

Cấu trúc

huydq@soict.hust.edu.vn

Khái niệm cấu trúc

- Thực tế chúng ta thường thao tác với các thực thể logic bao gồm một số thông tin đi theo nhóm
 - Ví dụ, số phức bao gồm giá trị phần thực và ảo, bản ghi 1 sinh viên chứa thông tin về tên, điểm thi, v.v
 - Mỗi thực thể như thế này được cấu tạo từ nhiều biến, nhưng có thể quản lí như 1 đơn vị logic
- Một cấu trúc (**struct**) là một tập hợp của một số biến khác kiểu dữ liệu mà được đóng gói vào một khối hợp nhất
- Một khai báo cấu trúc cho phép định một kiểu dữ liệu phức hợp cho các biến kiểu cấu trúc
- Tên của các biến trong một cấu trúc còn được gọi là trường cấu trúc

Khai báo cấu trúc

```
struct complex  
{  
    int real;  
    int img;  
};
```

Khai báo tên của
cấu trúc

Các trường của
cấu trúc

```
struct studentRec  
{  
    char name[80];  
    int mark;  
};
```

Đừng quên dấu ; sau
khai báo cấu trúc



Khai báo này mới chỉ tạo kiểu của cấu trúc chứ chưa có biến

Khai báo biến cấu trúc

- Để tạo một cấu trúc trong bộ nhớ máy tính, ta cần khai báo một biến cấu trúc như sau

```
struct complex num;  
struct studentRec john;
```



Tên kiểu cấu
trúc

Tên của
biến

Kết hợp khai báo

- Có thể kết hợp khai báo kiểu cấu trúc và biến trên cùng một dòng lệnh (nhưng ít khi sử dụng vì kiểu cấu trúc cần dùng nhiều khai báo biến khác nhau)

```
struct complex
{
    int real;
    int img;
} num;
```

```
struct studentRec
{
    char name[80];
    int mark;
} john;
```

Truy nhập cấu trúc


- Để truy nhập vào 1 trường của biến cấu trúc ta dùng **toán tử '.'** như ví dụ sau.

```
struct studentRec john;
```

```
strcpy(john.name, "John");
```

```
john.mark = 7;
```

```
printf("%s co diem la %d", john.name, john.mark);
```



Khởi tạo các trường
cấu trúc giống như
đối với các biến
thông thường


Truy nhập cấu trúc (tiếp)

- Trong trường hợp một biến cấu trúc được trỏ bởi một con trỏ, ta có thể truy cập vào các trường cấu trúc thông qua con trỏ bằng cách dùng **toán tử '->'**.

```
struct studentRec john;  
struct studentRec *ptr = &john;
```

```
strcpy(ptr->name, "John");  
ptr->mark = 7;
```

```
printf("%s co diem la %d", ptr->name, ptr->mark);
```



Khai báo một
con trỏ có kiểu
cấu trúc

Định nghĩa kiểu (**typedef**)

- Dùng **typedef** để định nghĩa tên 1 kiểu dữ liệu mới mà có thể dùng trong các khai báo biến

```
struct studentRec  
{  
    char name[80];  
    int mark;  
};
```

Cấu trúc dữ liệu
hiện có

Tên kiểu mới

```
typedef struct studentRec Student;
```

```
Student studA, studB, *ptr;  
Student stud_list[100];
```

Khai báo biến, con
trỏ hay mảng với
kiểu dữ liệu mới

Định nghĩa kiểu (tiếp)

- Có thể gộp định nghĩa kiểu với khai báo cấu trúc như sau

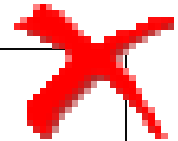
```
typedef struct studentRec
{
    char name[80];
    int mark;
} Student;
```

```
Student studA, studB, *ptr;
Student stud_list[100];
```

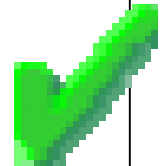
So sánh cấu trúc

- Không thể so sánh 2 cấu trúc bằng toán tử **==**
- Chỉ có thể so sánh từng trường của cấu trúc

