Bài thực hành số 03. DANH SÁCH TỔ CHỨC BẰNG MẢNG

Mục tiêu: thực hành các thao tác trên một danh sách cụ thể sử dụng kiểu dữ liệu mảng. Làm quen với lớp vector để thao tác với danh sách.

Nội dung thực hành

Bài 1. Danh sách khách hàng

Cần quản lý một danh sách các khách hàng sử dụng dịch vụ điện thoại, trong đó mỗi khách hàng gồm các thông tin: tên khách hàng, địa chỉ, các số điện thoại của khách hàng đăng ký sử dụng (giả sử tên khách hàng không trùng nhau).

Cho trước cách tổ chức dữ liệu kiểu mảng để lưu danh bạ với các thao tác cơ bản đã cài đặt:

Tổ chức dữ liệu:

```
#include <iostream>
#include <vector>
//Maximum of customers
#define max 100
// Data type for customer
struct Customer
{
  string        name;
  string        address;
  vector<string>phoneNumbers;
};
//Data type for list of customer by array
struct ListOfCustomers
{
  Customer arrCustomers[max];
  int      numOfCustomers;
};
```

```
//Input a customer
void inputCustomer(Customer &c)
{
  int i, n; string phone;
```

```
  cin.ignore();
  cout << "Name: "; getline(cin, c.name);
  cout << "Address: "; getline(cin, c.address);
  cout << "Num of phones: "; cin >> n;
  cin.ignore();
  for (i = 0; i < n; i++)
  {
       cout << "Phone number " << i << " : "; getline(cin, phone);
       c.phoneNumbers.push_back(phone);
  }
}
```

```
//Output a customer
void outputCustomer(Customer c)
{
  int i;
  cout << "Name: " << c.name << endl;
  cout << "Address: " <<  c.address << endl;
  cout << "List of phones: " ;
  for (i = 0; i < c.phoneNumbers.size(); i++)
       cout << c.phoneNumbers[i] << " ";
    cout << endl;
}
```

```
//Input a list of customers
void inputListOfCustomers(ListOfCustomers& list)
{
  cout << "Number of customers: "; cin >> list.numOfCustomers;
  for(int i = 0; i < list.numOfCustomers; i++)
    inputCustomer(list.arrCustomers[i]);
}
```

```
//Output a list of customers
void outputListOfCustomers(ListOfCustomers list)
```

```
{
  cout << "Number of customers: " << list.numOfCustomers << endl;
  for(int i = 0; i < list.numOfCustomers; i++)
    outputCustomer(list.arrCustomers[i]);
}
```

```
//Inserst a customer into the list at the specified index
void insertCustomer(ListOfCustomers& list, Customer customer, int index)
{
    if (index < 0 || index > list.numOfCustomers) {
        cout << "Invalid index. Please enter a valid index between 0 and
" << list.numOfCustomers << endl;
        return;
    }
    if (list.numOfCustomers == max) {
        cout << "Cannot insert customer. List is full." << endl;
        return;
    }
    for (int i = list.numOfCustomers; i > index; i--)
    list.arrCustomers[i] = list.arrCustomers[i - 1];
    list.arrCustomers[index] = customer;
    list.numOfCustomers++;
    cout << "Successfully inserted customer at index " << index << endl;
}
```

```
//Delete a customer in the list at the specified index
void deleteCustomer(ListOfCustomers& list, int index) {
    if (index < 0 || index >= list.numOfCustomers) {
        cout << "Invalid index. Please enter a valid index between 0 and
" << list.numOfCustomers - 1 << endl;
        return;
    }
    for (int i = index; i < list.numOfCustomers - 1; i++) {
        list.arrCustomers[i] = list.arrCustomers[i + 1];
    }
```

```
    list.numOfCustomers--;
    cout << "Successfully deleted customer at index " << index << endl;
}
```

```
//Search a customer in the list by name
int findCustomerByName(ListOfCustomers& list, string name) {
    for (int i = 0; i < list.numOfCustomers; i++) {
        if (list.arrCustomers[i].name == name) {
            return i;
        }
    }
    cout << "No customer found with name: " << name << endl;
    return -1;
}
```

Hàm main() thực hiện:

+ Nhập một danh sách khách hàng

+ In danh sách vừa nhập lên màn hình

+ Nhập một khách hàng và vị trí cần chèn khách hàng vào danh sách. Thực hiện chèn khách hàng vào danh sách tại vị trí cần chèn.

