Structure

Department of Information System
SoICT, HUST

Structure

- In C, a structure is known as a **struct**
- It contains a fixed number of parts, which may be of different types
- So for a friend, you may want to store name, phone number and the street they live in

Declare structure

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

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

Name of the struct

Fields of the struct

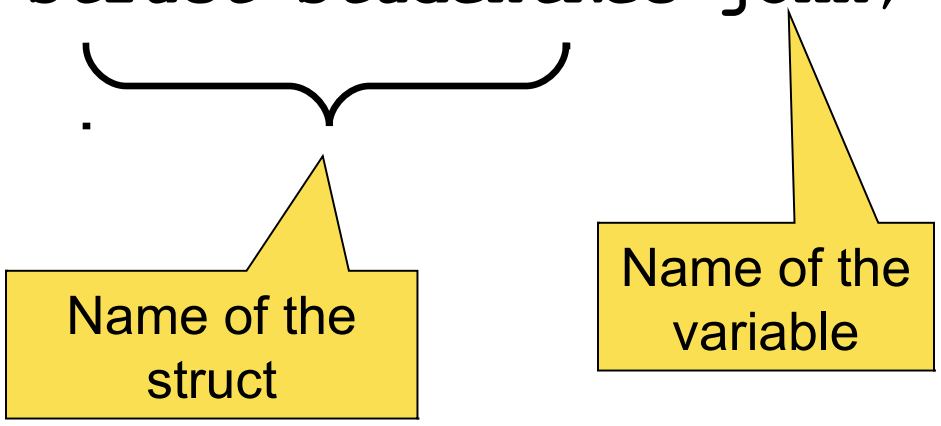
Do not forget ; after struct declaration

This declaration only create type of a struct, not struct variable

Declare a structure variable

- To declare a struct in the memory, a struct's variable should be declared as follow:

