МІНІСТЕРСТВО ОСВІТИ І НАУКИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ «ХПІ»

Кафедра «Обчислювальна техніка та програмування»

Розрахункове завдання з програмування

Тема: «РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ»

Пояснювальна записка

КІТ 119а

Розробники

Виконав:

студент групи КІТ-101.8б

\_\_\_\_\_\_\_\_\_**\_**\_**\_\_\_\_**/Чугунов В.Ю/

Перевірив:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Бартош М.В/

Харків 2020

ЗАТВЕРДЖЕНО  
 КІТ 119а

Розрахункове завдання з програмування

Тема: «РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ»

Пояснювальна записка

КІТ 119а

Аркушів\_\_\_

Харків 2020

АНОТАЦІЯ

Метою розрахунково-графічного завдання є набуття практичних навичок у розробці програмного забезпечення з використанням такої парадигми мови програмування як об’єктно-орієнтоване програмування. А також закріплення знань, отриманих протягом усього навчального року.

Дана робота присвячена розробці програми, яка допоможе викладачам створювати бази даних для робіт студентів. Програма надає можливість сортування та пошуку даних за різними критеріями, що дозволить набагато швидше обробляти таку базу даних.

АННОТАЦИЯ

Целью расчетно-графического задания является приобретение практических навыков в разработке программного обеспечения с использованием такой парадигмы языка программирования как объектно-ориентированное программирование. А также закрепление знаний, полученных на протяжении всего учебного года.

Данная работа посвящена разработке программы, которая поможет преподавателям создавать базы данных для работ студентов. Программа предоставляет возможность сортировки и поиска данных по разным критериям, что позволит намного быстрее обрабатывать такую базу данных.

ABSTRACT

The aim of the calculation and graphic task is to acquire practical skills in software development using a programming language paradigm like object oriented programming. As well as consolidating the knowledge gained throughout the school year.

This work is devoted to the development of a program that will help teachers create databases for student works. The program provides the ability to sort and search data according to different criteria, which will allow much faster processing of such a database.

ЗМІСТ

Вступ………………………………………………………………………………….3

1 Аналітичний огляд інформації……………….……………………………….......4

1.1 Призначення та галузь застосування………………………………….....4

1.2 Постановка завдання до роботи………………………………………….4

2 Розробка інформаційно-довідкової системи……………………………………..5

2.1 Опис об’єктів роботи……………………………………………………...5

2.2 Опис вхідних та вихідних даних………………………………………..12

Висновки ...……………………………………………………………………….....13

Список джерел інформації…………………………………………………………14

Додаток А…………………………………………………………………………...15

Додаток Б……………………………………………………………………………28

ВСТУП

Автоматизована інформаційна система – сукупність інформації, різних методів і моделей, апаратних, програмних, організаційних, технологічних засобів і відповідних фахівців. Отже, інформаційна система – це організаційно впорядкована сукупність фахівців, інформаційних ресурсів та інформаційних технологій, зокрема з використанням засобів обчислювальної техніки і зв'язку, що реалізовують такі інформаційні процеси, як отримання вхідних даних; обробка цих даних і/або зміна власного внутрішнього стану (внутрішніх зв’язків/відносин), видача результату або зміна свого зовнішнього стану (зовнішніх зв’язків/відносин).

За допомогою інформаційної системи надається можливість встановлення зв’язку між усіма елементами бізнес-процесів підприємства, що покращує можливості планування, контролю й регулювання процесів. Інформаційна система, при формуванні якої використано принцип зворотного зв’язку на всіх рівнях управління і сучасні інформаційно-комунікаційні технології, забезпечує зв’язок між елементами системи управління й елементами бізнес-процесів, тобто між усіма етапами прийняття рішень, а також надає можливість накопичення даних, аналізу і моделювання.

1 АНАЛІТИЧНИЙ ОГЛЯД ІНФОРМАЦІЇ

**1.1 Призначення та галузь застосування**

Призначення розробки – оперування даними про самостійні роботи учнів (студентів). База, що буде утворена на основі цих даних, зберігатиме усю необхідну інформацію, з метою її подальшого використання.

Програмне забезпечення надає можливість виконувати сортування даних за різними критеріями, здійснювати пошук конкретних даних, задавши при цьому ім’я потрібного учня (студента). Також користувач може власноруч додавати та видаляти дані.

Розробка може використовуватись у закладах середньої та вищої освіти. Буде користуватись популярністю серед викладачів, тому що дозволить структуровано зберігати усю необхідну інформацію.

**1.2 Постановка завдання до роботи**

Прикладна галузь «Роботи студентів». Базовий клас «Самостійна робота». Завдання для обходу: «Визначити, яку кількість робіт виконує студент за семестр». При виконанні завдання необхідно виконати наступне:

1. дослідити літературу стосовно прикладної галузі. За результатами аналізу літератури оформити перший, аналітичний розділ пояснювальної записки обсягом 2 – 3 сторінки;
2. для прикладної галузі розробити розгалужену ієрархію класів, яка складається не менш ніж з трьох класів. Класи повинні мати перевантажені оператори введення-виведення даних та порівняння;
3. розробити клас-контролер, що буде включати колекцію розроблених класів, та наступні методи роботи з цією колекцією:
   1. додавання елементів;
   2. видалення елементів;
   3. виведення даних на екран;
   4. сортування елементів за вказаним критерієм;
   5. пошук елементів;
4. розробити клас, який має відображати діалогове меню для демонстрації реалізованих функцій класу контролера;
5. оформити схеми алгоритмів функцій класів контролера та діалогового меню;
6. оформити документацію: пояснювальну записку.

2 РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ

**2.1 Опис об’єктів роботи**

Самостійна робота студентів – це такий вид навчальної діяльності, що виконується студентами з використанням розумових і (або) фізичних зусиль як під час аудиторних занять, так і в позааудиторний час, за завданням і під контролем викладача, але без особистої його участі, та спрямована на досягнення поставленої мети; головним завданням самостійної роботи є підвищення якості знань, формування пізнавальної активності, самостійності, позитивної мотивації, інтелектуальних умінь.

