**РУСЕНСКИ УНИВЕРСИТЕТ „АНГЕЛ КЪНЧЕВ“**

Катедра: Компютърни системи и технологии

**КУРСОВА ЗАДАЧА**

**по дисциплината**

**„Синтез и анализ на алгоритми“**

Изготвил: Даниел Ивелинов Панайотов

фак. номер: 233001

група: 25А

курс: 2

специалност: КСТ

Дата:................... Приел:.....................................

/доц. д-р Е. Големанова/

1. **Задание**

**No. 1: Да се разработи програма, която създава и поддържа телефонен и адресен**

**указател. За всяко лице указателят да съдържа следната информация:**

1. **трите имена**
2. **домашни и служебни адреси**
3. **телефони на съответните адреси**

**Указателят нека има следната нелинейна структура:**

A group of black text

Description automatically generated

**Програмата трябва да изпълнява следните функции:**

1. **добавяне на информация за ново лице;**
2. **извеждане на целия указател на екрана в следния вид:**

A close up of a phone number

Description automatically generated

1. **извежда информация за броя на работещите (лицето е работещо, ако има поне един служебен адрес).**
2. **Графично изображение на динамичния списък с конкретни примерни стойности**

A diagram of a computer flow

Description automatically generated

1. **Описание на алгоритмите, съответстващи на отделните подточки от заданието:**
2. **Съдържане на информация за лицето: три имена, домашни и служебни адреси, телефони за съответните адреси**

Има три структури които играят ролята на Node за всеки от трите едносвързани списъка: Telephone, Address, Contact. В тях се съдържа нужната информация, като адрес и тип на адреса, имена, телефонен номер, както и връзка към следващият елемент в списъка и връзка към първият елемент от починениея списък. За Contact е Address, за Address е Telephone. Може всеки контакт да има множество адреси, както и всеки адрес може да има множество телефони, но те пазят само първият елемент.

1. **Добавяне на информация за ново лице**

Има функция наречена „insertContact“, която запитва потребителя за имена, създава Node и го добавя в края на опашката, ако няма начало на опашката, се добавя като начален елемент. След това ще запита, дали потребителят иска да добави адрес, в който случай извиква друга функция „insertAddress“. Тя също запитва за адрес, какъв тип адрес и дали потребителят иска да добави телефонен номер, ако се съгласи, тогава програмата извиква трета функция „insertPhone“ и след изпълнението и запитва потребителя, дали иска да добави друг адрес. Ако се съгласи се повтарят горе-описаните операции. Функцията “insertPhone” запитва потребителят за телефон, добавя го в списъка и пита, дали потребителят желае да добави друг телефон, в който случай повтаря същите операции.

1. **Извеждане на указателя в конзолата**

Функцията се казва „listContacts“ три цикъла, два от които са вложени и за всяка тяхна итерация построяват един общ string, който след завършването на главния цикъл се извежда в конзолата и започва да се строи нов, докато има контакти.

1. **Извеждане на броя на работещите лица**

Функцията „listWorkingContacts“ изпълнява това действие, тя има един главен и втори вложен цикъл. Главният взима всеки контакт, за да може вложеният да проверява неговите адрес, дали един от тях е отбелязан за частен. Ако е частен увеличава брояч и продължава към следващия контакт. На края извежда сумата.

1. **Описание на използваните функции**
2. **Заглавна част на функцията:** void listContacts()

**Действие:** Извежда всички конктаки и тяхната информация до конзолата.

**Параметри:** няма.

**Използвани глобални променливи:** променлива, глобална за обвиващия клас „PhoneBook“, която се казва „head“.

**Извиквани функции:** std::format.

1. **Заглавна част на функцията:** void listAddresses(Contact\* contact)

**Действие:** Извежда всички адреси за даден контакт.

**Параметри:** Указател към елемент на едносвързан списък.

**Използвани глобални променливи:** няма.

**Извиквани функции:** std::format.

1. **Заглавна част на функцията:** void listWorkingContactsCount()

**Действие:** Извежда броя на работещи лица.

**Параметри:** няма.

**Използвани глобални променливи:** променлива, глобална за обвиващия клас „PhoneBook“, която се казва „head“.

**Извиквани функции:** няма.

1. **Заглавна част на функцията:** int promptNumber(string prompt)

**Действие:** Запитва потребителя с параметризирано съобщение и прочите въведено число в конзолата. Връща въведеното число.

