Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

**ИРКУТСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

Институт информационных технологий и анализа данных

Допускаю к защите

Руководитель \_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

подпись И.О.Фамилия

*Создание приложения для работы с файлом*

Наименование темы

**Пояснительная записка**

**к курсовой работе**

по дисциплине «Программирование»

1.043.00.00 ПЗ

**Разработал студент группы АСУб-20-2** \_\_\_\_\_\_\_\_ Арбакова А.В.

подпись И.О.Фамилия

**Нормоконтроль**  \_\_\_\_\_\_\_\_ Бахвалов С.В.

подпись И.О.Фамилия

**Курсовая работа защищена с оценкой** \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Иркутск 2021**

**Вариант 29**

**Задание**

Написать программу, регистрирующую следующие сведения о посетителях поликлиники: Ф.И.О., дата рождения, номер полиса, диагноз, Ф.И.О. лечащего врача, дата посещения. Определить нагрузку каждого врача, выдать статистику по заболеваниям (в виде диаграммы), сформировать список пациентов, страдающих хроническими заболеваниями (т.е. обращавшихся к врачу более 3 раз в течение года с одним и тем же диагнозом).

Оглавление

Введение и постановка задачи **4**

Таблица спецификаций глобальных переменных и переменных модуля main(), включая структуру записи файла **5**

Схема иерархии модулей**7**

Спецификация функций**8**

Алгоритмизация для каждой функции**10**

Таблица тестов **14**

Результаты тестирования **16**

Результаты работы программы **21**

Листинги файла проекта и всех модулей программы **23**

Руководство пользователя **40**

Заключение**42**

Список использованной литературы **43**

Введение и постановка задачи

В процессе выполнения курсовой работы планируется приобрести навыки по созданию windows-приложений в среде программирования «Dev-C++», на языке программирования С++.

Были поставлены задачи: разработки и написания алгоритма, составления таблицы спецификаций и проведения тестирования программы.

В ходе выполнения курсовой работы планируется разработать такие функции, как добавление новых данных в таблицу, их удаление. открытие, закрытие, сохранение файла, обработка запросов.

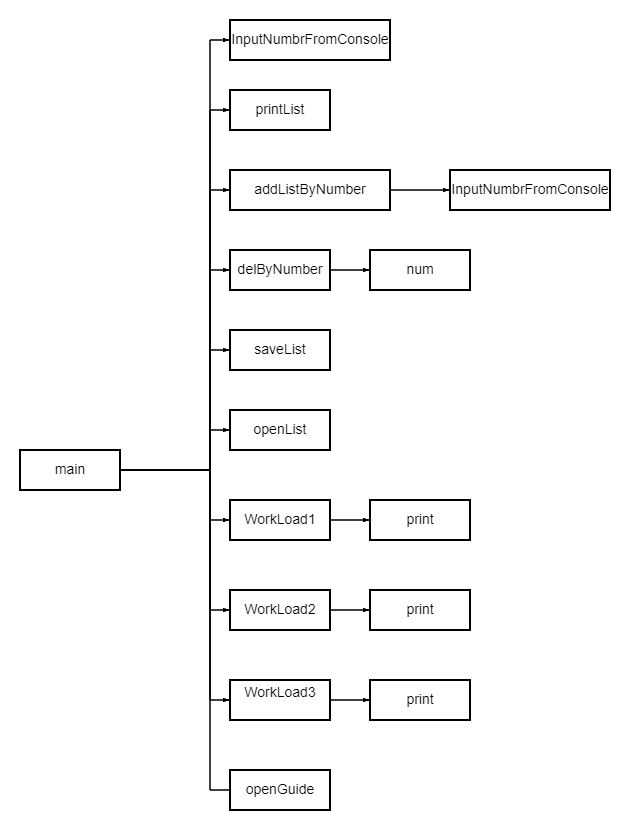
В программе планируется исправно обрабатывать запросы пользователя по составлению списка животных каждого ответственного, определения количества животных каждого вида и определения рациона питания животных. В процессе выполнения работы будут решены следующие задачи:

1. Разработка алгоритма решения поставленной задачи
2. Создание таблицы спецификаций
3. Составление иерархии модулей программы
4. Словесное описание используемого алгоритма
5. Написание программы
6. Тестирование и отладка программы