Самостійну роботу студентів можна класифікувати за різними критеріями:

1. З урахуванням на місце і час проведення:
   1. самостійну роботу студентів на аудиторних заняттях;
   2. позааудиторну самостійну роботу (3-4 години на день);
   3. самостійну роботу під контролем викладача;
2. За рівнем обов’язковості виокремлюють:
   1. обов’язкову, окреслену навчальними планами;
   2. бажану (участь у наукових гуртках, конференціях);
   3. добровільну (участь у конкурсах, олімпіадах);
3. З огляду на рівень прояву творчості студентів виокремлюють:
   1. репродуктивну самостійну роботу, що здійснюється за зразком;
   2. реконструктивну, яка передбачає слухання і доповнення лекції викладача, складання планів, конспектів тощо;
   3. евристичну, спрямовану на вирішення проблемних завдань, отримані нової інформації, її структурування і використання;
   4. дослідницьку, яка орієнтована на проведення наукових досліджень.

У даній роботі вирішено реалізувати два види самостійних робіт: розрахунково-графічну роботу та тест.

Для написання програми було обрано мову програмування С++. Незважаючи на те, що це досить стара мова, нею й досі активно користуються. Перевагами цієї мови є те, що виконувати роботу з пам’яттю можна на дуже низькому рівні. Також, в ній зручно користуватися об’єктно-орієнтованим програмуванням, що і потрібно для написання роботи.

Діалоговий режим забезпечує безпосередню взаємодію користувача з системою. Ініціатором діалогу може бути як користувач, так і ЕОМ. В останньому випадку на кожному кроці користувачу повідомляється, що треба робити.

Дані вирішено зберігати у STL контейнері – векторі. Причиною такого вибору є зручність у користуванні і мінімалістичність методів, бо використовувати потрібно тільки додавання у кінець та видалення елементів, немає необхідності у використанні більш складних за будовою контейнерів.

Середовищем програмування виступає Microsoft Visual Studio 2017. Це досить зручне у використанні середовище, у якому є усі необхідні для розробки програм інструменти.

Абстрактним а також базовим класом у роботі виступає клас Work (рис. 1). Його наслідують класи RGZ (рис. 2а) та Test (рис. 2б).

|  |  |
| --- | --- |
|  |  |
| а) | б) |
| Рисунок 1 – Методи та поля класу Work | |

|  |  |
| --- | --- |
|  |  |
| а) | б) |
| Рисунок 2 – Поля класу: а) RGZ; б) Test | |

Щоб виконувати операції над даними класами розроблено клас-контролер «Controller». Його елементи і поля подані на рис. 3. Також для можливості звертання до користувача розроблено клас – діалогове меню «Dialog». Його елементи та поля подані на рис. 4.

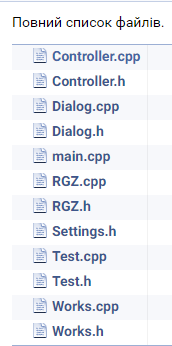
|  |  |
| --- | --- |
|  |  |
| а) | б) |
| Рисунок 3 – Клас Controller: а) методи класу; б) поля класу | |

|  |  |
| --- | --- |
|  |  |
| а) | б) |
| Рисунок 4 – Клас Dialog: а) методи класу; б) поля класу | |

Програма складається з таких модулів:

* головного, що містить функцію main;
* інтерфейсного, що містить опис класу Dialog;
* модуля, що містить опис класу Work;
* модуля, що містить опис класу RGZ;
* модуля, що містить опис класу Test;
* модуля, що містить опис класу Controller.

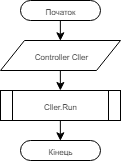
На рис. 5 подано структуру проекту розробленого програмного продукту.

  
Рисунок 5 – Структура проекту

Функція main координує роботу всіх функцій програми.

Схема алгоритму подана на рис. 6

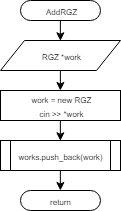
Порядок роботи: створюється об’єкт класу Controller, у об’єкта викликається метод Run.

  
Рисунок 6 – Схема алгоритму функції main

Функція AddRGZ додає у вектор нову роботу типу «RGZ».

Схема алгоритму подана на рис. 7.

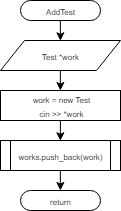
Порядок роботи: створюється новий об’єкт типу «RGZ», покажчик на цей об’єкт заноситься у вектор.

  
Рисунок 7 – Схема алгоритму функції AddRGZ

Функція AddTest додає у вектор нову роботу типу «Test».

Схема алгоритму подана на рис. 8.

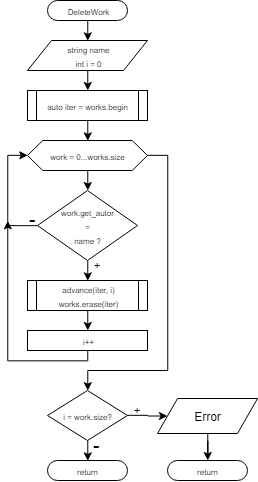
Порядок роботи: створюється новий об’єкт типу «Test», покажчик на цей об’єкт заноситься у вектор.

  
Рисунок 8 – Схема алгоритму функції AddTest

Функція DeleteWork видаляє роботу з вектору.

Схема алгоритму подана на рис. 9.

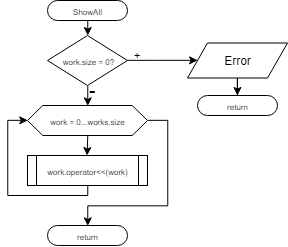
Порядок роботи: користувачу пропонується ввести ім’я студента, роботу якого буде видалено. У циклі шукається необхідна робота. Якщо роботу не було знайдено, виводиться повідомлення про відсутність даних. В іншому випадку виконується видалення об’єкту.

