ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
THI KẾT THÚC HỌC PHẦN
Học kỳ 1 - Năm học 2024 - 2025
Lớp: 24CTT3A

Tên HP:	Thực hành Cơ sở lập trình	Mã HP: CSC10012
Thời gian:	100 phút	Ngày thi: 18/12/2024

Hướng dẫn: Thí sinh được phép sử dụng bất kỳ tài liệu hay vật dụng nào.

Yêu cầu chung

Sử dụng ngôn ngữ C/C++ để viết chương trình quản lý cửa hàng điện thoại. Chương trình phải được chia thành các file: `Main.cpp`, `Header.h`, và `Source.cpp`. Bắt buộc sử dụng danh sách liên kết đơn để lưu trữ dữ liệu điện thoại (nếu không sử dụng sẽ hủy toàn bộ bài thi). Xây dựng menu để người dùng có thể lặp đi lặp lại các thao tác.

Chi tiết yêu cầu

1. Cấu trúc dữ liệu (0.5 điểm)

Mỗi chiếc điện thoại (`Phone`) có 4 thông tin:

- `id`: Mã số (chuỗi ký tự, có thể chứa dấu cách).
- `brand`: Hãng (chuỗi ký tự, có thể chứa dấu cách).
- `name`: Tên điện thoại (chuỗi ký tự, có thể chứa dấu cách).
- `price`: Giá tiền (số thực lớn hơn 0).

Mỗi `Node` có 2 thông tin:

- `data`: dữ liệu một chiếc điện thoại.
- `next`: Địa chỉ `Node` kế tiếp.

2. Menu chức năng (0.5 điểm)

Xây dựng menu cho phép người dùng thực hiện các chức năng sau:

1. Nhập thông tin điện thoại từ bàn phím.
2. In danh sách điện thoại ra màn hình.
3. Tìm và cập nhật hãng của điện thoại theo mã số rồi in ra màn hình danh sách sau khi thay đổi.
4. Xóa tất cả các điện thoại có giá lớn hơn một giá trị `k` nhập từ bàn phím rồi in ra màn hình danh sách còn lại.
5. Ghi danh sách điện thoại vào file `dienthoai.bin`.

6. Đọc danh sách điện thoại từ file `dienthoai.bin`, sắp xếp theo mã số tăng dần và in ra màn hình.
7. Thoát chương trình.

3. Nhập và xuất danh sách điện thoại (3 điểm)

- Viết hàm để nhập thông tin điện thoại từ bàn phím.
- Viết hàm để in danh sách điện thoại ra màn hình (đủ thông tin: mã số, hãng, tên, giá).

4. Cập nhật hãng theo mã số (2 điểm)

- Cho người dùng nhập mã số. Nếu mã số tồn tại, cho phép người dùng cập nhật lại hãng của điện thoại đó.
- Nếu mã số không tồn tại, thông báo lỗi.

5. Xóa điện thoại theo giá (2 điểm)

- Cho người dùng nhập một giá trị `k` (số thực).
- Xóa tất cả các điện thoại có giá lớn hơn `k` khỏi danh sách.

6. Ghi và đọc file binary (2 điểm)

- Viết hàm ghi danh sách điện thoại hiện có vào file `dienthoai.bin`.
- Viết hàm đọc danh sách điện thoại từ file `dienthoai.bin`, sắp xếp theo mã số tăng dần và in ra màn hình.

Ví dụ minh họa

Input:

Người dùng nhập các điện thoại sau:

```
001
Samsung
Galaxy S21
1200.0
002
Apple
iPhone 13
1500.0
003
Xiaomi
Mi 11
800.0
```

Output:

1. Danh sách điện thoại:

```
Ma so: 001, Hang: Samsung, Ten: Galaxy S21, Gia: 1200.0
Ma so: 002, Hang: Apple, Ten: iPhone 13, Gia: 1500.0
Ma so: 003, Hang: Xiaomi, Ten: Mi 11, Gia: 800.0
```

