Bài tập 1. Con trỏ hàm

Sửa đổi hàm so sánh để có thể thực hiện thuật toán quicksort đối với các kiểu dữ liệu khác nhau

* Số nguyên
* Số thực kiểu float, double.
* Xâu ký tự:
  + Có phân biệt chữ hoa/thường
  + Không phân biệt

Chú ý:

* Con trỏ void : KHÔNG định kiểu nên một số thao tác  
  void \*list;
  + list[i]
  + list + i \*sizeof(list[0])

Là không hợp lệ vì KHÔNG xác định được kích thước của 1 phần tử. Vậy làm thế nào?

Dùng **char\*** và nhảy số lượng vị trí theo từng ô nhớ

* Ép kiểu void về char\*
* Luôn phải truyền vào kích thước 1 phần tử thực sự

VD hàm hoán đổi giá trị 2 phần tử

void swap(void\* a, void \*b, int **t\_size**)

{

void \*tmp = malloc(**t\_size**);

memcpy(tmp,a, **t\_size**) ;

memcpy(a,b, **t\_size**) ;

memcpy(b,tmp, **t\_size**) ;

}

a và b là 2 con trỏ trỏ đến địa chỉ đầu tiên của 2 phần tử cần hoán đổi giá trị cho nhau

t\_size là kích thước của 1 phần tử thực sự dùng để xác định số lượng ô nhớ thực sự sẽ được đọc.

Mảng các phần tử có định kiểu

int A[10]

Địa chỉ phần tử thứ 2 sẽ là **A[1]** hoặc **A + 1** (chú ý: Tên mảng là hằng con trỏ trỏ tới địa chỉ phần tử đầu tiên).

Với mảng không định kiểu ( các phần tử kiểu void)

void B[100]

void \*B

Địa chỉ phần tử đầu tiên là B

Địa chỉ phần tử thứ i ?

* KHÔNG thể dùng B[i] vì các phần tử không có kiểu → không xác định được kích thước 1 phần tử → không xác định được vị trí lệch tương đối so với phần tử đầu tiên của mảng là B
* Làm thế nào để xác định phần tử ở vị trí i?
  + Cần: Kích thước của phần tử thực sự truyền thêm vào t\_size

**(char\*)B + i \*t\_size**

Hàm quick sort

**void quicksort(void \*number,int first,int last, int (\*compare)(void const \*,void const \*), int t\_size)**

**{**

**int i, j, pivot;**

**if(first<last){**

**pivot=first;**

**i=first;**

**j=last;**

**while(i<j){**

**while(compare((char\*)number + i\*t\_size,(char\*)number +pivot\*t\_size)==-1&&i<last)**

**i++;**

**while(compare((char\*)number + j\*t\_size,(char\*)number +pivot\*t\_size)==1)**

**j--;**

**// swap phan tư**

**if(i<j){**

**swap((char\*)number + i\*t\_size,(char\*)number + j\*t\_size, t\_size);**

**}**

**}**

**swap((char\*)number + j\*t\_size,(char\*)number + pivot\*t\_size, t\_size);**

**quicksort(number,first,j-1, compare, t\_size);**

**quicksort(number,j+1,last,compare, t\_size);**

**}**

**}**

Áp dụng với kiểu số nguyên **int**

quicksort(list,0,n-1, **int\_compare**,sizeof(**int**));

Với kiểu số thực **double**

quicksort(list,0,n-1, **double\_compare**,sizeof(**double**));

CHỈ CẦN VIẾT HÀM 1 LẦN, VÀ TÙY TỪNG KIỂU DỮ LIỆU VÀO MÀ TÙY CHỈNH THAM SỐ GỌI! → KHẢ NĂNG TÁI SỬ DỤNG CODE CAO HƠN

CODE ĐỦ

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

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <string.h>

// n là số lượng phần tử cần sinh

// Hàm sinh và trả về mảng cấp phát động chứa n phần tử

int\* generateArray(int n)

{

int\* list = (int\*)malloc(n\*sizeof(int));

srand(time(NULL));

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

list[i] = rand();

return list;

}

// hàm tạo ra 1 mảng copy của mảng ban đầu (cấp phát động)

int\* copyArray(int\* source, int n)

{

int\* list = (int\*)malloc(n\*sizeof(int));

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

list[i] = source[i];

return list;

}

int int\_compare(void const\* x, void const \*y) {

int m, n;

m = \*((int\*)x); // ep ve kieu so nguyen

n = \*((int\*)y);

if ( m == n ) return 0;

return m > n? 1: -1;

}

int double\_compare(void const\* x, void const \*y) {

double m, n;

m = \*((double\*)x); // ep ve kieu so nguyen

n = \*((double\*)y);

if ( m == n ) return 0;

return m > n? 1: -1;

}

void swap(void\* a, void \*b, int size)

{

void \*tmp = malloc(size);

memcpy(tmp,a, size) ;

memcpy(a,b, size) ;

memcpy(b,tmp, size) ;