  
Рисунок 9 – Схема алгоритму функції DeleteWork

Функція ShowAll виводить на екран інформацію про усі роботи.

Схема алгоритму подана на рис. 10.

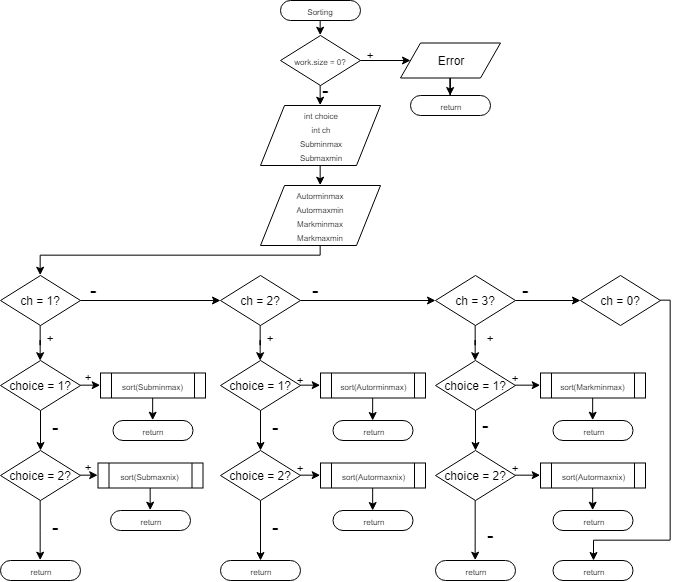
Порядок роботи: перевіряється кількість елементів вектору. Якщо кількість дорівнює нулю, виводиться повідомлення про відсутність даних. Інакше, використовуючи range-based цикл, функція виводить на екран інформацію про усі роботи.

  
Рисунок 10 – Схема алгоритму функції ShowAll

Функція Sorting призначена для сортування колекції за заданим критерієм.

Схема алгоритму подана на рис. 11.

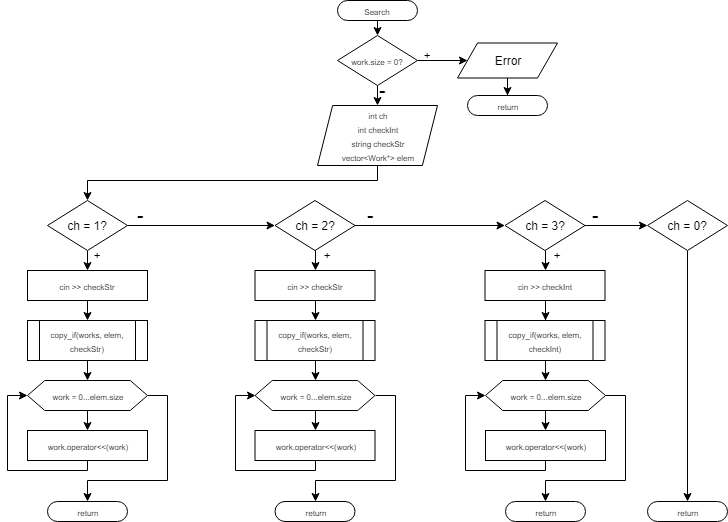
Порядок роботи: на вхід функція приймає вже вибраний критерій сортування. Сортувати можна за назвою предмета, ім’ям автора та оцінкою роботи. Далі, колекція перевіряється на відсутність елементів. Якщо кількість елементів дорівнює нулю, виводиться відповідне повідомлення. В іншому випадку створюються предикатори сортування для кожного критерію. Після чого необхідно обрати напрямок сортування. І на основі цих даних виконується сортування колекції.

  
Рисунок 11 – Схема алгоритму функції Sorting

Функція Search призначена для пошуку робіт за заданим критерієм.

Схема алгоритму подана на рис. 12.

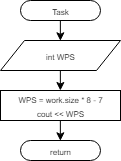
Порядок роботи: на вхід функція приймає вже вибраний критерій пошуку. Шукати можна за назвою предмета, ім’ям автора та оцінкою роботи. Далі, колекція перевіряється на відсутність елементів. Якщо кількість елементів дорівнює нулю, виводиться відповідне повідомлення. В іншому створюється проміжна колекція, в яку буде записано дані, які підходять по критерію. Далі слід ввести значення для пошуку елементів. Після чого виконується пошук та виведення на екран результату пошуку.

  
Рисунок 12 – Схема алгоритму функції Search

Функція Task рахує кількість виконаних робіт за семестр.

Схема алгоритму подана на рис. 13.

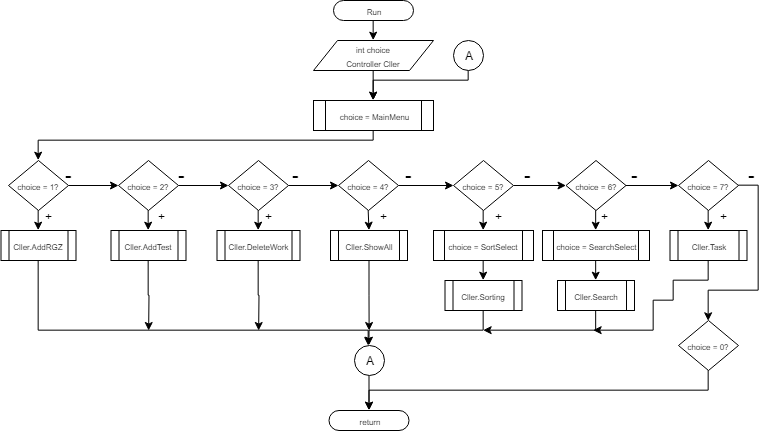
Порядок роботи: функція рахує кількість робіт за семестр по формулі [works.size \* 8 – 7]. Виводяться результати.