**Таблица спецификаций глобальных переменных и переменных модуля main(), включая структуру записи файла**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Имя | Назначение | Тип | ОДЗ |
| 11 | polyclinic | Структура данных, содержащая полную информацию о записи пациента. | struct | - |
| 22 | wLoadOne | Список нагрузки врачей. | class | - |
| 33 | wLoadTwo | Список статистики заболеваний. | class | - |
| 44 | wLoadThree | Список пациентов с хроническими заболеваниями. | class | - |
| 55 | \*head | Указатель на начало списка | polyclinic | - |
| 66 | mainMenu | Выбор пункта в главном меню | int | [0;3] |
| 77 | menu | Выбор пункта в под-меню | int | [0;8] |
| 88 | menu1 | Выбор пункта в под-меню | int | [0;2] |
| 99 | check | Логическая переменная в цикле while для повторного вывода меню | bool | - |
| 110 | check1 | Логическая переменная в цикле while для повторного вывода меню | bool | - |
| 111 | check2 | Логическая переменная в цикле while для повторного вывода меню | bool | - |
| 112 | number | Счетчик. | int | [0;99] |
| 113 | number1 | Для ввода номера для функции. | int | [1;99] |
| 114 | number2 | Для ввода номера для функции. | int | [1;99] |
| 115 | dayP | Переменная дня даты рождения пациента. | int | [1;31] |
| 116 | monthP | Переменная месяца даты рождения пациента. | int | [1;12] |
| 117 | yearP | Переменная года даты рождения пациента. | int | [1900;2022] |
| 118 | dayVisit | Переменная дня даты посещения поликлиники пациентом. | int | [1;31] |
| 119 | monthVisit | Переменная месяца даты посещения поликлиники пациентом. | int | [1;12] |
| 220 | yearVisit | Переменная года даты посещения поликлиники пациентом. | int | [1900;2022] |
| 221 | policyP | Переменная полиса пациента. | int | [1;999999999999] |
| 222 | fioP | Переменная, содержащая антропоним пациента. | string | - |
| 223 | fioD | Переменная, содержащая антропоним доктора. | string | - |
| 224 | diagnosisP | Переменная, содержащая диагноз пациента. | string | - |

**Схема иерархии модулей**

**Спецификация функций**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название | Назначение | Параметры |
| 11 | InputNumberFromConsole | Ввод числа с консоли. | Int min – Минимальное значение, которое может принимать проверяемое число.  Int max – Максимальное значение, которое может принимать проверяемое число.  string ask – Сообщение. |
| 22 | printList | Вывод структуры. | polyclinic\* head – указатель на структуру.  polyclinic\*\* pHead – указатель на структуру. |
| 33 | addListByNumber | Добавление элемента в структуру. | polyclinic\*\* pHead – указатель на структуру. |
| 44 | delByNumber | Удаление по номеру. | polyclinic\*\* pHead – указатель на структуру.  int number1 – введенный номер для удаления. |
| 55 | num | Проверка для функции удаления, ведущая подсчет количества элементов в структуре. | polyclinic\*\* pHead – указатель на структуру. |
| 66 | saveList | Сохранение структуры в текстовый файл. | polyclinic\*\* pHead – указатель на структуру.  polyclinic\* head – указатель на структуру. |
| 77 | openList | Открытие текстового файла в структуру. | polyclinic\*\* pHead – указатель на структуру.  polyclinic\* head – указатель на структуру. |
| 88 | WorkLoad1 | Обработка нагрузки врачей. | polyclinic\*\* pHead – указатель на структуру.  polyclinic\* head – указатель на структуру. |
| 99 | WorkLoad2 | Обработка статистики по заболеваниям. | polyclinic\*\* pHead – указатель на структуру.  polyclinic\* head – указатель на структуру. |
| 110 | WorkLoad3 | Обработка списка пациентов с хроническими заболеваниями. | polyclinic\*\* pHead – указатель на структуру.  polyclinic\* head – указатель на структуру. |
| 111 | print | Печать списка. | - |
| 112 | openGuide | Открытие текстового файла с руководством пользователя. | - |

**Алгоритмизация для каждой функции**

**Модуль main:**

1. Вывод на экран главного меню с 4 опциями:
   1. Задание.
   2. Работа со списком.
   3. Общие сведения о программе.
   4. Выход из программы.
2. Вызов функции InputNumberFromConsole для ввода варианта меню.

**При выборе пункта 1.1. в main:**

1. Вывод условия задания.

**При выборе пункта 1.2. в main:**

1. Вызов функции printList.
2. Вызов функции InputNumberFromConsole для ввода варианта меню.
3. Вывод меню с 9 опциями:
   1. Добавить элемент в структуру поликлиники.
      1. Вызов функции addListByNumber.
   2. Изменить элемент из списка.
      1. Вызов функции InputNumberFromConsole для ввода номера изменяемого элемента.
      2. Проверка на существование элемента с использованием функции удаления delByNumber, возвращение в меню при отсутствии элемента.
      3. Добавление элемента в список.
   3. Удалить элемент из списка.
      1. Вызов функции InputNumberFromConsole для ввода номера изменяемого элемента.
      2. Проверка на существование элемента с использованием функции удаления delByNumber, возвращение в меню при отсутствии элемента.
   4. Сохранить список в файл.
      1. Вызов функции saveList.
   5. Открыть файл.
      1. Вызов функции openList.
   6. Определить нагрузку каждого врача.
      1. Вызов функции WorkLoad1.
   7. Определить статистику по заболеваниям.
      1. Вызов функции WorkLoad2.
   8. Определить статистику хронических заболеваний у пациентов.
      1. Вызов функции WorkLoad3.
   9. Выход в главное меню.

**При выборе пункта 1.3. в main:**

1. Вызов функции InputNumberFromConsole для ввода варианта меню.
2. Вывод меню с 3 опциями:
   1. Вывод руководства пользователя из файла с помощью функции openGuide с возвращением в главное меню.
   2. Вывод информации о программе.
   3. Выход в главное меню.

**Описание функций:**

**InputNumberFromConsole**

Параметры:

Int min – Минимальное значение, которое может принимать проверяемое число.

