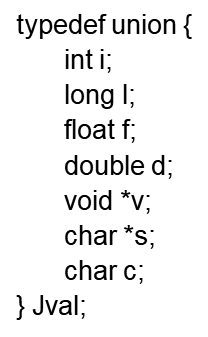
Nội dung: Kiểu dữ liệu tổng quát

* Thế nào là kiểu dữ liệu tổng quát
* So sánh với struct
* Bài tập ví dụ



Kích thước của kiểu dữ liệu = kích thước của dữ liệu lớn nhất ( dùng nhiều bộ nhớ nhất)  
VD Jval là kiểu double 8 Bytes

KHÁC với struct :

* Kích thước của struct bằng tổng kích thước của từng thành phần
* Các trường của **struct là độc lập** với nhau (bộ nhớ riêng), vì vậy giá trị các trường độc lập với nhau.
* Enum **bộ nhớ chia sẻ**, tại 1 thời điểm thì chỉ dùng như 1 kiểu dữ liệu. Nếu đọc khác kiểu → giá trị bị sai

**Jval a;**

**a.i = 10;**

**printf("a.i %d\n",a.i);**

**printf("a.l %d\n",a.l);**

**printf("a.f %d\n",a.f);**

==-====

Viết thành các hàm gán giá trị và đoc giá trị cho từng kiểu dữ liệu riêng

**Jval new\_jval\_i(int i) { Jval j; j.i = i; return j; }**

**Jval new\_jval\_l(long l) { Jval j; j.l = l; return j; }**

**Jval new\_jval\_f(float f) { Jval j; j.f = f; return j; }**

**Jval new\_jval\_d(double d) { Jval j; j.d = d; return j; }**

**Jval new\_jval\_v(void \*v) { Jval j; j.v = v; return j; }**

**int jval\_i(Jval j) { return j.i; }**

**long jval\_l(Jval j) { return j.l; }**

**float jval\_f(Jval j) { return j.f; }**

**double jval\_d(Jval j) { return j.d; }**

**void \*jval\_v(Jval j) { return j.v; }**

**int main()**

**{**

**printf("sizeof(Jval) %d\n",sizeof(Jval));**

**printf("sizeof(Sval) %d\n",sizeof(Sval));**

**Jval a;**

**//a.i = 10;**

**a = new\_jval\_i(10);**

**return 0;**

**}**

=======================

Mục đích là tạo ra giao diện tương tác dạng gọi hàm - API để cho người thiết kế kiểu dữ liệu và người dùng có thể dùng mà không cần nắm quá sâu về cách cài đặt cụ thể.

Bài tập 1. Viết lại hàm **tìm kiếm tuần tự** sử dụng kiểu dữ liệu tổng quát

// tim kiem tuan tu don gian cho so nguyen

int search(int \*list, int size, int key)

{

for(int i=0; i<size; i++)

if(list[i]==key) return i;

return -1; // khong tim thay

}

**Sửa thành hàm dùng kiểu dữ liệu tổng quát sau**

// ham kieu du lieu tong quat

int search\_gen(Jval \*list, int size, Jval key)

{

**// code here**

return -1; // khong tim thay

}

Phương án?

**Cách 1:** Tạo ra từng hàm riêng biệt cho từng kiểu dữ liệu

int search\_gen\_int(Jval \*list, int size, Jval key)

int search\_gen\_long(Jval \*list, int size, Jval key)

int search\_gen\_double(Jval \*list, int size, Jval key)

Vấn đề: có nhiều tên hàm khác nhau!

**Cách 2**: Dùng 1 tên hàm chung, thêm hàm so sánh tương ứng với từng kiểu dữ liệu (con trỏ hàm)

int search\_gen(Jval \*list, int size, Jval key, int (compare\*)(Jval \*,Jval \*))

=============

#include <stdio.h>

**typedef union {**

**int i;**

**long l;**

**float f;**

**double d;**

**void \*v;**

**char \*s;**

**char c;**

**} Jval;**

Jval new\_jval\_i(int i) { Jval j; j.i = i; return j; }

Jval new\_jval\_l(long l) { Jval j; j.l = l; return j; }

Jval new\_jval\_f(float f) { Jval j; j.f = f; return j; }

Jval new\_jval\_d(double d) { Jval j; j.d = d; return j; }

Jval new\_jval\_v(void \*v) { Jval j; j.v = v; return j; }

int jval\_i(Jval j) { return j.i; }

long jval\_l(Jval j) { return j.l; }

float jval\_f(Jval j) { return j.f; }

double jval\_d(Jval j) { return j.d; }

void \*jval\_v(Jval j) { return j.v; }

// tim kiem tuan tu don gian cho so nguyen

int search(int \*list, int size, int key)