  
Рисунок 13 – Схема алгоритму функції Task

Функція Run запускає механізм діалогу.

Схема алгоритму подана на рис. 14.

Порядок роботи: користувачу пропонується обрати дію з меню. Після вибору виконується відповідна функція.

  
Рисунок 14 – Схема алгоритму функції Run

**2.2 Опис вхідних та вихідних даних**

Вхідні дані отримуються шляхом введення їх користувачем з клавіатури. Вводити слід у наступній послідовності:

1. назва предмета;
2. автор роботи;
3. прізвище викладача, що приймає роботу;

Якщо додається РГЗ, то

1. завдання до роботи;
2. складність завдання;
3. оцінка.

Якщо додається тест, то:

1. тема тесту;
2. номер варіанту;
3. оцінка.

Усі вихідні дані за необхідністю виводяться на екран. Це дасть змогу одразу побачити результат роботи функцій.

ВИСНОВКИ

В даній роботі було розроблено програмне забезпечення для викладачів закладів середньої та вищої освіти. Було реалізовано базу даних для самостійних робіт, яка дозволяє додавати, видаляти, сортувати інформацію, а також шукати інформацію за заданим критерієм, рахувати кількість робіт, виконаних студентом (учнем) за семестр.

СПИСОК ДЖЕРЕЛ ІНФОРМАЦІЇ

1. Вірт Н. Агоритми + структури даних = програми / Н. Вірт. – М: Мир, 1985. – 274 с.

2. Ахо В. А. Структури даних та алгоритми / А. В. Ахо, Д. Е. Хопкрофт, Д. Д. Ульман. – М: Видавничий дім «Уільямс», 2000. – 384 с.

3. Физерс М. Эффективная работа с унаследованным кодом / М. Физерс – М: Вильямс, 2009. – 393 с.

ДОДАТОК А

Текст програми

Текст файлу main.cpp

#include "Settings.h"

#include "Dialog.h"

int main()

{

setlocale(LC\_ALL, "rus");

// Создание объэкта класса Dialog

Dialog Cller;

// Выполнение метода Run

Cller.Run();

system("PAUSE");

return 0;

}

Текст файлу Dialog.h

#pragma once

#include "Controller.h"

// Класс "Диалоговое меню"

class Dialog

{

public:

/\* Конструкторы, деструктор \*/

// Конструктор по-умолчанию

Dialog();

// Деструктор

~Dialog();

// Запуск диалогового меню

void Run();

private:

/\* Описание \*/

// Переменная, виполняющая действия над иерархией классов

Controller Cller;

// Главное меню

int MainMenu();

// Выбор критерия сортировки

int SortingSelect();

// Выбор критерия поиска

int SearchSelect();

};

Текст файлу Dialog.cpp

#include "Dialog.h"

Dialog::Dialog()

{

}

Dialog::~Dialog()

{

}

void Dialog::Run()

{

int choice;

do {

choice = MainMenu();

switch (choice)

{

case 1:

Cller.AddRGZ();

system("cls");

break;

case 2:

Cller.AddTest();

system("cls");

break;

case 3:

Cller.DeleteWork();

system("cls");

break;

case 4:

Cller.ShowAll();

system("cls");

break;

case 5:

choice = SortingSelect();

Cller.Sorting(choice);

system("cls");

break;

case 6:

choice = SearchSelect();

Cller.Search(choice);

system("cls");

break;

case 7:

Cller.Task();

system("cls");

break;

case 0:

SetColor(13, 0);

cout << "\nСпасибо за работу!\n";

SetColor(7, 0);

}

} while (choice);

}

int Dialog::MainMenu()

{

int choice = 0;

Info();

cout << "Выберите один из вариантов:\n";

cout << "1. Добавить РГЗ.\n";

cout << "2. Добавить Тест.\n";

cout << "3. Удалить работу.\n";

cout << "4. Вывести все работы на экран.\n";

cout << "5. Сортировка списка.\n";

cout << "6. Поиск работ.\n";

cout << "7. Определить количество работ за семестр.\n";

cout << "0. Выход.\n";

cout << "Введите ваш ответ сюда: ";

cin >> choice;

CheckSelect(0, 7, choice);

return choice;

}

int Dialog::SortingSelect()

{

int choice = 0;

Info();

cout << "Выберите один из критериев:\n";

cout << "1. Сотрировка по предмету.\n";

cout << "2. Сотрировка по имени автора.\n";

cout << "3. Сотрировка по оценке.\n";

cout << "0. Выход.\n";

cout << "Введите ваш ответ сюда: ";

cin >> choice;

CheckSelect(0, 3, choice);

return choice;

}

int Dialog::SearchSelect()

{

int choice = 0;

Info();

cout << "Выберите один из критериев:\n";

cout << "1. Поиск работы по предмету.\n";

cout << "2. Поиск работы по имени автора.\n";

cout << "3. Поиск работы по оценке.\n";

cout << "0. Выход.\n";

cout << "Введите ваш ответ сюда: ";

cin >> choice;

CheckSelect(0, 3, choice);

return choice;

}

Текст файлу Controller.h

#pragma once

#include <algorithm>

#include <vector>

#include <Windows.h>

#include "RGZ.h"

#include "Test.h"

using std::vector;

using std::sort;

using std::copy\_if;

using std::back\_inserter;

// Класс "Контроллер"

class Controller

{

public:

/\* Конструкторы, деструктор \*/

// Конструктор по-умолчанию

Controller();

// Деструктор

~Controller();

/\* Методы \*/

// Добавление РГЗ

void AddRGZ();

// Добавление Теста

void AddTest();

// Удаление работы

void DeleteWork();

// Вывод работ на экран

void ShowAll();

// Сортировка набора

void Sorting(int ch);

// Поиск работ

void Search(int ch);

// Выполнение задания

void Task();

private:

/\* Описание \*/

// Вектор, состоящий из работ

vector<Work\*> works;

};