Int max - Максимальное значение, которое может принимать проверяемое число.

String ask – Сообщение, которое выводится пользователю перед проверкой.

Описание функции: Осуществление проверки ввода натурального числа.

Алгоритмизация:

1. Инициализация int number.
2. Вывод ask.
3. Делать:
   1. Ввод number для сheck.
   2. Если введено не число:
      1. Очищение ввода.
      2. Вывод сообщения об ошибке.
   3. Если введено число, выходящее за диапазона:
      1. Очистка ввода.
      2. Вывод сообщения об ошибке.
4. Пока (Условия проверки ввода).
5. Вернуть number.

**printList**

1. Присвоение number=0.
2. Если список пуст:
   1. Вывод сообщения об этом.
3. Иначе выводим заголовки и пока список не заканчивается выводим содержимое структуры.

**addListByNumber**

1. Последовательный ввод с консоли данных о записи пациента.
2. Если список пуст, создаем первый новый элемент.
3. Иначе по указателю перемещаемся в конец списка и добавляем элемент.

**num**

1. Инициализация number=1.
2. Вводим указатели, для перемещения по списку и после считаем количество элементов в структуре, переходя по указателям, пока указатель не станет нулем.
3. Возвращение значения number.

**delByNumber**

1. Если список пуст возвращаем false, означающий пустоту списка.
2. Введенный number1 сравниваем с number, вызванный в функции num. Если number1 больше, то возвращаем false, означающий пустоту списка.
3. Иначе перемещаемся по указателям на значение number1, удаляем элемент по введенному номеру.
4. Возвращаем true, означающее выполненость.

**saveList**

1. Если список пуст, выводим сообщение об этом.
2. Иначе создает файл для записи структуры.
   1. Перемещаясь по указателю вводим последовательно данные структуры в текстовый файл.
3. Закрытие файла.

**openList**

1. Создаем файл для ввода из текстового файла в структуру.
2. Если файла не существует, выводим сообщение об этом.
3. Иначе пока файл не заканчивается:
   1. Записываем данные в переменные.
   2. Если данные закончились, то выходим.
   3. Если список пуст, то создаем новую структуру.
   4. Иначе находим последний элемент и записываем данные в структуру.
4. Закрытие файла.

**WorkLoad1**

1. Создаем list.
2. Создаем итератор.
3. Если список пуст, выводим сообщение об этом.
4. Иначе:
   1. Переменная founded=false.
   2. Пока список есть, ищем повторения по фамилии доктора.
      1. При нахождении счетчик увеличивается на 1, т.е. нахождение повторений.
      2. Переменная founded=true.
   3. Если повторения посчитаны, то данные записываются в list.
5. Вывод в цикле нагрузка каждого врача.

**WorkLoad2**

1. Создаем list.
2. Создаем итератор.
3. Если список пуст, выводим сообщение об этом.
4. Иначе:
   1. Переменная founded=false.
   2. Пока список есть, ищем повторения по заболеваниям.
      1. При нахождении счетчик увеличивается на 1, т.е. нахождение повторений.
      2. Переменная founded=true.
   3. Если повторения посчитаны, то данные записываются в list.
5. Вывод в цикле статистика по заболеваниям.

**WorkLoad3**

1. Создаем list.
2. Создаем итератор.
3. Если список пуст, выводим сообщение об этом.
4. Иначе:
   1. Переменная founded=false.
   2. Пока список есть, ищем повторения по диагнозу, антропониму пациента и году посещения.
      1. При нахождении счетчик увеличивается на 1, т.е. нахождение повторений.
      2. Переменная founded=true.
   3. Если повторения посчитаны, то данные записываются в list.
5. Вывод в цикле статистика по хроническим заболеваниям пациентов.

**openGuide**

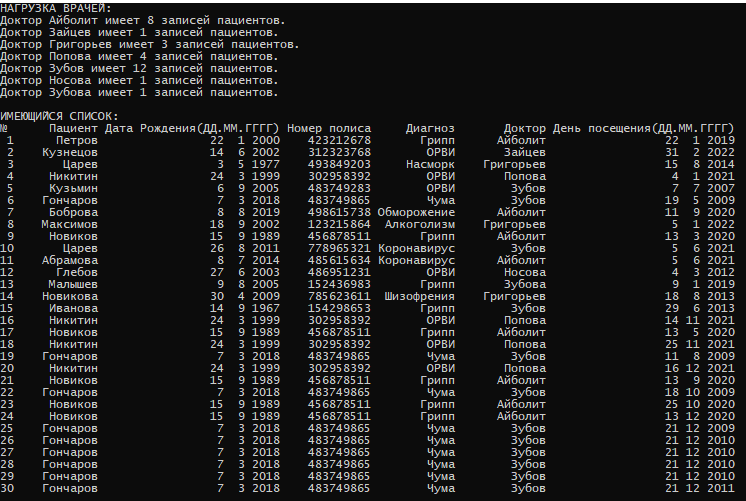
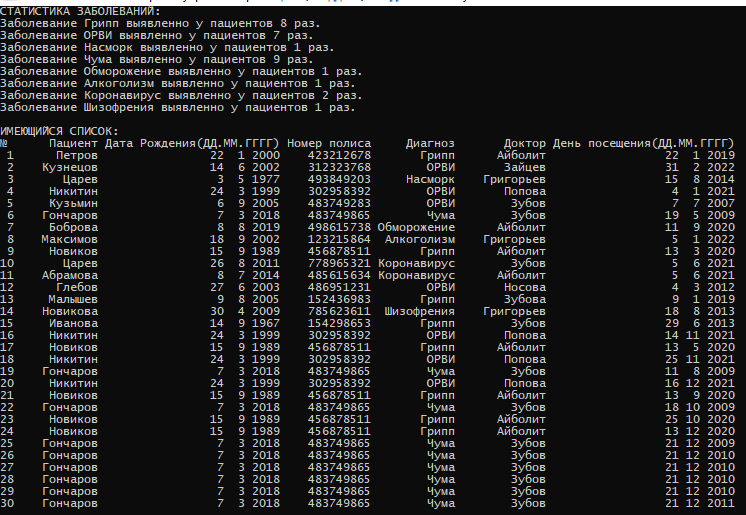
1. Создаем файл для ввода из текстового файла в структуру.
2. Если файла не существует, выводим сообщение об этом.
3. Иначе пока файл не заканчивается, построчно выводим содержимое файла.
4. Закрытие файла.