**Параметри:** string за съобщението, което да бъде изведено на конзолата.

**Използвани глобални променливи:** няма.

**Извиквани функции:** std::istream::ignore.

1. **Заглавна част на функцията:** Contact\* getContact(int contactI = 1)

**Действие:** Намира контакт съвпадащ на зададения индекс и го връща, ако не го намери извежда съобщение в конзолата и връща nullptr.

**Параметри:** число, индекс на търсеният елемент.

**Използвани глобални променливи:** променлива, глобална за обвиващия клас „PhoneBook“, която се казва „head“.

**Извиквани функции:** няма.

1. **Заглавна част на функцията:** Address\* getAddress(Contact\* contact, int addressI = 1)

**Действие:** Намира адрес по зададен индекс на зададен контакт и го връща, ако не го намери извежда съобщение на конзолата и връща nullptr.

**Параметри:** укъзател към контакт в едносвързан списък и индекс на търсеният адрес.

**Използвани глобални променливи:** няма.

**Извиквани функции:** няма.

1. **Заглавна част на функцията:** void insertContact()

**Действие:** Запитва потребителя за име, което да въведе в телефонния указател и го въвежда, след което пита, дали иска да добави адрес за същото лице. Ако потребителят се съгласи, програмата извиква друга функция.

**Параметри:** няма.

**Използвани глобални променливи:** променливи, глобални за обвиващия клас „PhoneBook“, които се казват „head“ и „tail“.

**Извиквани функции:** insertAddress, std::getline, std::istream::ignore.

1. **Заглавна част на функцията:** void insertAddress(Contact\* contact)

**Действие:** Запитва потребителя за адрес и дали адресът е публичен или частен, след което добавя елемент с информацията в едносвързан списък и запитва потребителя, дали иска да въведе телефонен номер. Накрая запитва, дали да се въведе друг адрес, в който случай се повтарят горе-описаните действия.

**Параметри:** укъзател към контакт за който да се въведе адресът.

**Използвани глобални променливи:** няма.

**Извиквани функции:** insertPhone, std::getline, std::istream::ignore.

1. **Заглавна част на функцията:** void insertPhone(Address\* address)

**Действие:** Запитва потребителя за телефон и добавя елемент с информацията в едносвързан списък. После запитва потребителя, дали иска да въведе друг телефонен номер и се повтарят същите действия.

**Параметри:** укъзател към адрес за който да бъде добавен телефонът.

**Използвани глобални променливи:** няма.

**Извиквани функции:** std::getline, std::istream::ignore.

1. **Заглавна част на функцията:** void run()

**Действие:** Стартира конзолната програма.

**Параметри:** няма.

**Използвани глобални променливи:** няма.

**Извиквани функции:** std::istream::ignore, insertContact, promptNumber, getContact, insertAddress, listAddresses, getAddress, insertPhone, listContacts, listWorkingContactsCount.

1. **Заглавна част на функцията:** int main()

**Действие:** главната функция, вход към програмата, където се инициализира указателя и се извиква функцията за стартиране на конзолното приложение.

**Параметри:** няма.

**Използвани глобални променливи:** няма.

**Извиквани функции:** PhoneBook::run.

1. **Тестови примери (снимки на екрана), доказващи коректността на**

**програмата**

**Тест 1**

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated A screenshot of a computer screen

