TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN CÔNG NGHỆ PHẦN MỀM
MÔN: **KĨ THUẬT LẬP TRÌNH**

HƯỚNG DẪN THỰC HÀNH

TUẦN 9

KĨ THUẬT TÙY BIẾN MÃ NGUỒN

TP.HCM, ngày 20 tháng 02 năm 2023

MỤC LỤC

1 Giới thiệu

Khi lập trình, ta bắt gặp một vài tình huống ở đó mã nguồn khá tương tự nhau, nhưng do vấn đề về kiểu dữ liệu hoặc điều kiện có sự thay đổi, ta buộc lòng phải viết những đoạn mã lặp lại với cùng tư tưởng. Như vậy, ta vô tình là tăng kích thước chương trình mà lẻ ra ta không cần làm như vậy. Tuần này ta sẽ tìm hiểu kĩ thuật về con trỏ hàm để khắc phục phần nào hạn chế trên.

2 Hàm swap

Hàm này là hàm có lẻ ta sử dụng rất nhiều khi có nhu cầu chuyển đổi nội dung giữa hai phần tử.

Dòng	
1	void swap(void * a, void* b, int size){
2	void* temp = malloc(size);
3	memcpy(temp, b, size);
4	memcpy(b, a, size);
5	memcpy(a, temp, size);
6	free(temp);
7	}
8	
9	void main(){
10	int a = 5, b = 6;
11	swap(&a, &b, sizeof(int));
12	cout << a << b << endl;
13	}

Trong đó tham số cuối cùng rất quan trọng, đây là dữ kiện để ta có thể dùng làm ranh giới trong khi hoán chuyển nội dung.

3 Hàm tìm phần tử theo yêu cầu trong một mảng số nguyên

Chúng ta có thể thấy việc tìm kiếm theo yêu cầu nào đó là hết sức phổ biến. Ví dụ tìm phần tử nhỏ nhất trong mảng một chiều hoặc tìm phần tử lớn nhất trong mảng một chiều. Hai thuật toán tìm kiếm cho hai vấn đề này gần như là giống nhau, chỉ khác ở điều kiện tìm kiếm. Vì vậy ta có thể tách phần điều kiện ra làm một hàm tiêu chuẩn riêng, để khi nào tìm theo điều kiện tiêu chuẩn nào thì ta xây dựng riêng tiêu chuẩn đó cho phù hợp.

Dòng	
1	int FindMinMax(int a[], int n, bool (*Con)(int, int)){

```
2    int index = -1;
3    for(int i = 1; i < n ; i++)
4      if(Con(a[i], a[index]))
5        index = i;
6    return index;
7  }
8
9  bool MaxCon(int a, int b){ return a > b;}
10 bool MinCon(int a, int b){return !MaxCon(a, b);}
11
12 void main(){
13   int a[] = {5, 3, 9, 1, 6};
14   cout << FindMinMax(a, 5, &MinCon) << endl;
15   cout << FindMinMax(a, 5, &MaxCon) << endl;
16 }
```

4 Hàm tìm phần tử theo yêu cầu trong mảng có kiểu bất kì

Trong ví dụ trên ta thấy tuy điều kiện tìm kiếm có thể linh hoạt, tuy nhiên kiểu dữ liệu thì vẫn chưa thể tùy biến (cụ thể chỉ xử lí trên mảng số nguyên). Ta có thể nghĩ ngay đến kĩ thuật void* để tổng quát quá kiểu dữ liệu, tuy nhiên điều này là không thể do void* là con trỏ vô kiểu, và khi chưa xác định kiểu thì toán tử lấy giá trị tại địa chỉ mà con trỏ này trỏ tới (toán tử *) là không sử dụng được. Vì vậy ta sẽ dùng kĩ thuật template trong C++ để giúp tổng quát hóa đoạn mã trên.

```
Dòng
0  template <class T>
1  int FindMinMax(T a[], int n, bool (*Con)(T, T)){
2    int index = -1;
3    for(int i = 1; i < n ; i++)
4      if(Con(a[i], a[index]))
5        index = i;
6    return index;
7  }
8
9  template <class T>
10 bool MaxCon(T a, T b){ return a > b;}
11
12 template <class T>
13 bool MinCon(T a, T b){return !MaxCon(a, b);}
14
```

```
15  void main(){
16      int a[] = {5, 3, 9, 1, 6};
17      cout << FindMinMax(a, 5, &MinCon) << endl;
18      float b[] = {5.1F, 3.6F, 1.9F, 1.0F, 4.3F};
19      cout << FindMinMax(b, 5, &MaxCon) << endl;
20  }
```

5 Bài tập

5.1 Xây dựng trên mảng một chiều

Sinh viên xây dựng **MỘT** hàm sắp xếp tổng quát sao cho hàm có khả năng sắp xếp các mảng có kiểu dữ liệu: int, float, long, double, PHANSO, string theo thứ tự **tăng**, **giảm**, và **bất kì**.

Lưu ý: thứ tự bất kì nhưng về cơ bản vẫn là so sánh 2 phần tử với nhau, sau đó trả ra true/false (Khai báo con trỏ hàm có dạng: bool (*Compare)(T a, T b))

Ví dụ: int a[] = {2,5,6,4,1}

float a[] = {2.0F, 5.7F, 6.7F, 4.1F, 1.9F}

PHANSO a[] = {{1,2}, {2,3}, {3,7}}

string s[] = {“hello”, “chao”, “bonjour”}

5.2 Xây dựng trên danh sách liên kết đơn

Yêu cầu **tương tự như 5.1**, tuy nhiên ta dùng danh sách liên kết đơn để chứa dữ liệu

Sinh viên tự thiết kế hàm main minh họa **các yêu cầu** bên trên theo hai loại cấu trúc mảng và danh sách liên kết đơn **MỘT CÁCH RÕ RÀNG**.