2. Cập nhật hãng của điện thoại có mã số "002" thành "Apple Inc.":

```
Dien thoai co ma so 002 da duoc cap nhat hang thanh Apple Inc.
Danh sach sau khi thay doi:
Ma so: 001, Hang: Samsung, Ten: Galaxy S21, Gia: 1200.0
Ma so: 002, Hang: Apple Inc., Ten: iPhone 13, Gia: 1500.0
Ma so: 003, Hang: Xiaomi, Ten: Mi 11, Gia: 800.0
```

3. Xóa các điện thoại có giá lớn hơn 1000.0:

```
Da xoa cac dien thoai co gia lon hon 1000.0.
Danh sach con lai:
Ma so: 003, Hang: Xiaomi, Ten: Mi 11, Gia: 800.0
```

4. Sau khi Ghi vào file dienthoai.bin thì Đọc từ file dienthoai.bin rồi sắp xếp và in ra màn hình:

```
Ma so: 001, Hang: Samsung, Ten: Galaxy S21, Gia: 1200.0
Ma so: 002, Hang: Apple, Ten: iPhone 13, Gia: 1500.0
Ma so: 003, Hang: Xiaomi, Ten: Mi 11, Gia: 800.0
```

Lời giải tham khảo

Header.h

```
#ifndef HEADER_H
#define HEADER_H

#include <iostream>
#include <string>
#include <fstream>

using namespace std;

struct phone {
    string id, brand, name;
    float price;
};

struct Node {
```

```
    phone data;
    Node* next;
};

void inputPhone(Node*& head);
void outputPhone(Node* head);
void updateBrand(Node* head, string id, string brand);
void removePriceMoreThanK(Node*& head, float k);
void writeToBinaryFile(Node* head, string fileName);
void readFromBinaryFileAndSortById(Node*& head, string fileName);

#endif
```

Source.cpp

```
#include "Header.h"

void inputPhone(Node*& head) {
    int n;
    cout << "Nhap so luong: ";
    cin >> n;
    Node* tail = nullptr;
    for (int i = 0; i < n; i++) {
        Node* newNode = new Node();
        cout << "Nhap ma cua dien thoai thu " << i + 1 << ": ";
        cin.ignore();
        getline(cin, newNode->data.id);
        cout << "Nhap thuong hieu cua dien thoai thu " << i + 1 << ": ";
        getline(cin, newNode->data.brand);
        cout << "Nhap ten cua dien thoai thu " << i + 1 << ": ";
        getline(cin, newNode->data.name);
        cout << "Nhap gia cua dien thoai thu " << i + 1 << ": ";
        cin >> newNode->data.price;
        newNode->next = nullptr;
        if (head == nullptr) {
            head = newNode;
            tail = newNode;
        }
        else {
            tail->next = newNode;
            tail = newNode;
        }
    }
}

void outputPhone(Node* head) {
    Node* current = head;
    while (current != nullptr) {
        cout << "Ma so: " << current->data.id << ", Hang: " << current->data.brand <<
            ", Ten: " << current->data.name << ", Gia: " << current->data.price <<
            endl;
        current = current->next;
    }
}
```

```
void updateBrand(Node* head, string id, string brand) {
    Node* current = head;
    while (current != nullptr) {
        if (current->data.id == id) {
            current->data.brand = brand;
        }
        current = current->next;
    }
    cout << "Danh sach sau khi thay doi: " << endl;
    outputPhone(head);
}

void removePriceMoreThanK(Node*& head, float k) {
    Node* current = head;
    Node* prev = nullptr;
    while (current != nullptr) {
        if (current->data.price > k) {
            if (prev == nullptr) {
                head = current->next;
            }
            else {
                prev->next = current->next;
            }
            Node* temp = current;
            current = current->next;
            delete temp;
        }
        else {
            prev = current;
            current = current->next;
        }
    }
    cout << "Danh sach sau khi xoa: " << endl;
    outputPhone(head);
}

void writeToBinaryFile(Node* head, string fileName) {
    ofstream ofs(fileName, ios::binary);
    if (!ofs) {
        cerr << "Khong the mo file de ghi!" << endl;
        return;
    }

    Node* current = head;
    while (current != nullptr) {
        const phone& p = current->data;

        // Ghi id
        size_t idLength = p.id.size();
        ofs.write((char*)&idLength, sizeof(idLength));
        ofs.write(p.id.c_str(), idLength);
```

```
        // Ghi brand
        size_t brandLength = p.brand.size();
        ofs.write((char*)&brandLength, sizeof(brandLength));
        ofs.write(p.brand.c_str(), brandLength);

        // Ghi name
        size_t nameLength = p.name.size();
        ofs.write((char*)&nameLength, sizeof(nameLength));
        ofs.write(p.name.c_str(), nameLength);

        // Ghi price
        ofs.write((char*)&p.price, sizeof(p.price));

        current = current->next;
    }
    ofs.close();

    cout << "Ghi thanh cong" << endl;
}

void readFromBinaryFileAndSortById(Node*& head, string fileName) {
    ifstream ifs(fileName, ios::binary);
    if (!ifs) {
        cerr << "Khong the mo file de ghi!" << endl;
        return;
    }

    // Xoa du lieu cu
    while (head != nullptr) {
        Node* temp = head;
        head = head->next;
        delete temp;
    }

    Node* tail = nullptr;
    while (ifs) {
        phone p;

        // Doc id
        size_t idLength;
        ifs.read((char*)&idLength, sizeof(idLength));
        if (!ifs) break; // Kiem tra xem co du lieu de doc khong
        p.id.resize(idLength);
        ifs.read(&p.id[0], idLength);

        // Doc brand
        size_t brandLength;
        ifs.read((char*)&brandLength, sizeof(brandLength));
        p.brand.resize(brandLength);
        ifs.read(&p.brand[0], brandLength);

        // Doc name
        size_t nameLength;
```

```
        ifs.read((char*)&nameLength, sizeof(nameLength));
        p.name.resize(nameLength);
        ifs.read(&p.name[0], nameLength);

        // Doc price
        ifs.read((char*)&p.price, sizeof(p.price));

        Node* newNode = new Node();
        newNode->data = p;
        newNode->next = nullptr;
        if (head == nullptr) {
            head = newNode;
            tail = newNode;
        }
        else {
            tail->next = newNode;
            tail = newNode;
        }
    }

    ifs.close();

    // Sap xep danh sach lien ket theo id
    if (head == nullptr) return;
    for (Node* i = head; i->next != nullptr; i = i->next) {
        for (Node* j = i->next; j != nullptr; j = j->next) {
            if (i->data.id > j->data.id) {
                swap(i->data, j->data);
            }
        }
    }

    cout << "Danh sach sau khi sap xep: " << endl;
    outputPhone(head);
}
```