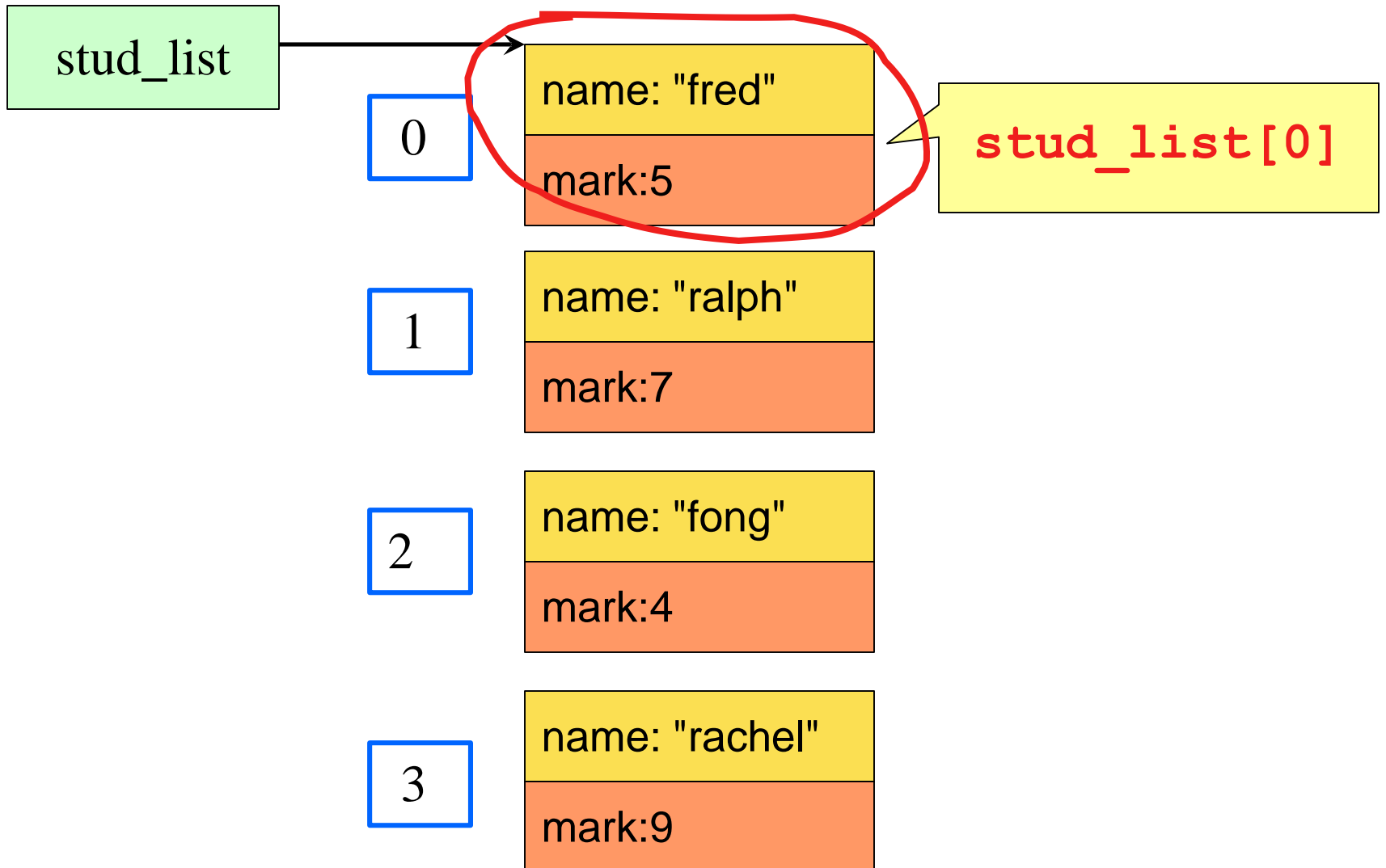
```
if (studA == studB)
{
    printf("Du lieu trung nhau.\n");
}
```



```
if (strcmp(studA.name, studB.name) == 0
    && (studA.mark == studB.mark) )
{
    printf("Du lieu trung nhau.\n");
}
```



