Final Project:
Pharmacy

Kowalczyk Damian 307443

Description

Aim of this program is to manage pharmacy chain. Main tasks:

- Working with pharmacy chain.
- Creating lists of customers, medicines, employees and pharmacies.
- Carrying out operations on medicines, customers, employees, pharmacies
- Data present according to needs.

Special features include: adding, removing and changing all kind of data, finding an alternative for a prescribed medicines, finding pharmacy in the same town which obtain wanted medicine, buying medicines.

Limits & Restrictions

1. Pharmacy chain
   1.1 There have to be at least one pharmacy.
   1.2 Maximum number of pharmacies in chain is 10.

2. Customer
   2.1 Every customer has ID.

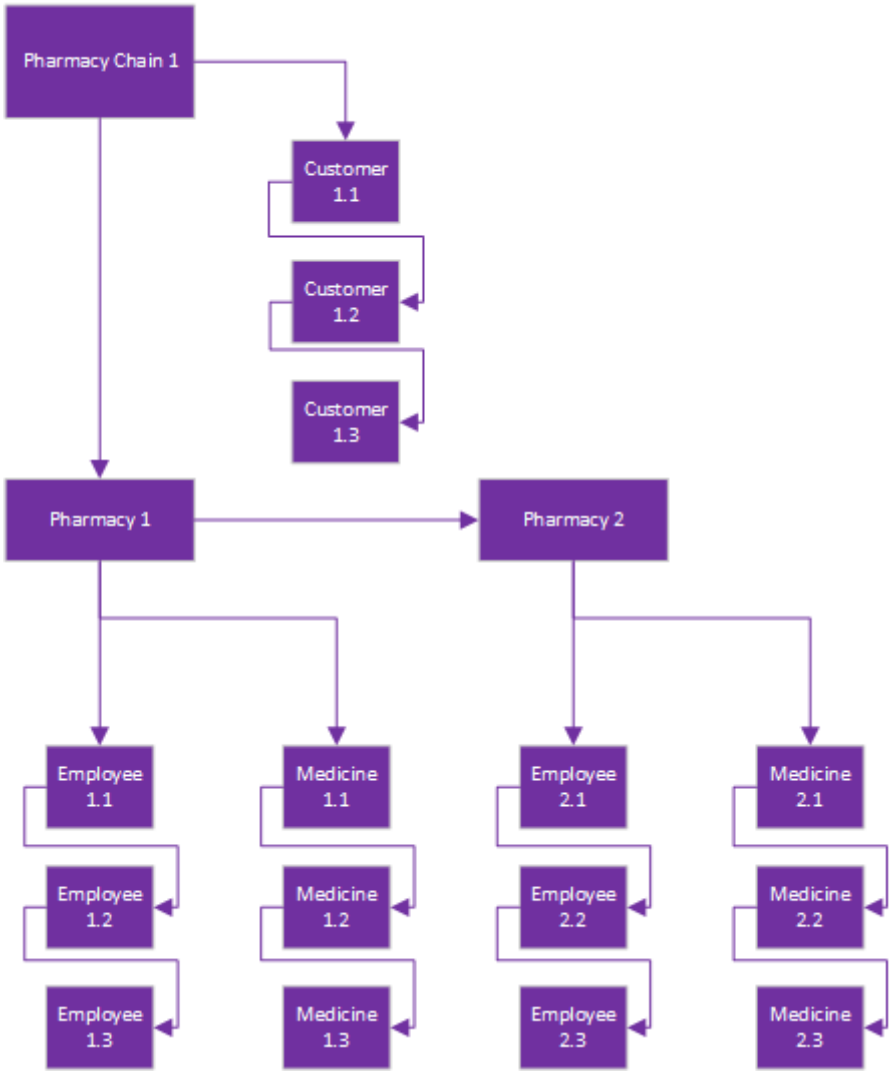
3. Pharmacy
   3.1 Cannot sell a medicine if customer does not have a prescription.
   3.2 There cannot be more medicines than 3000- amount of them.
   3.3 Every pharmacy has ID.

4. Medicine
   4.1 Every medicine has ID.

5. Employee
   5.1 Every employee has ID.
   5.2 Maximum number of employees is 5.

Memory map

This memory map show how objects are connected.



