

МІНІСТЕРСТВО ОСВІТИ І НАУКИ  
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ “ХПІ”

Кафедра “Обчислювальна техніка та програмування”

Розрахункове завдання з програмування

Тема: «РОЗРОБКА ІНФОРМАЦІЙНО-ДОВІДКОВОЇ СИСТЕМИ»

Пояснювальна записка  
1КІТ.102.8А. 18046-01 81 01-1 –А3

Розробник  
Виконав:

студент групи 1.КІТ-102.8А  
\_\_\_\_\_ / Кабак О.Р./

Перевірив:  
\_\_\_\_\_ /Старший викладач. Молчанов Г.І./

Харків 2019

ЗАТВЕРДЖЕНО  
1.КІТ102.8А.18046-01 81 01-1 –А3

Розрахункове завдання з дисципліни  
«Алгоритми та структури даних»

Пояснювальна записка  
1KIT.102.8A.18046-01 81 01-1 -АЗ

Листів 20

Харків 2019  
РОЗРАХУНКОВОГО ЗАВДАННЯ З ДИСЦИПЛІНИ  
«ПРОГРАМУВАННЯ»  
*Тема роботи.* Розробка інформаційно-довідкової системи.

*Мета роботи.* Закріпити отримані знання з дисципліни «Програмування» шляхом виконання типового комплексного завдання.

## 1 ВИМОГИ

### 1.1 Розробник

- Кабак Олександр Русланович;
- Студент групи КІТ 102.8(а);
- 31-05-2019р.

### 1.2 Загальне завдання

*Завдання до роботи:*

Кожний студент отримує індивідуальне завдання. Варіант завдання обирається за номером прізвища студента у журналі групи. При виконанні завдання з розробки інформаційно-довідкової системи необхідно виконати наступне:

- 1) з табл. 1, відповідно до варіанта завдання, обрати прикладну галузь;
- 2) дослідити літературу стосовно прикладної галузі. За результатами аналізу літератури оформити перший, аналітичний розділ пояснювальної записки обсягом 2–3 сторінки;
- 3) для прикладної галузі розробити розгалужену ієрархію класів, яка складається з не менш ніж трьох класів, один з яких є «батьком» для інших (класів-спадкоємців). Класи повинні мати перевантажені оператори введення-виведення даних та порівняння;
- 4) розробити клас-контролер, що буде включати колекцію розроблених класів, та наступні методи роботи з цією колекцією:
  - а) читання даних з файлу та їх запис у контейнер;
  - б) запис даних з контейнера у файл;
  - в) сортування елементів у контейнері за вказаними критеріями: поле та напрям сортування, які задаються користувачем з клавіатури;
  - г) пошук елементів за вказаним критерієм (див. «Завдання для обходу колекції» в табл. 1);
- 5) розробити клас, який має відображати діалогове меню для демонстрації реалізованих функцій класу контролера;
- 6) оформити схеми алгоритмів функцій класів контролера та діалогового меню;
- 7) оформити документацію: пояснювальну записку (див. розділ 2 даних методичних вказівок).

*Увага.* Текст програми та результати роботи програми мають бути подані в додатках.

*Вимоги:*

- усі класи повинні мати конструктори та деструктори;
- якщо функція не змінює поля класу, вона має бути декларована як константна;
- рядки повинні бути типу string;
- при перевантаженні функції треба використовувати ключове слово `override`;
- програмний код усіх класів має бути 100 % `doxygen` документований;
- у звіті текст програми слід оформляти стилем Courier new 8 пт, інтервал – одиничний; довжина рядка не повинна перевищувати 80 символів.

*Додаткові вимоги на оцінку «добре»:*

- виконання основного завдання та додаткових наступних вимог:
- додати обробку помилок; при цьому функція, що генерує виключення, при її декларуванні повинна мати ключове слово `throw`;
- виконати перевірку вхідних даних за допомогою регулярних виразів.

*Додаткові вимоги на оцінку «відмінно»:*

- виконати завдання відповідно до вимог на оцінку «добре» та додаткові наступні вимоги:
- критерій для пошуку та сортування задавати у вигляді функтора;
- розробити клас-тестер, основною метою якого буде перевірка коректності роботи класу-контролера.

Індивідуальне завдання:

Варіант: 6

Прикладна галузь: Самостійні роботи студентів

Базовий клас: Розрахунково-Графічне Завдання (РГЗ)

Завдання для обходу колекції: Визначити кількість РГЗ, що виконує студент за весь період навчання в інституті відповідно до навчального плану.