Mảng của cấu trúc



Chương trình ví dụ (1)

```
#include <stdio.h>

#define MAXLEN  80
#define MAXN    40

typedef struct studentRec
{
    char  lastname[MAXLEN];
    int mark;
} Student;

int main()
{
    int      total, i;
    Student stud_list[MAXN];

    printf("Co bao nhieu sinh vien? ");
    scanf("%d", &total);
```

Chương trình ví dụ (2)

```
if (count > MAXN) {
    printf("So qua lon! Khong du bo nho.\n");
    exit(1);
}

printf("\nNhap danh sach ten va diem:\n");
for (i=0; i < total; i++) {
    printf("Sinh vien %d: ", i+1);
    scanf("%s %d", stud_list[i].name, &(stud_list[i].mark) );
}

printf("\nDanh sach nhung nguoi thi lai:\n\n");
for (i=0; i < total; i++)
    if (stud_list[i].mark < 5) {
        printf("Ten : %s\n", stud_list[i].name);
        printf("Diem: %d\n\n", stud_list[i].mark);
    }

return 0;
}
```

Truyền cấu trúc làm tham số

- Giống như mọi biến khác, cấu trúc có thể được dùng làm tham số của hàm
- Cũng có hai cách truyền tham số cho một cấu trúc
 - Truyền cấu trúc theo dạng sử dụng giá trị của các trường sẽ không làm thay đổi nội dung của biến cấu trúc gốc
 - Truyền địa chỉ của cấu trúc để có thể làm thay đổi nội dung của cấu trúc gốc

Hàm trả về cấu trúc

- Trả về một “gói” chứa nhiều giá trị

```
Student readRecord ( void )  
{  
    Student newStud;  
    printf("Nhap ten va diem: ");  
    scanf("%s %f", newStud.name, &(newStud.mark)) ;  
    return newStud;  
}
```

```
main()  
{  
    Student studA;  
    studA = readRecord();  
}
```

Truyền cấu trúc qua con trỏ

```
void readStudent ( Student* item )
{
    printf("Please enter name and ID\n");
    scanf("%s", s->.name);
    scanf("%f", &(s->mark) );
}
```

```
int main()
{
    Student studentA;
    readStudent(&studentA);
}
```


Chương trình số phức (1)

```
#include <stdio.h>

typedef struct complexStruct
{
    int real;
    int img;
} Complex;

Complex addComplex(Complex a, Complex b)
{
    Complex c;
    c.real = a.real + b.real;
    c.img = a.img + b.img;
    return c;
}

Complex readComplex( void )
{
    Complex c;
    printf("Nhap phan thuc ao cua so phuc: ");
    scanf("%d %d", &(c.real), &(c.img));
    return c;
}
```

Chương trình số phức (2)

```
void printComplex(Complex c)
{
    if ( c.img >= 0 )
        printf("%d + %di", c.real, c.img);
    else
        printf("%d - %di", c.real, -c.img);
}

int main()
{
    Complex a, b, tong;

    a = readComplex();
    b = readComplex();

    tong = addComplex(a, b);

    printf("Ket qua cua tong hai so phuc: ");
    printComplex(tong);

    return 0;
}
```

Bài tập

- Một phân số được biểu diễn bằng một cấu trúc gồm hai trường tử số và mẫu số
- Viết hàm cho phép nhập giá trị cho một phân số
- Viết hàm in một phân số ra màn hình
- Xây dựng các hàm để tính một phân số rút gọn, tổng, hiệu, tích, thương của hai phân số
- Viết một chương trình để thử nghiệm các hàm đã xây dựng với hai phân số được nhập vào sau đó tính tổng, hiệu, tích và thương của chúng.