Classes

Class Pharmacy_Chain is the main class which contain linked list of customers and pharmacies. It has many methods to manipulate data of all classes.

```
class Pharmacy_Chain
{
    Pharmacy * headP;
    Customer * headC;
public:
    Pharmacy_Chain();  //Constructor
    ~Pharmacy_Chain(); //Destructor which include deleting all lists
    bool add_Customer(int id, const char * name , const char * surname, int pr
escription);
    //Creates Customer and adds it to a list (This list is a part of Pharmacy_
Chain)
    void remove_Customer(const char * name);
    //Removes customer of a given name from a list
    void remove_1Customer(int id);
    //Removes customer from a list
    bool customerpresent(int id,const char * name , const char * surname);
    //Returns true if customer is present on a list
    int number_of_Customers();
    //Return number of all customers
    int amount_of_1Customer(const char * name);
    //Return number of all customers of given name
    bool add_Pharmacy(int id, const char * name , const char * town);
    //Creates Pharmacy and adds it to a list (This list is a part of Pharmacy_
Chain)
    void remove_Pharmacy(int id);
    //Removes pharmacy of a given id from a list
    int number_of_Pharmacies();
    //Return number of all pharmacies
    int number_of_Pharmacies_in_same_town(const char * name);
    //Return number of all pharmacies located in the same town
    int amount_of_1Pharmacy(const char * name);
    //Return number of all pharmacies of given name
    bool pharmacypresent(int id,const char * name , const char * town);
    //Returns true if pharmacy is present on a list
    bool add_Medicine(int idpharmacy,int id, const char * name , const char *
type,int amount,int precond);
    //Creates Medicine and adds it to a list (This list is a part of Pharmacy)
    bool changeamount_Medicine(int idpharmacy,int amount,int id);
    //New amount is replaced in a given medicine
    int number_of_Medicines(int idpharmacy);
    //Return number of all medicines in a chosen pharmacy
    void remove_MedicinebyID(int idpharmacy,int id);
    //Remove medicine by id from a list
```

```
    bool changeprescription_Medicine(int idpharmacy,int idmedicine);
    //Change the prescription status of a certain medicine to 0 which means no
n prescribed
    bool medicinepresent(int idpharmacy, int id, const char * name , const cha
r * type,int amount,int precond);
    //Returns true if medicine is present on a list
    bool add_Employee(int idpharmacy,int id, const char * name , const char *
surname);
    //Creates Employee and adds it to a list (This list is a part of Pharmacy)
    bool employeepresent(int idpharmacy, int id, const char * name , const cha
r * surname);
    //Returns true if employee is present on a list
    int number_of_Employees(int idpharmacy);
    //Return number of all employees in a chosen pharmacy
    void remove_EmployeebyID(int idpharmacy,int id);
    //Remove employee by id from a list
    void printPC();
    //Print informations of all objects
    void printPL();
    //Print information about all pharmacies
    void printCL();
    //Print information about all customers
    void printML();
    //Print information about all medicines
    void printEL();
    //Print information about all employees
    const char* samemedicineindifferentpharmacy(const char* town,int idcustome
r);
    //Return name of a pharmacy which is in the same town and have a prescript
ion
    const char* give_alternative_to_prescripted_medicine(int medicineID,int ph
armacyID);
    //Return non prescripted alternative for a medicine
    bool transfer_medicine_to_another_pharmacy(int idmedicine, int idpharmacyt
o,int pharmacyfrom);
    //Move medicine from one pharmacy to another (not working)
    bool buy_medicine(int idmedicine, int amount, int idpharmacy, int idcustom
er);
    //Return true if customer is able to buy a medicine(there is enough of it
in the pharmacy
    //and customer has a prescription)
};
```