## 2 ОПИС ПРОГРАМИ

### 2.1 Функціональне призначення

Програма призначена для виконання комплексних задач з курсу програмування

### 2.2 Опис логічної структури

Нижче продемонстрована діаграми класів (див. рис. 2.2.1)

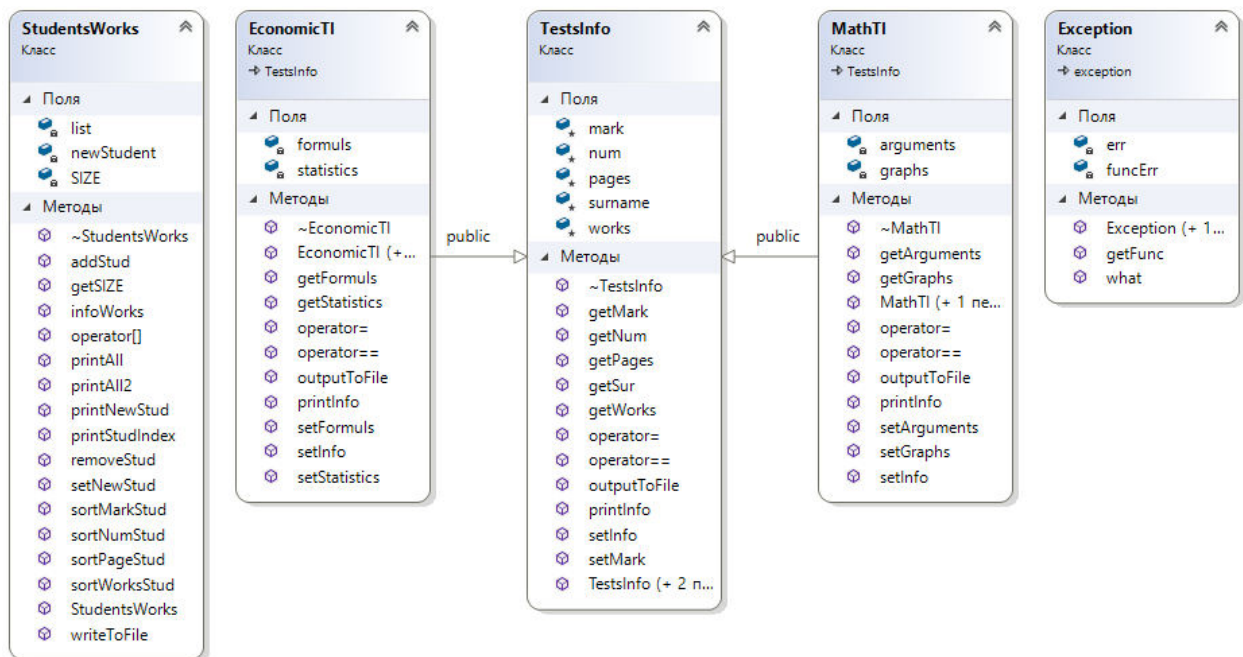


Рис.2.2.1 - Діаграма класів

Усі пояснення див. у документації.

## 2.3. Документація

### 1)Базовий клас TestsInfo:

- mark - зберігає оцінку учня;
- num - зберігає номер студента у групі;
- pages - зберігає середню кількість сторінок у роботах учня;
- surname - зберігає прізвище(а також може містити ім'я та індекс);
- works - зберігає кількість зданих робіт студента;

### 2)Спадкоємець класу TestsInfo - MathTI(містить додаткову інформацію):

- graphs - зберігає к-сть графіків у роботі(при 5 графіках оцінка 5);
- arguments -зберігає к-сть вагомих аргументів(при 3 оцінка 5);

### 3)Спадкоємець класу TestsInfo -EconomicTI(містить додаткові інформацію);

- formuls - зберігає к-сть важких економічних формул у роботі;
- statistics - зберігає к-сть статистик наведених у роботі;

### 4)Клас-масив StudentsWorks котрий містить у собі динамічний масив об'єктів класів спадкоємців TestsInfo:

- list - список студентів;
- newStudent - зрозуміло;
- SIZE - розмір групи;