+ In danh sách sau khi chèn

+ Nhập tên khách hàng cần tìm. Thực hiện tìm và hiển thị kết quả.

+ Nhập vị trí khách hàng cần xóa. Thực hiện xóa khách hàng tại vị trí cần xóa.

```
int main()
{
  Customer c; ListOfCustomers l; string name; int k;
  //Input a list of customers
  inputListOfCustomers(l);
  //Output list of customers
  outputListOfCustomers(l);
  //Insert a customer
  inputCustomer(c);
  cout << "Index for insert: " ; cin >> k;
  insertCustomer(l, c, k);
  //Output list of customers after insert
```

```
  outputListOfCustomers(l);
  //Find a customer
  cout << "Name of customer: "; getline(cin, name);
  k = findCustomerByName(l, name);
  if (k==-1)
    cout << "Not found the customer name: " << name;
  else
  {
    cout << "Found a customer: ";
    outputCustomer(l.arrCustomers[k]);
  }
  //Delete a customer
  cout << "Index for delete: " ; cin >> k;
  deleteCustomer(l, k);
  //Output list of customers after delete
  outputListOfCustomers(l);return 0;
}
```

Hãy thực hành các thao tác sau trên danh sách khách hàng:

a) Viết hàm xóa một khách hàng theo tên. Sử dụng hàm này xóa một khách hàng tên Nam.

```
void deleteCustomerByName(ListOfCustomers &list, string name)
{



}
```

Bổ sung các lệnh trong hàm main()

```

```

b) Viết hàm tìm khách hàng theo số điện thoại. Sử dụng tìm khách hàng có số điện thoại "0912345678".

Hướng dẫn: duyệt từng khách hàng trong danh sách, với mỗi khách hàng duyệt các số điện thoại của khách hàng để tìm số điện thoại cần tìm.

Hàm trả về vị trí khách hàng có số điện thoại cần tìm trong danh sách hoặc -1 nếu không có số điện thoại trong danh sách có dạng:

```
int findCustomerByPhone(ListOfCustomers list, string phone)
{



}
```

Câu lệnh trong hàm main()

```

```

c) Viết hàm xóa một số điện thoại cụ thể của khách hàng trong danh sách. Sử dụng xóa một số điện thoại nhập vào từ bàn phím. Sau đó in danh sách khách hàng để kiểm tra.

Hướng dẫn: sử dụng hàm `findCustomerByPhone()` để tìm khách hàng có số điện thoại cần xóa. Nếu tìm được thì xóa số điện thoại trong danh sách các số điện thoại của khách hàng này.

```
void deletePhone(ListOfCustomers &list, string phone)
{



}
```

Câu lệnh trong hàm main()

```

```

d) Viết hàm thêm một số điện thoại cho một khách hàng có tên cho trước. Sử dụng hàm này thêm số điện thoại "0989123456" cho khách hàng tên "An".

```
void appendPhone(ListOfCustomers &list, string name, string phone)
{



}
```

Câu lệnh trong hàm main()

```

```

e) *Viết hàm thêm một khách hàng vào danh sách khách hàng. Nếu khách hàng chưa có trong danh sách thì thêm vào cuối danh sách; nếu đã có thì cập nhật các số điện thoại của khách hàng nếu chưa có trong danh sách.

Hướng dẫn: tìm khách hàng đã có trong danh sách chưa theo tên. Nếu chưa có thì thêm khách hàng vào cuối danh sách. Nếu có rồi thì lấy từng số điện thoại của khách hàng cần thêm thêm vào các số điện thoại của khách hàng đã có trong danh sách nếu số điện thoại đấy chưa có.

```
void mergeCustomer(ListOfCustomers &list, Customer c)
{




}
```

Câu lệnh trong hàm main()

```

```

Câu hỏi: *hãy dùng vector để quản lý danh sách khách hàng thay cho dùng mảng như trên. Hãy sửa lại các hàm cho phù hợp với cách tổ chức này. Cách tổ chức này có gì tốt hơn cách dùng mảng?*