// Предикатор сортировки по предмету мин-макс

class SortingMinMaxSubject

{

public:

bool operator() (const Work \*a, const Work \*b) { return a->get\_subject() < b->get\_subject(); };

};

// Предикатор сортировки по предмету макс-мин

class SortingMaxMinSubject

{

public:

bool operator() (const Work \*a, const Work \*b) { return a->get\_subject() > b->get\_subject(); };

};

// Предикатор сортировки по автору мин-макс

class SortingMinMaxAutor

{

public:

bool operator() (const Work \*a, const Work \*b) { return a->get\_autor() < b->get\_autor(); };

};

// Предикатор сортировки по автору макс-мин

class SortingMaxMinAutor

{

public:

bool operator() (const Work \*a, const Work \*b) { return a->get\_autor() > b->get\_autor(); };

};

// Предикатор сортировки по оценке мин-макс

class SortingMinMaxMark

{

public:

bool operator() (const Work \*a, const Work \*b) { return a->get\_mark() < b->get\_mark(); };

};

// Предикатор сортировки по оценке макс-мин

class SortingMaxMinMark

{

public:

bool operator() (const Work \*a, const Work \*b) { return a->get\_mark() > b->get\_mark(); };

};

// Функция смены цвета консоли

void SetColor(int text, int background);

// Проверка выбора варианта

bool CheckSelect(int b, int e, int i);

// Уведомление об успешной операции

void OperationPass();

// Уведомление об отмене операции

void OperationCancel();

// Информация о работе

void Info();

Текст файлу Controller.cpp

#include "Controller.h"

Controller::Controller()

{

}

Controller::~Controller()

{

}

void Controller::AddRGZ()

{

Info();

cout << "Информация об РГЗ:\n\n";

RGZ \*work = new RGZ;

cin >> \*work;

works.push\_back(work);

OperationPass();

}

void Controller::AddTest()

{

Info();

cout << "Информация о Тесте:\n\n";

Test \*work = new Test;

cin >> \*work;

works.push\_back(work);

OperationPass();

}

void Controller::DeleteWork()

{

Info();

string name;

cout << "Удаление работы:\n\n";

cout << "Введите имя студента: ";

cin >> name;

int i = 0;

auto iter = works.begin();

for (auto &work : works)

{

if (work->get\_autor() == name)

{

std::advance(iter, i);

works.erase(iter);

}

i++;

}

if (i == works.size())

{

SetColor(12, 0);

cout << "Студент с таким именем не писал работу.\n";

SetColor(7, 0);

system("pause");

return;

}

OperationPass();

}

void Controller::ShowAll()

{

Info();

if (works.empty())

{

SetColor(12, 0);

cout << "\nНет информации. Сначала добавте что-либо.\n";

SetColor(7, 0);

system("pause");

return;

}

for (auto &work : works)

{

work->operator<<(work);

cout << endl;

}

OperationPass();

}

void Controller::Sorting(int ch)

{

Info();

if (works.empty())

{

SetColor(12, 0);

cout << "\nНет информации. Сначала добавте что-либо.\n";

SetColor(7, 0);

system("pause");

return;

}

int choice;

SortingMinMaxSubject minmaxS;

SortingMaxMinSubject maxminS;

SortingMinMaxAutor minmaxA;

SortingMaxMinAutor maxminA;

SortingMinMaxMark minmaxM;

SortingMaxMinMark maxminM;

do {

cout << "Выберите направление сортировки:\n";

cout << "1. Сортировать по возрастанию.\n";

cout << "2. Сортировать по убыванию.\n";

cout << "0. Выход.\n";

cout << "Введите ваш ответ сюда: ";

cin >> choice;

} while (!CheckSelect(0, 2, choice));

switch (ch)

{

case 1:

switch (choice)

{

case 1:

sort(works.begin(), works.end(), minmaxS);

OperationPass();

return;

case 2:

sort(works.begin(), works.end(), maxminS);

OperationPass();

return;

case 0:

OperationCancel();

return;

}

break;

case 2:

switch (choice)

{

case 1:

sort(works.begin(), works.end(), minmaxA);

OperationPass();

return;

case 2:

sort(works.begin(), works.end(), maxminA);

OperationPass();

return;

case 0:

OperationCancel();

return;

}

break;

case 3:

switch (choice)

{

case 1:

sort(works.begin(), works.end(), minmaxM);

OperationPass();

return;

case 2:

sort(works.begin(), works.end(), maxminM);

OperationPass();

return;

case 0:

OperationCancel();

return;

}

break;

case 0:

cout << endl;

OperationCancel();

}

}

void Controller::Search(int ch)

{

Info();

if (works.empty())

{

SetColor(12, 0);

cout << "\nНет информации. Сначала добавте что-либо.\n";

SetColor(7, 0);

system("pause");

return;

}

vector<Work \*> elements;

string checkStr;

int checkInt;

cout << "\nВведите искомое значение значение: ";

switch (ch)

{

case 1:

cin >> checkStr;

copy\_if(works.begin(), works.end(), back\_inserter(elements), [&checkStr](Work \*a) {

return a->get\_subject() == checkStr;

});

for (auto &work : elements)

{

work->operator<<(work);

cout << endl;

}

OperationPass();

break;

case 2:

cin >> checkStr;

copy\_if(works.begin(), works.end(), back\_inserter(elements), [&checkStr](Work \*a) {

return a->get\_autor() == checkStr;

});

for (auto &work : elements)

{

work->operator<<(work);

cout << endl;

}

OperationPass();

break;

case 3:

cin >> checkInt;

copy\_if(works.begin(), works.end(), back\_inserter(elements), [&checkInt](Work \*a) {

return a->get\_mark() == checkInt;

});

for (auto &work : elements)

{

work->operator<<(work);

cout << endl;

}

OperationPass();

break;

case 0:

cout << endl;

OperationCancel();

}

}

void Controller::Task()