Всю іншу інформацію можна знайти у документації Doxu у коді програми

## 3 ВАРІАНТИ ВИКОРИСТАННЯ

```

Current student journal:

Student number:1
Number of completed works:8
Average volume of work:4
Number of statistics in work:3
Number of strong formuls:2
GPA:5
Surname :Zaycev

Student number:2
Number of completed works:10
Average volume of work:5
Number of graphs in work:4
Number of strong arguments:3
GPA:5
Surname :Volkov Ivan

Student number:3
Number of completed works:7
Average volume of work:3
Number of graphs in work:3
Number of strong arguments:2
GPA:5
Surname :Volkov Ivan 2

```

Рис.3.1 - Запуск програми, читання даних з файлу ,створення списку з 3 студентів, 1 економіст і 2 математика

StartInfo — Блокнот					
Файл	Правка	Формат	Вид	Справка	
0# Zaycev	8	4	3	2	
1# Volkov Ivan	10	5	4	3	
1# Volkov Ivan 2	7	3	3	2	

Рис.3.2 - Вихідні дані у файлі

```
C:\Users\Greed\source\repos\RGZ\Debug\RGZ.exe

Menu:
1.Check your list.
2.Check new student information.
3.Check student information by index
4.Add a new student to the list.
5.Remove student from the list by number.
6.Change new student.
7.Number of completed student work.
8.Surnames and names of students
9.Sort student list by grade.
10.Write current list to file
11.EXIT(press 0).
You choose:2

Student number:4
Number of completed works:10
Average volume of work:5
Number of statistics in work:5
Number of strong formuls:3
GPA:5
Surname :Volkova Ira
```

Рис.3.3 - Перегляд інформації про нового студента

```
Select student index:3

Student number:3
Number of completed works:7
Average volume of work:3
Number of graphs in work:3
Number of strong arguments:2
GPA:5
Surname :Volkov Ivan 2
```

Рис.3.4 - Отримання інформації про студента по індексу

```
Select student index:1

Result:
The number of finished student works is 7/10
```

Рис.3.5 - Отримання інформації про виконані завдання студента за індексом



```
Choose a place in the journal for a new student:2
Result:
Current student journal:

Student number:1
Number of completed works:8
Average volume of work:4
Number of statistics in work:3
Number of strong formulae:2
GPA:5
Surname :Zaycev

Student number:4
Number of completed works:10
Average volume of work:5
Number of statistics in work:5
Number of strong formulae:3
GPA:5
Surname :Volkova Ira

Student number:2
Number of completed works:10
Average volume of work:5
Number of graphs in work:4
Number of strong arguments:3
GPA:5
Surname :Volkov Ivan

Student number:3
Number of completed works:7
Average volume of work:3
Number of graphs in work:3
Number of strong arguments:2
GPA:5
Surname :Volkov Ivan 2
```

Рис.3.6 - Додавання цього елементу

```

Select the student number of which you want to remove from the list 3
Result:
Current student journal:

Student number:1
Number of completed works:8
Average volume of work:4
Number of statistics in work:3
Number of strong formuls:2
GPA:5
Surname :Zaycev

Student number:4
Number of completed works:10
Average volume of work:5
Number of statistics in work:5
Number of strong formuls:3
GPA:5
Surname :Volkova Ira

Student number:3
Number of completed works:7
Average volume of work:3
Number of graphs in work:3
Number of strong arguments:2
GPA:5
Surname :Volkov Ivan 2