Main.cpp

```
#include "Header.h"

int main() {
    Node* phones = nullptr;
    int n;
    cout << "Menu: " << endl;
    cout << "1. Input phone" << endl;
    cout << "2. Output phone" << endl;
    cout << "3. Update brand" << endl;
    cout << "4. Remove phone with price more than k" << endl;
    cout << "5. Write to binary file" << endl;
    cout << "6. Read from binary file and sort by id" << endl;
    cout << "7. Exit" << endl;

    string id, brand;
    float k;
```

```
    do {
        cout << endl;
        cout << "Nhap lua chon: ";
        cin >> n;
        switch (n) {
        case 1:
            inputPhone(phones);
            break;
        case 2:
            outputPhone(phones);
            break;
        case 3:
            cout << "Nhap id can thay doi: ";
            cin.ignore();
            getline(cin, id);
            cout << "Nhap thuong hieu moi: ";
            getline(cin, brand);
            updateBrand(phones, id, brand);
            break;
        case 4:
            cout << "Nhap gia k: ";
            cin >> k;
            removePriceMoreThanK(phones, k);
            break;
        case 5:
            writeToBinaryFile(phones, "dienthoai.bin");
            break;
        case 6:
            readFromBinaryFileAndSortById(phones, "dienthoai.bin");
            break;
        case 7:
            exit(0);
            break;
        default:
            cout << "Lua chon khong thich hop!" << endl;
            break;
        }
    } while (n != 7);

    // Clean up the linked list
    while (phones != nullptr) {
        Node* temp = phones;
        phones = phones->next;
        delete temp;
    }

    return 0;
}
```

HẾT

(Bài làm gồm 7 trang.)