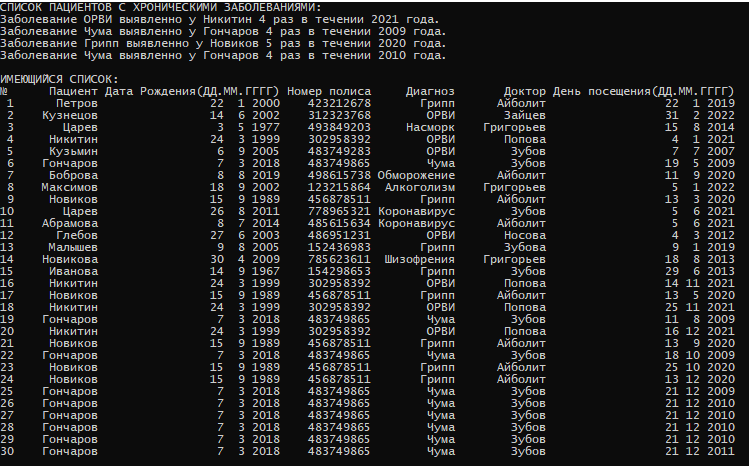
**Таблица тестов**

|  |  |  |
| --- | --- | --- |
| № | Действие | Результат |
| 1 | Запуск программы | Запустив программу выводится главное меню  (рис. 1) |
| Главное меню | | |
| 2 | Выбор пункта меню 1 (главное меню) | После выбора пункта меню на экран выводится задание по курсовой работе  (рис. 2) |
| 3 | Ввод некорректных данных при выборе меню  (символы “#”, “!”, “/” и др.) | Выполняется вывод сообщения об ошибке и запрос о повторном вводе (рис 3) |
| 4 | Выбор пункта меню 2 (главное меню) | После выбора пункта меню выводится список и меню с возможными действиями (меню№1)  (рис 4) |
| 5 | Выбор пункта меню 3 (главное меню) | После выбора пункта меню выводится меню с руководством и информацией (меню№2)  (рис 5) |
| 6 | Выбор пункта меню 0 (главное меню) | После выбора пункта меню осуществляется завершение работы программы |
| Меню №1 | | |
| 7 | Выбор пункта 1  Добавление элемента в список | Пользователю предлагается ввести количество вводимых элементов, далее идет добавление элемента в список  (рис 6) |
| Ввод некорректного значения при добавлении | Вывод сообщения об ошибке  (рис 7) |
| 8 | Выбор пункта 2  Изменение элемента в списке | Элемент с заданным критерием изменен  (рис 8) |
| Ввод некорректного значения при изменении | Вывод сообщения об ошибке  (рис 9) |
| 9 | Выбор пункта 3  Удаление элемента в списке | Элемент с заданным критерием удален  (рис 10) |
| Ввод некорректного значения при изменении | Вывод сообщения об ошибке  (рис 11) |
| 10 | Выбор пункта 4  Сохранение в файл | Программа при наличии списка сохраняет список записей пациентов (рис 12) |
| 11 | Выбор пункта 5  Открытие файла | Программа при наличии списка открывает список записей пациентов (рис 13) |
| 12 | Выбор пункта 6  Определить нагрузку каждого врача | Программа при наличии списка выводит список с нагрузкой каждого врача  (рис 14) |
| Вывод при отсутствии списка | Вывод сообщения  (рис 15) |
| 13 | Выбор пункта 7  Выдать статистику по заболеваниям | Программа при наличии списка выводит статистику по заболеваниям (рис 16) |
| Вывод при отсутствии списка | Вывод сообщения  (рис 17) |
| 14 | Выбор пункта 8  Сформировать список пациентов с хроническими заболеваниями | Программа при наличии списка выводит список пациентов с хроническими заболеваниями  (рис 18) |
| Вывод при отсутствии списка | Вывод сообщения  (рис 19) |
| 15 | Выбор пункта 0  Выход в главное меню | После выбора пункта осуществляется выход в главное меню |
|  | Меню №2 | |
| 16 | Выбор пункта 1  Вывод руководства пользователя | Загрузка руководства пользователя (рис 20) |
| 21 | Выбор пункта 2  Вывод информации о программе | Вывод информации о программе  (рис 21) |
| 22 | Выбор пункта 0  Выход в главное меню | После выбора пункта осуществляется выход в главное меню |