{

Info();

int worksPerSemester = works.size() \* 8 - 7;

cout << "\nСтудент выполнил " << worksPerSemester << " раюот за семестр.\n";

OperationPass();

}

bool CheckSelect(int b, int e, int i)

{

if (i < b || i > e)

{

SetColor(12, 0);

cout << "\nВы ошиблись. Введите число от " << b << " до " << e << endl;

SetColor(7, 0);

system("PAUSE");

system("cls");

return false;

}

return true;

}

void OperationPass()

{

SetColor(10, 0);

cout << "Операция прошла успешно!\n";

SetColor(7, 0);

system("PAUSE");

system("cls");

}

void OperationCancel()

{

SetColor(12, 0);

cout << "Операция отменена.\n";

SetColor(7, 0);

system("PAUSE");

system("cls");

}

void Info()

{

system("cls");

SetColor(9, 0);

cout << " РГЗ по дисциплине \"Программрование\" \*Выполнил: Пумня Александр\n";

SetColor(7, 0);

}

void SetColor(int text, int background)

{

HANDLE hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hStdOut, (WORD)((background << 4) | text));

}

Текст файлу Works.h

#pragma once

#include <iostream>

#include <string>

#include <sstream>

using std::string;

using std::ostream;

using std::istream;

using std::stringstream;

using std::cout;

using std::cin;

using std::endl;

// Базовый абстрактный класс "Работа студента"

class Work

{

public:

/\* Конструкторы, деструктор \*/

// Конструктор по-умолчанию

Work();

// Конструктор с параметрами

Work(string sub, string a, string tName, int m);

// Деструктор

~Work();

/\* Получене доступа \*/

// Получение ID

inline const int& get\_id() const { return this->id; };

// Получение типа

inline const string& get\_type() const { return this->type; };

/\* Геттеры \*/

// Получение названия предмета

inline const string& get\_subject() const { return this->subjec; };

// Получение имени автора

inline const string& get\_autor() const { return this->author; };

// Получение фамилии преподователя

inline const string& get\_teacherName() const { return this->teachName; };

// Получение оценки

inline const int& get\_mark() const { return this->mark; };

/\* Виртуальные методы \*/

// Получение темы теста

virtual const string& get\_theme() const = 0;

// Получение заданий РГЗ

virtual const string& get\_task() const = 0;

// Получение варианта теста

virtual const int& get\_variant() const = 0;

// Получение сложности РГЗ

virtual const int& get\_diff() const = 0;

/\* Сеттеры \*/

// Сеттер для названия предмета

inline const void set\_subject(const string &s) { this->subjec = s; };

// Сеттер для имени автора

inline const void set\_autor(const string &a) { this->author = a; };

// Сеттер для фамилии преподователя

inline const void set\_teacherName(const string &t) { this->teachName = t; };

// Сеттер для оценки

inline const void set\_mark(const int &m) { this->mark = m; };

// Виртуальное переопределение оператора вывода

virtual ostream& operator<< (const Work\* work) = 0;

protected:

/\* Описание \*/

// Название предмета

string subjec;

// Имя автора

string author;

// Фамилия преподователя

string teachName;

// Оценка

int mark;

/\* Идентификаторы \*/

// Идентификатор типа

string type;

// ID

int id;

// Генерация ИД

static int id\_gen;

};

Текст файлу Works.cpp

#include "Works.h"

Work::Work()

{

this->subjec = '\0';

this->author = '\0';

this->teachName = '\0';

this->mark = 0;

this->id\_gen++;

}

Work::Work(string sub, string a, string tName, int m)

{

this->subjec = sub;

this->author = a;

this->mark = m;

this->id\_gen++;

}

Work::~Work()

{

}

Текст файлу RGZ.h

#pragma once

#include "Works.h"

// Класс "РГЗ" - наследник класса "Работа студента"

class RGZ : public Work

{

public:

/\* Конструкторы, деструктор \*/

// Конструктор по-умолчанию

RGZ();

// Конмтруктор с параметрами

RGZ(string sub, string a, string tName, int m, string task, int diff);

// Деструктор

~RGZ();

/\* Получение доступа \*/

/\* Геттеры \*/

// Переопределённое получение задания РГЗ

inline const string& get\_task() const override { return this->task; };

// Переопределённое получение сложности РГЗ

inline const int& get\_diff() const override { return this->difficulty; };

// Переопределённое получение темы теста

inline const string& get\_theme() const override { return ""; };

// Переопределённое получение варианта студента

inline const int& get\_variant() const override { return 0; };

/\* Сеттеры \*/

// Сеттер для задания РГЗ

inline const void set\_task(const string &t) { this->task = t; };

// Сеттер для сложносты РГЗ

inline const void set\_diff(const int &d) { this->difficulty = d; };

/\* Операторы \*/

// Переопределение оператора "<"

const bool& operator< (const RGZ& obj) const;

// Переопределение оператора ">"

const bool& operator> (const RGZ& obj) const;

// Переопределение оператора "=="

const bool& operator== (const RGZ& obj) const;

// Переопределение оператора "<="

const bool& operator<= (const RGZ& obj) const;

// Переопределение оператора ">="

const bool& operator>= (const RGZ& obj) const;

// Переопределение оператора "<<"

ostream& operator<< (const Work\* obj) override;

// Переопределение оператора ">>"

friend istream& operator>> (istream& is, RGZ& obj);

private:

/\* Описание \*/

// Задание РГЗ

string task;

// Сложносит РГЗ

int difficulty;

/\* Идентификаторы \*/

// Идентификатор типа

static string Type;

};

/\* Операторы \*/

// Переопределение оператора >>

istream& operator>> (istream& is, RGZ& obj);

Текст файлу RGZ.cpp

#include "RGZ.h"

// Конструктор по-умолчанию

RGZ::RGZ() : Work()

{

this->type = this->Type;

this->id = this->id\_gen;

this->task = '\0';

this->difficulty = 0;