Description automatically generated A screenshot of a computer menu

Description automatically generated

**Тест 2**A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

1. **Разпечатка на кода**

**#include <iostream>;**

**#include <string>;**

**#include <format>;**

**using namespace std;**

**struct Telephone {**

**public:**

**string phoneNumber;**

**Telephone\* next;**

**Telephone(string telephone) : next(nullptr) {**

**this->phoneNumber = telephone;**

**}**

**};**

**struct Address {**

**public:**

**//1 for home address | 2 for work address**

**int home;**

**string addressName;**

**Telephone\* telephone;**

**Address\* next;**

**Address(string address, int home) : telephone(nullptr), next(nullptr) {**

**this->addressName = address;**

**this->home = home;**

**}**

**};**

**struct Contact {**

**public:**

**string name;**

**Address\* address;**

**Contact\* next;**

**Contact(string name) : address(nullptr), next(nullptr) {**

**this->name = name;**

**};**

**};**

**class PhoneBook {**

**private:**

**//first name entry in the linked list**

**Contact\* tail;**

**//last name entry in the linked list**

**Contact\* head;**

**//List out all the contacts in the linked list, numbered starting from 1**

**void listContacts() {**

**int contactI = 1;**

**Contact\* currContact = head;**

**Address\* currAddress = nullptr;**

**Telephone\* currPhone = nullptr;**

**//check if theres a start to the linked list**

**if (!currContact) { cout << "No contacts" << endl; return; }**

**//loop the linked list and post the stream contact info**

**string contactInfo;**

**while (currContact) {**

**contactInfo = format("\n{}. {}", contactI, currContact->name);**

**//concatenate the string with info for each address**

**currAddress = currContact->address;**

**while (currAddress) {**

**contactInfo += format("\n\nAddress: {}",**

**currAddress->addressName);**

**//concatenate string with each phone number**

**currPhone = currAddress->telephone;**

**while (currPhone) {**

**contactInfo += format("\nTel: {}",**

**currPhone->phoneNumber);**

**currPhone = currPhone->next;**

**}**

**currAddress = currAddress->next;**

**}**

**cout << contactInfo << "\n------------------" << endl;**

**//increment the contact index & get next contact**

**contactI++;**

**currContact = currContact->next;**

**}**

**}**

**//list all addresses of given contact, each indexed starting from 1**

**void listAddresses(Contact\* contact) {**

**int addressI = 1;**

**Address\* currAddress = contact->address;**

**string addressInfo;**

**while (currAddress) {**

**addressInfo = format("\n{}. {}", addressI, currAddress->addressName);**

**cout << addressInfo << endl;**

**addressI++;**

**currAddress = currAddress->next;**

**}**

**}**

**//Gives the total count of working contacts**

**//A contact is a working contact, when they have at least one address**

**void listWorkingContactsCount() {**

**Contact\* currContact = head;**

**int workingCount = 0;**

**while (currContact) {**

**Address\* currAddress = currContact->address;**

**while (currAddress) {**

**if (currAddress->home == 2) { workingCount++; break; }**

**currAddress = currAddress->next;**

**}**

**currContact = currContact->next;**

**}**

**cout << "There are total of " << workingCount << " working contacts" << endl;**

**}**

**//prompt user for a number with the given prompt**

**//return number**

**int promptNumber(string prompt) {**

**int number;**

**cout << prompt;**

**cin >> number;**

**cin.ignore();**

**return number;**

**}**

**//retrieve contact with a given index, index defaults to 1 if left**

**//return Contact\* | nuullptr**

**Contact\* getContact(int contactI = 1) {**

**auto curr = head;**

**int loopI = 1;**

**while (curr && loopI < contactI) { loopI++; curr = curr->next; }**

**//check if both indexes match OR if contact is invalid**

**if (loopI != contactI || !curr) { cout << "Couldn't retrieve contact." << endl;**

**return nullptr; }**

**return curr;**

**}**

**//retrieve address with a given index from given contact, index defaults to 1 if left**

**//return Address\* | nuullptr**

**Address\* getAddress(Contact\* contact, int addressI = 1) {**

**Address\* currAddress = contact->address;**

**int loopI = 1;**

**while (currAddress && loopI < addressI) {**

**loopI++;**

**currAddress = currAddress->next;**

**}**