**Результаты тестирования**

|  |  |
| --- | --- |
| № рисунка | Результат |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |
| 16 |  |
| 17 |  |
| 18 |  |
| 19 |  |
| 20 |  |
| 21 |  |

**Результаты работы программы **

****

**Листинги файла проекта и всех модулей программы**

#include <iostream>

#include <fstream>

#include <istream >

#include <string>

#include <Windows.h>

#include <iomanip>

#include <list>

#include <iterator>

using namespace std;

struct polyclinic

{

int numberList; //порядковый номер

string FIOpatient; //фио пациента

int birthDay; //день р пациента

int birthMonth; //месяц р пациента

int birthYear; //год р пациента

int policyNumber; //номер полиса

string diagnosis; // диагноз

string FIOdoctor; //фио доктора

int visitDay; //день посещения

int visitMonth; //месяц посещения

int visitYear; //год посещения

polyclinic\* next;

};

int InputNumberFromConsole(int min, int max, string ask) //ввод числа для меню в диапазоне

{

int number;

bool check;

do{

cout << ask;

check = false;

cin >> number;

if (cin.peek() != '\n'){

cout << "Ошибка. Попробуйте снова.\n" << endl;

cin.clear();

cin.ignore(2147483647, '\n');

check = true;

}

if (!check && (number < min || number > max)){

cout << "Недопустимый диапазон. Попробуйте снова.\n";

cin.clear();

cin.ignore(2147483647, '\n');

check = true;

}

} while (check);

return number;

}

void openGuide(){ //открытие руководства из текстового файла

cout<<"РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ:\n";

ifstream file;

file.open("Guide.txt");

if (!file){

cout<<"Отсутствует файл для открытия!\n";

}

else{

string str;

while(!file.eof()){

str="";

getline(file,str);

cout<<str<<endl;}

cout<<endl;

file.close();

}

}

int number = 0;

int dayP;

int monthP;

int yearP;

int dayVisit;

int monthVisit;

int yearVisit;

int policyP;

string fioP;

string diagnosisP;

string fioD;

void addListByNumber(polyclinic\*\* pHead) //добавление элемента в структуру

{

cout << "ДОБАВЛЕНИЕ ЭЛЕМЕНТА В СПИСОК\n";

number++;

cout<<"Пациент: ";

cin.ignore(332767,'\n');

getline(cin,fioP);

cout<<"Дата рождения(день, месяц, год):\n";

dayP = InputNumberFromConsole(1, 31, "День: ");

monthP = InputNumberFromConsole(1, 12, "Месяц: ");

yearP = InputNumberFromConsole(1900, 2022, "Год: ");

int policy=9999999999999999;

policyP = InputNumberFromConsole(1, policy, "Номер полиса: ");

cout<<"Диагноз: ";

cin.ignore(332767,'\n');

getline(cin,diagnosisP);

dayVisit = InputNumberFromConsole(1, 31, "День: ");

monthVisit = InputNumberFromConsole(1, 12, "Месяц: ");

yearVisit = InputNumberFromConsole(1900, 2022, "Год: ");

cout<<"Доктор: ";

cin.ignore(332767,'\n');

getline(cin,fioD);

if (\*pHead==0){

\*pHead = new polyclinic{number, fioP, dayP, monthP, yearP, policyP, diagnosisP, fioD, dayVisit, monthVisit, yearVisit, 0};

}

else{

polyclinic\* ch = \*pHead;

while (ch->next) ch=ch->next;

ch->next=new polyclinic{number, fioP, dayP, monthP, yearP, policyP, diagnosisP, fioD, dayVisit, monthVisit, yearVisit, 0};

}

}

void printList (polyclinic\* head, polyclinic\*\* pHead) //вывод структуры

{

number=0;

cout << "ИМЕЮЩИЙСЯ СПИСОК:\n";

if (\*pHead==0){

cout<<"Список пуст.\n";

}

else{

cout<<setw(2)<<"№ "<<setw(12)<<"Пациент"<<setw(18)<<"Дата Рождения(ДД."<<setw(3)<<"ММ."<<setw(5)<<"ГГГГ)"<<setw(13)<<"Номер полиса"<<setw(12)<<"Диагноз"<<setw(13)<<"Доктор"<<setw(19)<<"День посещения(ДД."<<setw(3)<<"ММ."<<setw(5)<<"ГГГГ)\n";

while(head){

number++;

cout<<setw(2)<<number<<setw(12)<<head->FIOpatient<<setw(18)<<head->birthDay<<setw(3)<<head->birthMonth<<setw(5)<<head->birthYear<<setw(13)<<head->policyNumber<<setw(12)<<head->diagnosis<<setw(13)<<head->FIOdoctor<<setw(19)<<head->visitDay<<setw(3)<<head->visitMonth<<setw(5)<<head->visitYear<<"\n";

head = head->next;

}

}

cout<<endl;