}

// Конструктор с параметрами

RGZ::RGZ(string sub, string a, string tName, int m, string task, int diff)

: Work(sub, a, tName, m)

{

this->type = this->Type;

this->id = this->id\_gen;

this->task = task;

this->difficulty = diff;

}

// Деструктор

RGZ::~RGZ()

{

}

// Перегрузка оператора "<"

const bool & RGZ::operator<(const RGZ & obj) const

{

bool expr = this->author < obj.get\_autor()

&& this->subjec < obj.get\_subject()

&& this->teachName < obj.get\_teacherName()

&& this->mark < obj.get\_mark()

&& this->task < obj.get\_task()

&& this->difficulty < obj.get\_diff();

return expr;

}

// Перегрузка оператора ">"

const bool & RGZ::operator>(const RGZ & obj) const

{

bool expr = this->author > obj.get\_autor()

&& this->subjec > obj.get\_subject()

&& this->teachName > obj.get\_teacherName()

&& this->mark > obj.get\_mark()

&& this->task > obj.get\_task()

&& this->difficulty > obj.get\_diff();

return expr;

}

// Перегрузка оператора "=="

const bool & RGZ::operator==(const RGZ & obj) const

{

bool expr = this->author == obj.get\_autor()

&& this->subjec == obj.get\_subject()

&& this->teachName == obj.get\_teacherName()

&& this->mark == obj.get\_mark()

&& this->task == obj.get\_task()

&& this->difficulty == obj.get\_diff();

return expr;

}

// Перегрузка оператора "<="

const bool & RGZ::operator<=(const RGZ & obj) const

{

bool expr = this->author <= obj.get\_autor()

&& this->subjec <= obj.get\_subject()

&& this->teachName <= obj.get\_teacherName()

&& this->mark <= obj.get\_mark()

&& this->task <= obj.get\_task()

&& this->difficulty <= obj.get\_diff();

return expr;

}

// Перегрузка оператора ">="

const bool & RGZ::operator>=(const RGZ & obj) const

{

bool expr = this->author >= obj.get\_autor()

&& this->subjec >= obj.get\_subject()

&& this->teachName >= obj.get\_teacherName()

&& this->mark >= obj.get\_mark()

&& this->task >= obj.get\_task()

&& this->difficulty >= obj.get\_diff();

return expr;

}

// Перегрузка оператора "<<"

ostream & RGZ::operator<<(const Work \* obj)

{

cout << "ID: " << obj->get\_id() << endl

<< "Type: " << obj->get\_type() << endl

<< "Предмет: " << obj->get\_subject() << endl

<< "Автор: " << obj->get\_autor() << endl

<< "Преподователь: " << obj->get\_teacherName() << endl

<< "Задание: " << obj->get\_task() << endl

<< "Сложность: " << obj->get\_diff() << endl

<< "Оценка: " << obj->get\_mark() << endl;

return cout;

}

// Перегрузка оператора ">>"

istream & operator>>(istream& is, RGZ & obj)

{

cin.ignore(100500, '\n');

cout << "Введите название предмета: ";

std::getline(is, obj.subjec);

cout << "Введите имя автора работы: ";

std::getline(is, obj.author);

cout << "Введите фамилию преподователя: ";

std::getline(is, obj.teachName);

cout << "Введите задание работы: ";

std::getline(is, obj.task);

cout << "Введите сложность задания: ";

is >> obj.difficulty;

cout << "Введите оценку: ";

is >> obj.mark;

return is;

}

Текст файлу Test.h

#pragma once

#include "Works.h"

// Класс "Тест" - наследник класса "Работа студента"

class Test : public Work

{

public:

/\* Конструкторы, деструктор \*/

// Конструктор по-умолчанию

Test();

// Конструктор с параметрами

Test(string sub, string a, string tName, int m, string theme, int var);

// Деструктор

~Test();

/\* Получение доступа \*/

/\* Геттеры \*/

// Переопределённое получение темы теста

inline const string& get\_theme() const override { return this->theme; };

// Переопределённое получение варианта студента

inline const int& get\_variant() const override { return this->variant; };

// Переопределённое получение задания РГЗ

inline const string& get\_task() const override { return ""; };

// Переопределённое получение сложности РГЗ

inline const int& get\_diff() const override { return 0; };

/\* Сеттеры \*/

// Сеттер для темы теста

inline const void set\_theme(const string &th) { this->theme = th; };

// Сеттер для варианта

inline const void set\_variant(const int &var) { this->variant = var; };

/\* Операторы \*/

// Переопределение оператора ">"

const bool& operator< (const Test& obj) const;

// Переопределение оператора "<"

const bool& operator> (const Test& obj) const;

// Переопределение оператора "=="

const bool& operator== (const Test& obj) const;

// Переопределение оператора "<="

const bool& operator<= (const Test& obj) const;

// Переопределение оператора ">="

const bool& operator>= (const Test& obj) const;

// Переопределение оператора "<<"

ostream& operator<< (const Work\* obj) override;

// Переопределение оператора ">>"

friend istream& operator>> (istream& is, Test& obj);

private:

/\* Описание \*/

// Тема теста

string theme;

// Вариант теста

int variant;

/\* Идентификаторы \*/

// Идентификатор типа

static string Type;

};

/\* Операторы \*/

// Переопределение оператора ">>"

istream& operator>> (istream& is, Test& obj);

Текст файлу Test.cpp

#include "Test.h"

// Конструктор по-умолчанию

Test::Test() : Work()

{

this->type = this->Type;

this->id = this->id\_gen;

this->theme = '\0';

this->variant = 0;

}

// Конструктор с параметрами

Test::Test(string sub, string a, string tName, int m, string theme, int var)

: Work (sub, a, tName, m)

{

this->type = this->Type;

this->id = this->id\_gen;

this->theme = theme;

this->variant = var;

}

// Деструктор

Test::~Test()

{

}

// Перегрузка оператора "<"

const const bool & Test::operator<(const Test & obj) const

{

bool expr = this->author < obj.get\_autor()