{

for(int i=0; i<size; i++)

if(list[i]==key) return i;

return -1; // khong tim thay

}

int compare\_i(Jval\* a, Jval\* b) {

if ( jval\_i(\*a)==jval\_i(\*b) ) return 0;

if ( jval\_i(\*a) < jval\_i(\*b) ) return -1;

else return 1;

}

int compare\_f(Jval\* a, Jval\* b) {

if ( jval\_f(\*a)==jval\_f(\*b) ) return 0;

if ( jval\_f(\*a) < jval\_f(\*b) ) return -1;

else return 1;

}

// ham kieu du lieu tong quat

int search\_gen(Jval \*list, int size, Jval key, int (\*compare)(Jval\*, Jval\*) )

{

// code here

for(int i=0; i<size; i++)

if(compare(&list[i],&key)==0) return i;

return -1; // khong tim thay

}

int main()

{

Jval list\_i[]={1,2,3,2,5,6,7,8};

Jval list\_f[5];//={1.5,4.2,10.3,22.3,5.9,6.8};

for(int i=0; i<5; i++)

list\_f[i] = new\_jval\_f(i);

Jval key;

//số nguyên int

key = new\_jval\_i(2);

int ret = search\_gen(list\_i,5,key, compare\_i);

if(ret>=0) printf("Found!\n");

else printf("NOT Found!\n");

// so thuc float

for(int i=0; i<5; i++)

printf("%f ,",jval\_f(list\_f[i]));

printf("\n");

key = new\_jval\_f(4.0);

ret = search\_gen(list\_f,5,key, compare\_f);

if(ret>=0) printf("Found!\n");

else printf("NOT Found!\n");

return 0;

}

================

**Chú ý: So sánh với con trỏ**

* Con trỏ trỏ tới địa chỉ ô nhớ!
* Nếu chỉ so sánh 2 con trỏ → so sánh xem 2 con trỏ có trỏ cùng vào 1 địa chỉ ô nhớ hay không: so sánh nông - shallow compare
* Nếu so sánh giá trị tại vùng nhớ mà con trỏ đang trỏ tới → phải đọc và so sánh giá trị tại vùng nhớ đang trỏ bởi từng con trỏ với nhau: so sánh sâu - deep compare

**Viết nốt cho 3 kiểu còn lại là long, double và void\***

==============================

**Bài tập 2**. Tạo chương trình quản lý danh bạ điện thoại

Thông tin 1 bản ghi gồm:

* họ tên: char 50
* SDT: char 15

typedef struct {

char name[80];

long number;

} PhoneEntry;

typedef struct {

PhoneEntry \* entries;

int total;

int size;

} PhoneBook;

Danh bạ được lưu trữ trong mảng cấp phát động, danh bạ ban đầu là 10 phần tử, mỗi khi danh bạ đầy ta sẽ mở rộng ra mảng động mới với kích thước tăng thêm 10 mỗi lần mở rộng.

Total : tổng số bản ghi hiện tại

Size: kích thước mảng hiện tại

Size >= total

Viết các hàm

* Hàm khởi tạo danh bạ với kích thước ban đầu là 10 phần tử  
  **PhoneBook createPhoneBook();**
* Hàm thêm một bản ghi mới vào danh bạ   
  Nếu cần phải điều chỉnh thì sẽ thực hiện trong hàm này  
  **void addPhoneNumber(char \* name, long number, PhoneBook\* book);**
* Hàm tra SDT theo họ tên  
  **PhoneEntry \* getPhoneNumber(char \* name, PhoneBook book);**
* Hàm xóa danh bạ  
  **void dropPhoneBook(PhoneBook\* book);**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct {

char name[80];

long number;

} PhoneEntry;

typedef struct {

PhoneEntry \* entries; // mang ban ghi

int total;

int size;

} PhoneBook;

#define INITIAL\_SIZE 10

#define INCREMENTAL\_SIZE 5

PhoneBook createPhoneBook()

{

PhoneBook pb;

pb.total = 0;

pb.size = INITIAL\_SIZE;

pb.entries = (PhoneEntry\*)malloc(sizeof(PhoneEntry)\*pb.size);

return pb;

}

void dropPhoneBook(PhoneBook\* book)

{

free(book->entries);

book->entries = NULL;

book->total=0;

book->size=0;

}

// them 1 ban ghi moi vao book

void addPhoneNumber(char \* name, long number, PhoneBook\* book)

{

if(book->size>book->total)// chua day, co the them

{

strcpy((book->entries[book->total]).name, name);

(book->entries[book->total]).number = number;

book->total++;

}

else // cap phat dong mang moi

{

**// todo**

}

}

PhoneEntry \* getPhoneNumber(char \* name, PhoneBook book)

{

**// todo**

}

int main()

{

PhoneBook pb;

pb = createPhoneBook();

addPhoneNumber("Nguyen Van Mau",123456789,&pb);

return 0;

}