}

int num(polyclinic\*\* pHead) //для проверки функции удаления, подсчет количества элементов в структуре

{

int number = 1;

polyclinic\* ptr = \*pHead;

polyclinic\* temp = \*pHead;

do{

ptr = temp;

temp = temp->next;

number++;

} while (temp->next != NULL);

return number;

}

int delByNumber(polyclinic\*\* pHead, int number1) //удаление по номеру

{

if (\*pHead==0)

return false;

polyclinic\* ptr = \*pHead;

polyclinic\* temp = \*pHead;

int number = num(pHead);

if (number1 > number)

return false;

else{

for (int i = 1; i < number1; i++){

ptr = temp;

temp = temp->next;

}

if (number1 == 1)

\*pHead = (\*pHead)->next;

ptr->next = temp->next;

delete temp;

return true;

}

return false;

}

void saveList(polyclinic\*\* pHead, polyclinic\* head){ //сохранение структуры в файл

if (\*pHead==0){

cout<<"Невозможно сохранить пустой список. (СПИСОК ПУСТ)\n";

}

else{

ofstream file;

file.open("newfile.txt");

number=0;

while(head){

number++;

file<<number<<" "<<head->FIOpatient<<" "<<head->birthDay<<" "<<head->birthMonth<<" "<<head->birthYear<<" "<<head->policyNumber<<" "<<head->diagnosis<<" "<<head->FIOdoctor<<" "<<head->visitDay<<" "<<head->visitMonth<<" "<<head->visitYear<< " "<<"\n";

head = head->next;

}

file.close();

}

}

void openList(polyclinic\*\* pHead, polyclinic\* head){ //открытие файла

ifstream file;

file.open("newfile.txt");

if (!file){

cout<<"Отсутствует файл для открытия!\n";

}

else{

\*pHead = 0;

while(!file.eof()){

number=0;

file>>number;

file>>fioP;

file>>dayP; file>>monthP; file>>yearP;

file>>policyP;

file>>diagnosisP;

file>>fioD;

file>>dayVisit; file>>monthVisit; file>>yearVisit;

if (number == 0)

break;

if (\*pHead==0){

\*pHead = new polyclinic{number, fioP, dayP, monthP, yearP, policyP, diagnosisP, fioD, dayVisit, monthVisit, yearVisit, 0};

}

else{

polyclinic\* ch = \*pHead;

while (ch->next) ch=ch->next;

ch->next=new polyclinic{number, fioP, dayP, monthP, yearP, policyP, diagnosisP, fioD, dayVisit, monthVisit, yearVisit, 0};

}

}

}

file.close();

}

class wLoadOne{ //для 1 подзадачи

public:

string FIOdoctor;

int PatientCount=0;

wLoadOne(string Doc){

FIOdoctor=Doc;

PatientCount=1;}

void print(){

cout<<"Доктор "<<FIOdoctor<<" имеет "<<PatientCount<<" записей пациентов.\n";}

};

void WorkLoad1 (polyclinic\* head, polyclinic\*\* pHead) //1 подзадача по выводу нагрузки врачей

{

cout<< "НАГРУЗКА ВРАЧЕЙ:\n";

list <wLoadOne> loadList;

list <wLoadOne> :: iterator i;

if (\*pHead==0){

cout<<"Список пуст.\n";

}

else{

bool founded=false;

while(head){

founded=false;

for (i=loadList.begin(); i != loadList.end();++i)

if ((\*i).FIOdoctor==head->FIOdoctor){

(\*i).PatientCount+=1;

founded=true;

}

if (founded==false){

loadList.push\_back(wLoadOne(head->FIOdoctor));}

head = head->next;

}

}

for (i=loadList.begin(); i != loadList.end();++i)

(\*i).print();

cout<<endl;

}

class wLoadTwo{ //для 2 подзадачи

public:

string diagnosis;

int diagnosisCount=0;

wLoadTwo(string Dia){

diagnosis=Dia;

diagnosisCount=1;}

void print(){

cout<<"Заболевание "<<diagnosis<<" выявленно у пациентов "<<diagnosisCount<<" раз.\n";}

};

void WorkLoad2 (polyclinic\* head, polyclinic\*\* pHead) //2 подзадача по выводу статистики по заболеваниям

{

cout<< "СТАТИСТИКА ЗАБОЛЕВАНИЙ:\n";

list <wLoadTwo> loadList;

list <wLoadTwo> :: iterator i;

if (\*pHead==0){

cout<<"Список пуст.\n";

}

else{

bool founded=false;

while(head){

founded=false;

for (i=loadList.begin(); i != loadList.end();++i)

if ((\*i).diagnosis==head->diagnosis){

(\*i).diagnosisCount+=1;

founded=true;

}

if (founded==false){

loadList.push\_back(wLoadTwo(head->diagnosis));}

head = head->next;

}

}

for (i=loadList.begin(); i != loadList.end();++i)

(\*i).print();

cout<<endl;

}

class wLoadThree{ //для 3 подзадачи

public:

string FIOpatient;

string diagnosis;

int diagnosisCount=0;

int visitYear=0;

wLoadThree(string Pat, string Dia, int vYear){

FIOpatient=Pat;

diagnosis=Dia;

diagnosisCount=1;

visitYear=vYear;}

void print(){

if (diagnosisCount>3)

cout<<"Заболевание "<<diagnosis<<" выявленно у "<<FIOpatient<<" "<<diagnosisCount<<" раз в течении "<<visitYear<< " года.\n";}