&& this->subjec < obj.get\_subject()

&& this->teachName < obj.get\_teacherName()

&& this->mark < obj.get\_mark()

&& this->theme < obj.get\_theme()

&& this->variant < obj.get\_variant();

return expr;

}

// Перегрузка оператора ">"

const bool & Test::operator>(const Test & obj) const

{

bool expr = !(Test::operator<(obj));

return expr;

}

// Перегрузка оператора "=="

const bool & Test::operator==(const Test & obj) const

{

bool expr = this->author == obj.get\_autor()

&& this->subjec == obj.get\_subject()

&& this->teachName == obj.get\_teacherName()

&& this->mark == obj.get\_mark()

&& this->theme == obj.get\_theme()

&& this->variant == obj.get\_variant();

return expr;

}

// Перегрузка оператора "<="

const bool & Test::operator<=(const Test & obj) const

{

bool expr = this->author <= obj.get\_autor()

&& this->subjec <= obj.get\_subject()

&& this->teachName <= obj.get\_teacherName()

&& this->mark <= obj.get\_mark()

&& this->theme <= obj.get\_theme()

&& this->variant <= obj.get\_variant();

return expr;

}

// Перегрузка оператора ">="

const bool & Test::operator>=(const Test & obj) const

{

bool expr = this->author >= obj.get\_autor()

&& this->subjec >= obj.get\_subject()

&& this->teachName >= obj.get\_teacherName()

&& this->mark >= obj.get\_mark()

&& this->theme >= obj.get\_theme()

&& this->variant >= obj.get\_variant();

return expr;

}

// Перегрузка оператора "<<"

ostream & Test::operator<<(const Work\* obj)

{

cout << "ID: " << obj->get\_id() << endl

<< "Type: " << obj->get\_type() << endl

<< "Предмет: " << obj->get\_subject() << endl

<< "Автор: " << obj->get\_autor() << endl

<< "Преподователь: " << obj->get\_teacherName() << endl

<< "Тема: " << obj->get\_theme() << endl

<< "Вариант: " << obj->get\_variant() << endl

<< "Оценка: " << obj->get\_mark() << endl;

return cout;

}

// Перегрузка оператора ">>"

istream & operator>>(istream & is, Test & obj)

{

cin.ignore(10, '\n');

cout << "Введите название предмета: ";

std::getline(is, obj.subjec);

cout << "Введите имя автора работы: ";

std::getline(is, obj.author);

cout << "Введите фамилию преподователя: ";

std::getline(is, obj.teachName);

cout << "Введите тему теста: ";

std::getline(is, obj.theme);

cout << "Введите номер варианта: ";

is >> obj.variant;

cout << "Введите оценку: ";

is >> obj.mark;

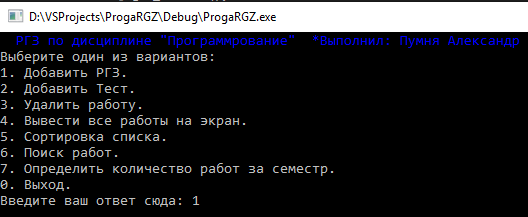
return is;

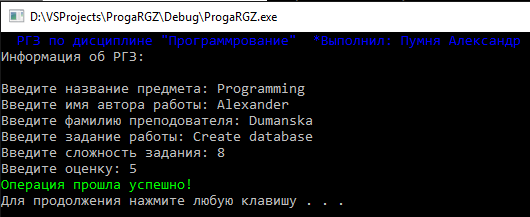
}

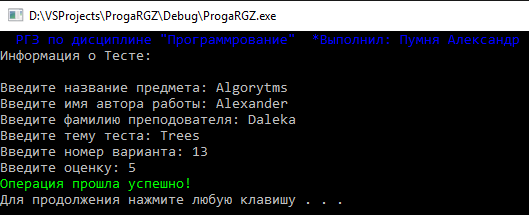
ДОДАТОК Б

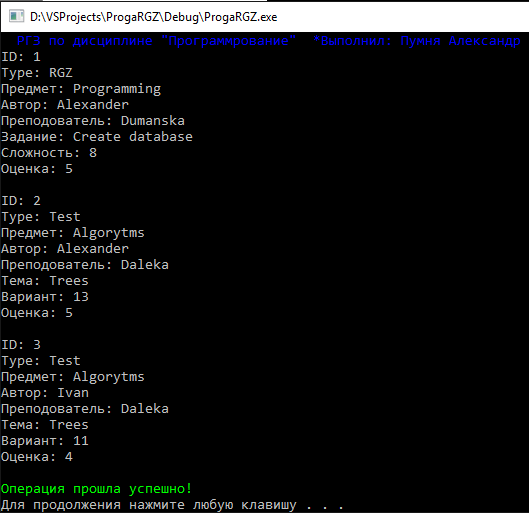
Результати роботи програми

Далі на рисунках 15 – 22 подано результати роботи функцій та методів, розроблених у програмі.

  
Рисунок 15 – Результат виконання функцій MainMenu та Run

  
Рисунок 16 – Результат виконання функції AddRGZ

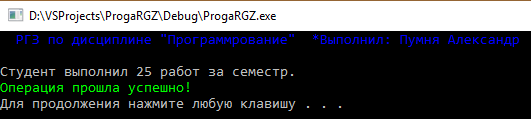
  
Рисунок 17 – Результат виконання функції AddTest

  
Рисунок 18 – Результат виконання функції ShowAll

|  |  |
| --- | --- |
|  |  |
| а) | б) |
| Рисунок 19 – Результат виконання функції DeleteWork | |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| а) | б) |
|  |  | в) |
| Рисунок 20 – Результат виконання функції Sorting | | |

|  |  |
| --- | --- |
|  |  |
| а) | б) |
| Рисунок 21 – Результат виконання функції Search | |

  
Рисунок 22 – Результат виконання функції Task