}

// mảng đầu vào number

// kích thước 1 phần tử t\_size

// hàm so sánh compare

**void quicksort(void \*number,int first,int last, int (\*compare)(void const \*,void const \*), int t\_size)**

**{**

**int i, j, pivot;**

**if(first<last){**

**pivot=first;**

**i=first;**

**j=last;**

**while(i<j){**

**while(compare((char\*)number + i\*t\_size,(char\*)number +pivot\*t\_size)==-1&&i<last)**

**i++;**

**while(compare((char\*)number + j\*t\_size,(char\*)number +pivot\*t\_size)==1)**

**j--;**

**// swap phan tư**

**if(i<j){**

**swap((char\*)number + i\*t\_size,(char\*)number + j\*t\_size, t\_size);**

**}**

**}**

**swap((char\*)number + j\*t\_size,(char\*)number + pivot\*t\_size, t\_size);**

**quicksort(number,first,j-1, compare, t\_size);**

**quicksort(number,j+1,last,compare, t\_size);**

**}**

**}**

void printArr(int \*list, int n)

{

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

printf("%d, ",list[i]);

printf("\n");

}

int main()

{

const int n=10;

double time\_spent = 0.0;

printf("So luong phan tu n=%d\n",n);

clock\_t begin = clock();

int \*source = generateArray(n);

clock\_t end = clock();

// calculate elapsed time by finding difference (end - begin) and

// divide by CLOCKS\_PER\_SEC to convert to seconds

time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("Sinh mang mat thoi gian la %f micro seconds\n", time\_spent\*1e6);

int \*copy1, \*copy2, \*copy3;

begin = clock();

copy1 = copyArray(source, n);

end = clock();

time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("Copy mang mat thoi gian la %f micro seconds\n", time\_spent\*1e6);

copy3 = copyArray(source, n);

printArr(copy3,n);

begin = clock();

quicksort(copy3,0,n-1, int\_compare,sizeof(int));

end = clock();

time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;

printf("Sap xep QuickSort mat thoi gian la %f micro seconds\n", time\_spent\*1e6);

printArr(copy3,n);

// testing

printf("\n");

printf("%d %d\n",copy3[0],copy3[1]);

swap(copy3, copy3 + 1, sizeof(copy3[0]));

printf("%d %d\n",copy3[0],copy3[1]);

// giai phong bo nho dong

free(copy3);

free(source);

return 0;

}

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

**Bài tập 1**.

Sửa lại hàm sắp xếp nổi bọt và sắp xếp chèn để có thể sắp xếp trên nhiều kiểu dữ liệu khác nhau

**//hàm sắp xếp nổi bọt**

**void bubblesort(int \*list, int n)**

**{**

**for(int i=n; i>1; i--)**

**for(int j=1;j<i; j++)**

**{**

**if(list[j-1]> list[j])**

**{**

**int tmp = list[j];**

**list[j] = list[j-1];**

**list[j-1] = tmp;**

**}**

**}**

**}**

**// hàm sắp xếp lựa chọn**

**void selectionsort(int \*list, int n)**

**{**

**for(int i=n; i>1; i--)**

**{**

**int vtmax = 0;**

**for(int j=1; j<i; j++)**

**if(list[j]>list[vtmax]) vtmax = j;**

**int tmp = list[i-1];**

**list[i-1] = list[vtmax];**

**list[vtmax] = tmp;**

**}**

**}**

**Bài tập 2**. Xây dựng hàm so sánh xâu ký tự có và KHÔNG phân biệt chữ HOA/thường để sắp xếp 1 danh sách họ tên được nhập từ bàn phím

// có phân biệt HOA/thường

int string\_compare\_case(void const\* x, void const \*y) {

char \*m, \*n;

// so sánh tại đây

}

// không phân biệt

int string\_compare\_nocase(void const\* x, void const \*y) {

char \*m, \*n;

// so sánh tại đây

}

Cách làm

* So sánh từng ký tự của 2 xâu cho tới hết
* Nếu ký tự thứ i của 2 xâu bằng nhau → so sánh tiếp i+1
* Xâu có các ký tự giống nhau nhưng ngắn hơn là xâu nhỏ hơn. VD ABC và ABCDE

Dùng vòng lặp while

int **string\_compare\_case**(void const \*str1, void const \*str2)

{

char \*s1, \*s2;

s1 = (char\*)str1; // ep ve kieu xau

s2 = (char\*)str2;

return strcmp(s1,s2);

}

int **string\_compare\_nocase**(void const \*str1, void const \*str2)

{

char \*s1, \*s2;

s1 = (char\*)str1; // ep ve kieu xau

s2 = (char\*)str2;

for (;; s1++, s2++) {

int d = tolower((unsigned char)\*s1) - tolower((unsigned char)\*s2);

if (d != 0 || !\*s1)

return d>0?1:(d<0?-1:0);

}

}

Áp dụng 2 hàm trên vào quicksort để sắp xếp mảng tên sau theo 2 cách

**char namelist[][30]={"iPhone","Apple","apple","Samsung","Google","Facebook","Xiaomi","Huawei"};**