};

void WorkLoad3 (polyclinic\* head, polyclinic\*\* pHead) //3 подзадача по выводу хронических заболеваний пациентов

{

cout<< "СПИСОК ПАЦИЕНТОВ С ХРОНИЧЕСКИМИ ЗАБОЛЕВАНИЯМИ:\n";

list <wLoadThree> loadList;

list <wLoadThree> :: iterator i;

if (\*pHead==0){

cout<<"Список пуст.\n";

}

else{

bool founded=false;

while(head){

founded=false;

for (i=loadList.begin(); i != loadList.end();++i){

if (((\*i).diagnosis==head->diagnosis) && ((\*i).FIOpatient==head->FIOpatient) && ((\*i).visitYear==head->visitYear)){

(\*i).diagnosisCount+=1;

founded=true;

}

}

if (founded==false){

loadList.push\_back(wLoadThree(head->FIOpatient, head->diagnosis, head->visitYear));}

head = head->next;

}

}

for (i=loadList.begin(); i != loadList.end();++i)

(\*i).print();

cout<<endl;

}

int main()

{

SetConsoleCP(1251); //русский язык в консоли на ввод и вывод

SetConsoleOutputCP(1251);

int mainMenu, menu, menu1;

bool check = true, check1 = true, check2 = true; //флажок для do while заканчивается при выходе из программы

polyclinic\* head = 0;

int number1, number2;

do{

cout << "\nГЛАВНОЕ МЕНЮ:\n1. Постановка задачи (условие).\n2. Работа со списком. \n3. Общие сведения о программе.\n0. Выйти из программы.";

mainMenu = InputNumberFromConsole(0, 3, "\n\nВЫБЕРИТЕ ПУНКТ МЕНЮ: ");

system("cls"); //очищает консоль

switch (mainMenu)

{

case 1: //Вывод условия задания

cout << "29 ВАРИАНТ.\nНаписать программу, регистрирующую следующие сведения о посетителях поликлиники: Ф.И.О., дата рождения, номер полиса, диагноз, Ф.И.О. лечащего врача, дата посещения.\nОпределить нагрузку каждого врача, выдать статистику по заболеваниям, сформировать список пациентов, страдающих хроническими заболеваниями (т.е. обращавшихся к врачу более 3 раз в течение года с одним и тем же диагнозом).\n";

break;

case 2: //Действия задачи

do{

printList (head, &head);

cout << "РАБОТА СО СПИСКОМ:\n1. Добавить элемент списка.\n2. Изменить элемент списка. \n3. Удалить элемент списка.\n4. Сохранить список в файл.\n5. Открыть файл со списком.\nЗАДАНИЯ:\n6. Определить нагрузку каждого врача.\n7. Выдать статистику по заболеваниям.\n8. Сформировать список пациентов, страдающих хроническими заболеваниями.\n0. Выход в главное меню.";

menu = InputNumberFromConsole(0, 8, "\n\nВЫБЕРИТЕ ПУНКТ МЕНЮ: ");

system("cls");

switch (menu)

{

case 1: // Добавить элемент списка

addListByNumber(&head);

system("cls");

break;

case 2: // Изменить элемент списка

number2 = InputNumberFromConsole(1, 99, "ВВЕДИТЕ НОМЕР ДЛЯ ИЗМЕНЕНИЯ: ");

if (delByNumber(&head, number2) == false){

cout << "Такого элемента не существует. Введите правильный номер элемента.\n\n";

break;

}

number = 0;

addListByNumber(&head);

system("cls");

break;

case 3: // Удалить элемент списка

number1 = InputNumberFromConsole(1, 99, "ВВЕДИТЕ НОМЕР ДЛЯ УДАЛЕНИЯ: ");

if (delByNumber(&head, number1) == false){

cout << "Такого элемента не существует. Введите правильный номер элемента.\n\n";

}

number = 0;

break;

case 4: // Сохранить в файл список

saveList(&head, head);

break;

case 5: // Открыть файл

openList(&head, head);

break;

case 6: // Нагрузка каждого врача

WorkLoad1 (head, &head);

break;

case 7: // Статистика заболеваний

WorkLoad2 (head, &head);

break;

case 8: // Список пациентов с хроническими

WorkLoad3 (head, &head);

break;

case 0:

check1 = false; //выход в главное меню

system("cls");

break;

}

} while (check1);

break;

case 3: //руководство и информация

do{

cout << "ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ:\n1. Руководство пользователя.\n2. Информация о программе. \n0. Выйти в главное меню.";

menu1 = InputNumberFromConsole(0, 2, "\n\nВЫБЕРИТЕ ПУНКТ МЕНЮ: ");

system("cls");

switch (menu1)

{

case 1: // Руководство

openGuide();

break;

case 2: // Информация

cout << "ИНФОРМАЦИЯ О ПРОГРАММЕ:\nBaseOfPolyclinic\n15.03.2022\nCopyright © 2022 AnastasiaArbakova. All rights reserved.\nРоссия, г. Иркутск.\nПочта:arbakova.nastya@gmail.ru\n\n";

break;

case 0: //Выход в гл меню

check2 = false;

system("cls");

break;

}

} while(check2);

break;

case 0: // Выход из программы

check = false;

system("cls");

break;

}

} while(check);

return 0;

}

**Руководство пользователя**

BaseOfPolyclinic - программа, регистрирующая сведения о записях пациентов в поликлинике.

