Information:  Name: Harry Age: 24  Name: Gary  Age: 32 |

Trong ví dụ trên, sau khi người dùng nhập số lượng n thì ta mới tiến hành cấp phát đúng n ô nhớ sử dụng dòng lệnh này:

|  |
| --- |
| ptr = (struct person\*) malloc(n \* sizeof(struct person)); |

Và sau đó, ta dùng con trỏ ptr để truy cập vào các thành viên của person.

Hoặc bài tập cấp phát động cho kiểu cấu trúc sinh viên dưới đây, ta thực hiện nhập, xuất và sắp xếp danh sách sinh viên theo điểm sử dụng cấp phát động cho con trỏ trong C++ (new và delete):

|  |
| --- |
| #include<iostream>  #include<stdio.h>  #include<math.h>  #include<string.h>    using namespace std;    struct NGAYTHANG {      int ngay;      int thang;      int nam;  };  struct SV {      char masv[12];      char hodem[30];      char ten[10];      NGAYTHANG ngsinh;      char gioitinh[4];      char hokhau[20];      float diem;  };  void Nhapsv(SV \* sv) {      cout << "\tMa sv: ";      cin >> sv->masv;      cout << "\tHo dem: ";      cin.ignore();      fgets(sv->hodem, sizeof(sv->hodem), stdin);      sv->hodem[strlen(sv->hodem) - 1] = '\0';      cout << "\tTen: ";      cin.ignore();      fgets(sv->ten, sizeof(sv->ten), stdin);      cout << "\tNgay sinh: ";      cin >> sv->ngsinh.ngay;      cout << "\tThang sinh: ";      cin >> sv->ngsinh.thang;      cout << "\tNam sinh: ";      cin >> sv->ngsinh.nam;      cout << "\tGioi tinh: ";      cin.ignore();      fgets(sv->gioitinh, sizeof(sv->gioitinh), stdin);      sv->gioitinh[strlen(sv->gioitinh) - 1] = '\0';      cout << "\tHo khau: ";      cin.ignore();      fgets(sv->hokhau, sizeof(sv->hokhau), stdin);      sv->hokhau[strlen(sv->hokhau) - 1] = '\0';      cout << "\tDiem: ";      cin >> sv->diem;    }  void Hienthisv(SV \* sv) {      cout << sv->masv;      cout << "\t" << sv->hodem;      cout << " " << sv->ten;      cout << "\t" << sv->ngsinh.ngay;      cout << "-" << sv->ngsinh.thang;      cout << "-" << sv->ngsinh.nam;      cout << "\t" << sv->gioitinh;      cout << "\t" << sv->hokhau;      cout << "\t" << sv->diem;  }  void Nhapds(SV \*p, int n) {      for (int i = 0; i < n; i++) {          cout << "Nhap thong tin cua sv thu " << i + 1 << " :" << endl;          Nhapsv(p + i);        }  }  void Hienthids(SV \*p, int n) {      for (int i = 0; i < n; i++) {            Hienthisv(p + i);          cout << "\n";      }  }  void Sapxep(SV \*p, int n) {      for (int i = 0; i < n; i++) {          for (int j = i + 1; j < n; j++)              if ((p + i)->diem > (p + j)->diem) {                  SV tmp = \* (p + j);                  \*(p + j) = \* (p + i);                  \*(p + i) = tmp;              }      }  }  int main() {      SV \*p;      int n;      do {          cout << "Nhap vao so sv: ";          cin >> n;      }      while (n < 0 || n > 10);      p = new SV[n];      cout << "Nhap vao thong tin " << n << " sv: " << endl;      Nhapds(p, n);      cout << "Hien thi thong tin vua nhap: " << endl;      Hienthids(p, n);      cout << "\nDanh sach sau khi sap xep la: " << endl;      Sapxep(p, n);      Hienthids(p, n);      delete p;      return 0;  } |

**Kết quả :**

|  |
| --- |
| Nhap vao so sv: 3  Nhap vao thong tin 3 sv:  Nhap thong tin cua sv thu 1 :          Ma sv: 5          Ho dem: 5          Ten: 5          Ngay sinh: 5          Thang sinh: 5          Nam sinh: 5          Gioi tinh: 5          Ho khau: 5          Diem: 5  Nhap thong tin cua sv thu 2 :          Ma sv: 3          Ho dem: 3          Ten: 3          Ngay sinh: 3          Thang sinh: 3          Nam sinh: 3          Gioi tinh: 3          Ho khau: 3          Diem: 3  Nhap thong tin cua sv thu 3 :          Ma sv: 9          Ho dem: 9          Ten: 9          Ngay sinh: 9          Thang sinh: 9          Nam sinh: 9          Gioi tinh: 9          Ho khau: 9          Diem: 9  Hien thi thong tin vua nhap:  5       5          5-5-5   5               5  3       3          3-3-3   3               3  9       9          9-9-9   9               9    Danh sach sau khi sap xep la:  3       3          3-3-3   3               3  5       5          5-5-5   5               5  9       9          9-9-9   9               9 |

Ngoài ra, mình cũng có một bài hướng dẫn chi tiết và nâng cao hơn: [Bài tập quản lý sinh viên](https://nguyenvanhieu.vn/chuong-trinh-quan-ly-sinh-vien-su-dung-struct/) sử dụng struct trong C. Các bạn tiếp tục tham khảo bài viết này nhé!

Tài liệu tham khảo

<http://uet.vnu.edu.vn/~diepht/s12_int2202/lect06_struct_class.pdf>

[C Struct (Structure)](https://www.programiz.com/c-programming/c-structures)