**//check if both indexes match OR if address is invalid**

**if (loopI != addressI || !currAddress) {**

**cout << "Couldn't retrieve address." << endl; return nullptr;**

**}**

**return currAddress;**

**}**

**//prompt the user for a name and append it to the linked list, then ask if they would like to add an address**

**void insertContact() {**

**string name;**

**//prompt user**

**cout << "Enter full name: ";**

**getline(cin, name);**

**auto contact = new Contact(name);**

**//append the contact to the linked list**

**if (!head) {**

**head = contact;**

**tail = head;**

**}**

**else {**

**tail->next = contact;**

**tail = tail->next;**

**}**

**int enterAddress;**

**//prompt user whether to add an address or not**

**cout << "Would you like to add an address?\n1. Yes\n2. No" << endl;**

**cin >> enterAddress;**

**//ignore \n**

**cin.ignore();**

**if (enterAddress != 1) return;**

**insertAddress(contact);**

**}**

**//prompt user for address and insert it to the given contact**

**void insertAddress(Contact\* contact) {**

**string addressName;**

**int homeAddress;**

**int addAddress = 1;**

**while (addAddress == 1) {**

**cout << "Contact: " << contact->name << endl;**

**cout << "Enter address: ";**

**getline(cin, addressName);**

**//prompt**

**cout << "Is this address a home or working address?\n1. Home\n2. Working" << endl;**

**cin >> homeAddress;**

**cin.ignore();**

**//set initial address**

**auto address = contact->address;**

**if (!address) { contact->address = new Address(addressName, homeAddress); address = contact->address; }**

**else {**

**//get the last valid address node**

**while (address->next) {**

**address = address->next;**

**}**

**//set new address**

**address->next = new Address(addressName, homeAddress);**

**}**

**//prompt user to add a phone**

**int enterPhone;**

**cout << "Would you like to add a phone number?\n1. Yes\n2. No" << endl;**

**cin >> enterPhone;**

**cin.ignore();**

**if (enterPhone == 1) { insertPhone(address); }**

**cout << "Would you like to add another address?\n1. Yes\n2. No" << endl;**

**cin >> addAddress;**

**cin.ignore();**

**}**

**}**

**//prompt user for a phone number and add it to the given address, can add multiple phone numbers**

**void insertPhone(Address\* address) {**

**string phoneNumber;**

**int addPhone = 1;**

**while (addPhone == 1) {**

**cout << "Address: " << address->addressName << endl;**

**cout << "Enter phone number: ";**

**getline(cin, phoneNumber);**

**if (!address->telephone) { address->telephone = new Telephone(phoneNumber); }**

**else {**

**//get the last valid phone node**

**auto phone = address->telephone;**

**while (phone->next) {**

**phone = phone->next;**

**}**

**//set new address**

**phone->next = new Telephone(phoneNumber);**

**}**

**cout << "Would you like to add another number to this address?\n1. Yes\n2. No" << endl;**

**cin >> addPhone;**

**cin.ignore();**

**}**

**}**

**public:**

**PhoneBook() : tail(nullptr), head(nullptr) {}**

**~PhoneBook() {}**

**//method to start the program**

**void run() {**

**const string menu =**

**R"(**

**-----Phonebook Menu-----**

**1. Add contact**

**2. Add address to contact**

**3. Add phone number to address of contact**

**4. List contacts**

**5. List working contacts**

**0. Exit**

**)";**

**int command = 1;**

**while (command != 0) {**

**cout << menu << endl;**

**cout << "Please choose an option: ";**

**cin >> command;**

**//remove \n from buffer**

**cin.ignore();**

**//perform desired command**

**switch (command) {**

**case 1:**

**insertContact();**

**break;**

**case 2: {**

**int contactI = promptNumber("Enter number of contact: ");**

**auto contact = getContact(contactI);**

**if (!contact) break;**

**insertAddress(contact);**

**break;**

**}**

**case 3: {**

**int contactI = promptNumber("Enter number of contact: ");**

**auto contact = getContact(contactI);**

**if (!contact) break;**

**listAddresses(contact);**

**int addressI = promptNumber("Enter number of address: ");**

**Address\* address = getAddress(contact, addressI);**

**if (!address) break;**

**insertPhone(address);**

**break;**

**}**

**case 4:**

**listContacts();**

**break;**

**case 5:**

**listWorkingContactsCount();**

**break;**

**}**

**}**

**}**

**};**

**int main() {**

**PhoneBook x;**

**x.run();**

**return 0;**

**}**