Функциональное назначение программы.

1. Программа предназначена для непосредственного ввода информации и её редактирования.

2. Программа содержит меню, с помощью которого осуществляется взаимодействие с программой.

Обращение к программе.

1. Осуществление обращения к программе производится через консоль.

Входные и выходные данные.

1. Осуществление ввода данных производится через клавиатуру или импортируются из файла.

2. Осуществление вывода данных производится из файла или ОЗУ.

Назначение и условия применения программы

1. Описание возможностей пунктов главного меню.

О главном меню.

В предложенном сообщении при вводе:

1 - Выводится информации о задании, с описанием установленного функционала.

2 - Открывается меню с возможными действиями, работающими с данными.

3 - Открывается меню с руководством пользователя и сведениями о программе.

0 - Осуществляется выход из программы.

О под-меню при вводе "2" в главном меню.

Загружается изначально пустой список записей пациентов в поликлинике.

В предложенном сообщении при вводе:

1 - Осуществляется добавление записи пациента вручную пользователем.

2 - Предлагается ввод номера записи пользователем, и после изменяется запись пациента, при условии что такая запись существует.

3 - Предлагается ввод номера записи пользователем, и после удалется запись пациента, при условии что такая запись существует.

4 - Имеющийся и ранее воспроизведенный список сохраняется в текстовый файл.

5 - Данные из текстового файла записываются в имеющийся и ранее воспроизведенный список, заменяя былые данные, при их существовании.

6 - По имеющемуся списку формируется нагрузка каждого врача, чем является количество записей пациентов у конкретного доктора.

7 - По имеющемуся списку формируется статистика по заболеваниям пациентов.

8 - По имеющемуся списку формируется список пациентов с хроническими заболеваниями.

0 - Осуществляется выход в главное меню.

О под-меню при вводе "3" в главном меню.

В предложенном сообщении при вводе:

1 - Из текстового файла считывается информация, являющаяяся руководством пользователя с пояснением функционала программы.

2 - Выводится на экран информация о программе и контактные данные разработчика.

0 - Осуществляется выход в главное меню.

Условия, необходимые для выполнения программы:

1. Программное обеспечение (Windows 7, Windows 8(.1), Windows 10, Linux)

Характеристики программы:

1. Средства контроля правильности выполнения программы.

1. Наличие проверки вводимой информации пользователем.

2. Наличие проверки существования файлов перед осуществления работы с ними.

2. Режим работы программы.

1. Программа осуществляет работу пока она запущена.

2. Завершение программы осуществляется выбором пункта меню номер.

Информация о названии программы и языках, используемых при разработке программы:

1.BaseOfZoo. Языки: С++.

**Заключение**

В процессе выполнения курсовой работы были приобретены навыки по созданию консольного приложения в среде программирования «Dev-C++», на языке программирования С++.

Все поставленные задачи: разработка и написание алгоритма, составление таблицы спецификаций и проведение тестирования программы — были решены.

В ходе выполнения курсовой работы были разработаны такие функции, как добавление новых данных в таблицу, их удаление. открытие, закрытие, сохранение файла, обработка запросов.

Программа исправно выполняет задания, прописанные в индивидуальном задании.

Разработанный программный продукт работает исправно и выполняет свои функции, что подтверждается результатами тестирования.

**Список использованных источников**

Рекомендуемая литература:

1. Павловская Т.А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. — СПб.: Питер, 2003 — 461 с: ил.
2. Иванов В. Б. Прикладное программирование на C/C++: с нуля до мультимедийных и сетевых приложений. М.: «СОЛОН-Пресс». 2008 – 240 с. [Электронный ресурс]. – Режим доступа: http://e.lanbook.com/books/ element.php&pl1\_id=13740
3. Хохлов Д.Г. Методы программирования на языке С: практикум Ч.1. «Бином. Лаборатория знаний». 2014–336с.[Электронный ресурс].–Режим доступа: http://e.lanbook.com/books/element.php ?pl1\_id=50555
4. Хохлов Д.Г. Методы программирования на языке С: практикум Ч.2. «Бином. Лаборатория знаний». 2014–377с.[Электронный ресурс].–Режим доступа: http://e.lanbook.com/books/element.php ?pl1\_id=50555
5. Иванов В. Б. Прикладное программирование на C/C++: с нуля до мультимедийных и сетевыхприложений. М.: «СОЛОН-Пресс». 2008 – 240 с. [Электронный ресурс]. – Режим доступа: http://e.lanbook.com/books/ element.php&pl1\_id=13740
6. Липман С., Лажойе Ж. Язык программирования С++. Полное руководство. СПб.: Невский диалект, М.: ДМК Пресс. 2006 – 1104с. [Электронный ресурс]. – Режим доступа: http://e.lanbook.com/view/book/ 1216/