Class Pharmacy contain linked list of Medicine and Employees, and can manipulate them. It also has attributes which are ID, name, and town.

```
class Pharmacy
{
    int id;
    char * name;
    char * town;
    Medicine * headM;
    Employee * headE;
    Pharmacy * next;
public:
    Pharmacy(int id, const char * name , const char * town); //Constructor
    ~Pharmacy();   //Destructor
    int &rID();    //Reference to private attribute of ID
    char* &rname();//Reference to private attribute of name
    char* &rtown();//Reference to private attribute of town
    Pharmacy * &refNext();
    void set_next(Pharmacy * t);
    void printP();
    //Prints pharmacy data
    void printEE();
    //Prints employees data
    void printMM();
    //Prints medicines data
    bool add_Medicine(int id, const char * name , const char * type,int amount
,int precond);
    //Creates medicine and adds it to a list (This list is a part of Pharmacy)
    void remove_Medicine(const char * name);
    //Removes medicine of a given name from a list
    bool changeamount_Medicine(int amount,int id);
    //Change amount of medicine of a given id
    bool changeprescription_Medicine(int idmedicine);
    //Change status of medicine to non prescripted
    int amount_of_Medicines();
    //Return number of all medicines
    int amount_of_1Medicine(const char * name);
    //Return number of all medicines of given name
    void remove_MedicinebyID(int id);
    //Removes medicine of a given id from a list
    int number_of_allMedicines();
    //Return sum of all medicines in stock
    bool medicinepresent(int id, const char * name , const char * type,int amo
unt,int precond);
    //Returns true if medicine is present on a list
    bool add_Employee(int id, const char * name , const char * surname);
    //Creates employee and adds it to a list (This list is a part of Pharmacy)
    void remove_Employee(int id);
    //Remove employee from a list
```

```
    int amount_of_Employees();
    //Return number of all employee
    void delete_lists();
    //Part of a big destructor which deletes medicine and employee list
    bool employeepresent(int id,const char * name , const char * surname);
    //Returns true if employee is present on a list
    bool buymed(int precond,int amount,int idmedicine);
    //Return true if there is looked medicine with appriopriate amount to buy
    //Utility methods
    Medicine * &  return_headM(int idmedicine);
    bool add_medicinebypointer(Medicine * head);
    const char* give_alternative1(const char* type,int medicineID);
    bool checkmedicineprecond(int precond);
    const char* give_alternative(int medicineID);
    bool check_med(int idmedicine);
};
```

Class Customer plays the role of a list. It also has attributes which are ID, name, surname and prescription status.

```
class Customer
{
    int id;
    char * name;
    char * surname;
    int prescription;
    Customer * next;
public:
    Customer(int id, const char * name , const char * surname, int prescriptio
n); //Constructor
    ~Customer();         //Destructor
    int &rID();          //Reference to private attribute of ID
    char* &rname();      //Reference to private attribute of name
    char* &rsurname();   //Reference to private attribute of surname
    int &rprescription();//Reference to private attribute of prescription
    Customer * &refNext();
    void set_next(Customer * t);
    void printC();       //Print Customer data
};
```

Class Employee plays the role of a list. It also has attributes which are ID, name and surname.

```
class Employee
{
    int id;
    char * name;
    char * surname;
    Employee * next;
public:
    Employee(int id, const char * name , const char * surname); //Constructor
    ~Employee();        //Destructor
    int &rID();         //Reference to private attribute of ID
    char* &rname();     //Reference to private attribute of name
    char* &rsurname();  //Reference to private attribute of surname
    Employee * &refNext();
    void set_next(Employee * t);
    void printE();      //Print Employee data
};
```

Class Medicine plays the role of a list. It also has attributes which are ID, name, type, amount of medicine in stock and status if it is prescripted.

```
class Medicine
{
    int id;
    char * name;
    char * type;
    int amount;
    int precond;
    Medicine * next;
public:
    Medicine(int id, const char * name , const char * type,int amount,int prec
ond); //Constructor
    Medicine(const Medicine & Med);
    ~Medicine();       //Destructor
    int &rID();        //Reference to private attribute of ID
    char* &rname();    //Reference to private attribute of name
    char* &rtype();    //Reference to private attribute of type
    int &ramount();    //Reference to private attribute of amount of medicine
in stock 0 means nonprescripted
    int &rprecond();   //Reference to private attribute of prescription status
    Medicine * &refNext();
    void set_next(Medicine * t);
    void printM();     //Print Medicine data
};
```

Test ideas

1. Pharmacies Chain
   1.1. Try to add many customers and pharmacies.
   1.2. Try to remove customer from an empty list.
   1.3. Try to remove pharmacy from an empty list.
   1.4. Try to overfill maximum number of pharmacies.
   1.5. Check the number of pharmacies after removing some of them.
   1.6. Check the number of customers after removing some of them.
   1.7. Check the number of Pharmacies in the same town.
   1.8. Check if customer can buy a medicine.
   1.9. Check if customer can get medicine in another pharmacy in the same town.
   1.10. Try to move medicine to another pharmacy.
   1.11. Check if there is an alternative of non-prescribed medicine.

2. Customer
   2.1. Try to print data.

3. Pharmacy
   3.1. Try to add many medicines and employees.
   3.2. Try to remove medicine from an empty list.
   3.3. Try to remove employee from an empty list.
   3.4. Check the amount of medicines after change.
   3.5. Try to print data.
   3.6. Try to overfill maximum amount of medicines.

4. Medicine
   4.1. Try to print data.

5. Employee
   5.1. Try to print data.