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



Declared combination

- We can declare both structure and variable, in a statement but not recommend

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

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

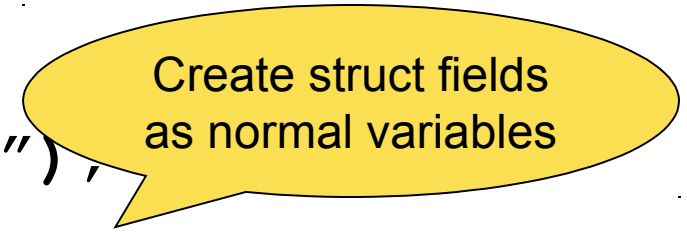
Access a structure

- To access a member of a structure, you use the '.'

```
struct studentRec john;

strcpy(john.name, "John");
john.mark = 7;

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



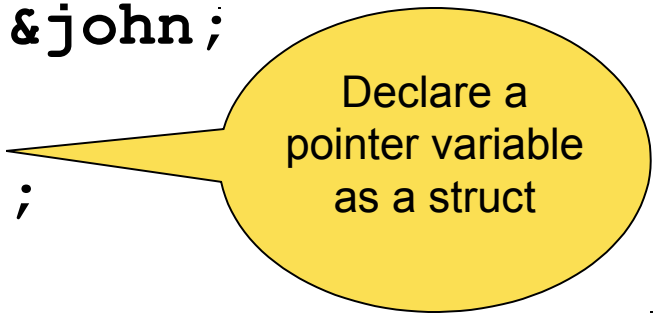
Access a structure

- Operator -> is used to access members of a structure pointed by a pointer

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

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

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



Typedef struct

- A **typedef** statement makes an identifier equivalent to a type specification

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

typedef struct studentRec Student;

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

Existing data type

New data type

Declare variables, pointers or array with new data type

Typedef struct

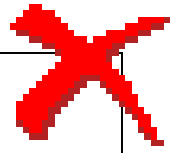
- We can declare both structure and variable, in a statement but not recommend

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

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

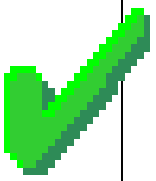
Compare structs

- Cannot compare two structs by operator ==
- Can only compare struct fields

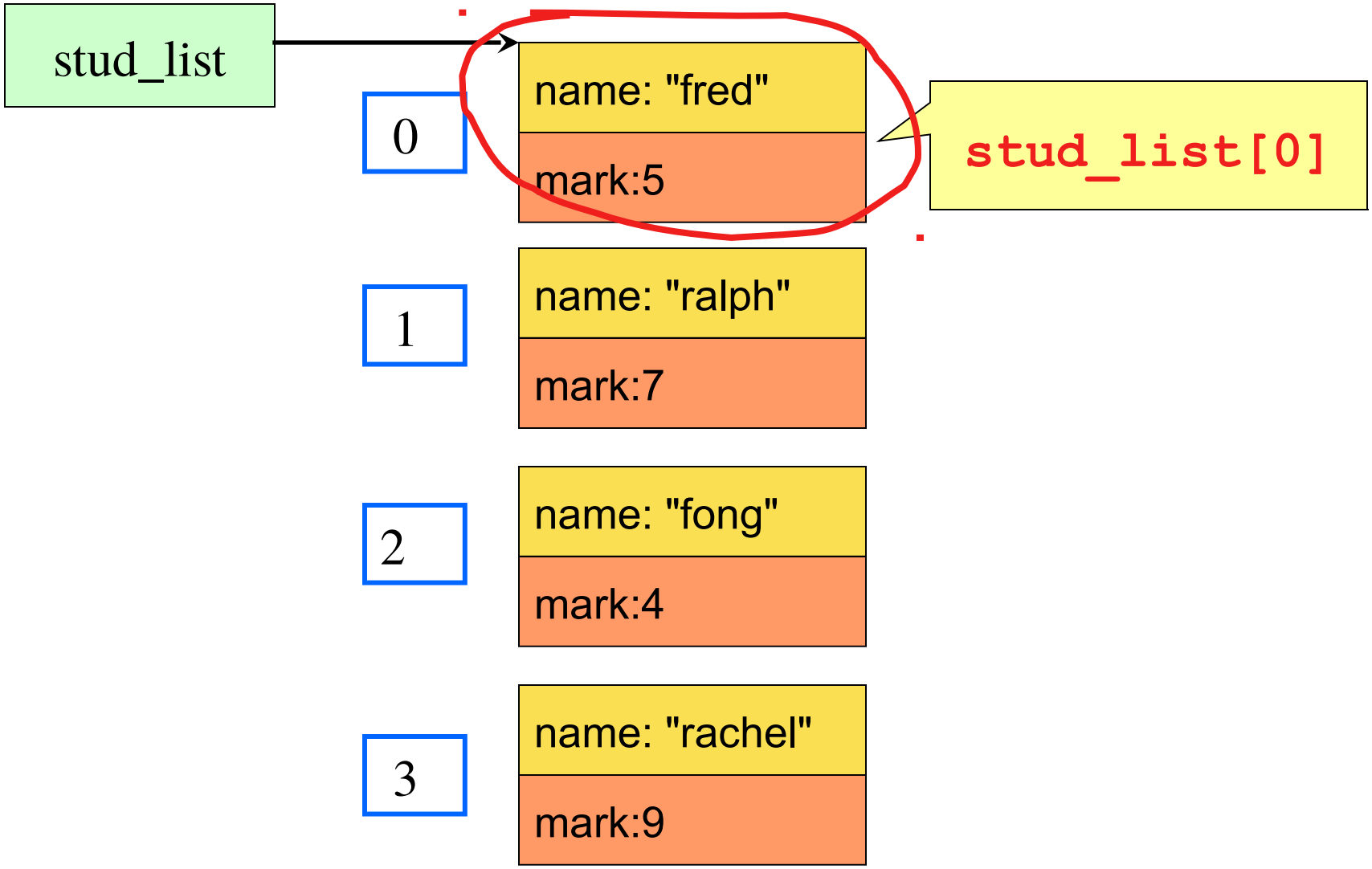


```
if (studA == studB)
{
   printf("Du lieu trung nhau.\n");
}
```

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



Array of structure



Example

```
#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("How many students? ");
  scanf("%d", &total);
```

Example

```
  if (total > MAXN) {
    printf("Number is lagre! Not enough memory.\n");
    exit(1);
  }

  printf("\nInput name and mark:\n");
  for (i=0; i < total; i++) {
    printf("Student %d: ", i+1);
    scanf("%s %d", stud_list[i].name,
&(stud_list[i].mark) );
  }

  printf("\nList of retesters:\n\n");
  for (i=0; i < total; i++)
     if (stud_list[i].mark < 5) {
        printf("Name : %s\n", stud_list[i].name);
        printf("Mark: %d\n\n", stud_list[i].mark);
     }

  return 0;
}
```

Passing a struct as a parameter

- Like any other variable, you can pass a struct as a parameter to a function
- Two ways of passing structs to functions
  - Passing structs by value doesn't change the content of the structs
  - Passing structs by reference can change the content of the structs

A function can return also a struct

- Return a “packet” that contains several values

```
Student readRecord ( void )
{
  Student newStud;
  printf(“Input Name and Mark: ");
  scanf("%s %f",newStud.name,&(newStud.mark));
  return newStud;
}
```

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

Passing struct by reference

```
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);
}
```

Exercise

Declare a structure to represent a complex number and write functions for operators add, minus, multiply,… on complex numbers

Program: complex struct

```
#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("Enter the real part and imaginary part of the complex
number: ");
  scanf("%d %d", &(c.real), &(c.img));
  return c;
}
```

Program: complex struct

```
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, sum;

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

  sum = addComplex(a, b);

  printf("Sum of two complex numbers: ");
  printComplex(sum);

  return 0;
}
```

Exercise

A fraction is represented by a struct that consists of two fields: numerator and denominator.

1. Write a function to receive value for the fraction
2. Write a function to print a fraction
3. Write functions to add, minus, multiply, divide two fractions
4. Write a program to test the above functions