```

Рис3.7 Видалення 3-ого студента з заданого списку

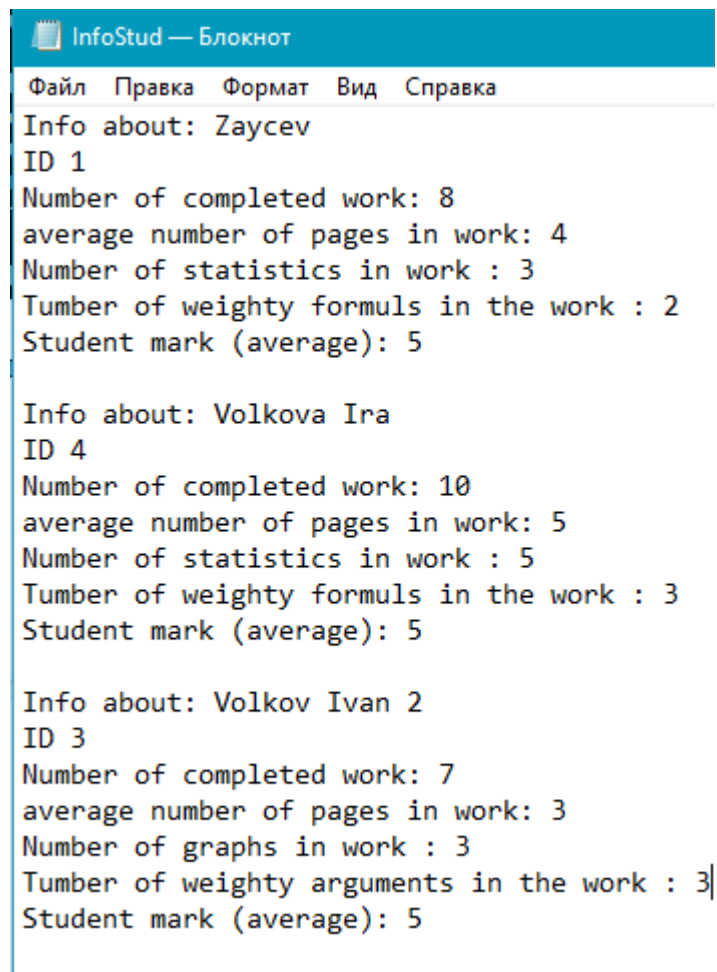
```

Do you want another new student ?
Ok it's him:
Enter student's last name:Kabak
Enter the number of student works(0-10):10
Enter the number of pages in works(1-5):4
From which group is the student?(Math-0,Economic-1)
0
Enter the number of graphs (1-5):5
Enter the number of arguments(1-3):2

Student number:4
Number of completed works:10
Average volume of work:4
Number of graphs in work:5
Number of strong arguments:2
GPA:5
Surname :Kabak

```

Рис.3.8 Заповнення інформації про нового студента

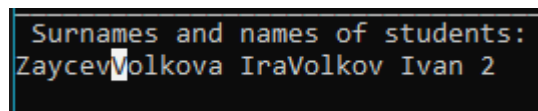


```
InfoStud — Блокнот
Файл  Правка  Формат  Вид  Справка
Info about: Zaycev
ID 1
Number of completed work: 8
average number of pages in work: 4
Number of statistics in work : 3
Tumber of weighty formuls in the work : 2
Student mark (average): 5

Info about: Volkova Ira
ID 4
Number of completed work: 10
average number of pages in work: 5
Number of statistics in work : 5
Tumber of weighty formuls in the work : 3
Student mark (average): 5

Info about: Volkov Ivan 2
ID 3
Number of completed work: 7
average number of pages in work: 3
Number of graphs in work : 3
Tumber of weighty arguments in the work : 3
Student mark (average): 5
```

Рис.3.9 - Результат виводу поточного списку у файл



```
Surnames and names of students:
ZaycevVolkova IraVolkov Ivan 2
```

Рисунок.3.10 - Результат пошуку 2 слів в полі surname

## ВИСНОВОК

В ході виконання поставленої задачі були закріплені отримані знання з дисципліни «Програмування» шляхом виконання типового комплексного завдання.

Приклад тексту програми

main.cpp

```

/**
 * @file main.cpp
 * File assignment          | Main function          + menu that works with a class object
 StudentsWorks
 * @author                  | Kabak A. R.
 * @version 1.0
 * @date 2019.05.31
 */

#include "StudentsWorks.h"

bool sort(int a, int b) {
    return a < b;
}
bool sort2(int a, int b) {
    return a > b;
}

int main(){
    try {
        int N, work=1;
        stringstream ss;
        string s;
        bool(*p)(int a, int b);

        cout << "Enter the number of students:";
        cin >> s;
        ss << s;
        ss >> N;

        if (N <= 0) {
            cout << "You entered incorrect number of students";
        }
        else {
            StudentsWorks Students(s);
            cout << "\nYour list\n";
            Students.printAll();
            getchar(); getchar();

            do{
                system("cls");
                cout <<
                "
                cout << "Menu:\n1.Check your list.\n2.Check new student information.\n3.Check
student information by index\n4.Add a new student to the list.\n";
                cout << "5.Remove student from the list by number.\n6.Change new
student.\n7.Number of completed student work.\n8.Surnames and names of students\n";
                cout << "9.Sort student list by grade.\n10.Write current list to
file\n11.EXIT(press 0).\nYou choose:";
                cin >> N;
                cout <<
                "

                if (N < 0 || N>10) {
                    cout << "You have chosen a nonexistent method";
                    getchar(); getchar();
                }
                else {
                    switch (N) {
                        case 1:
                            Students.printAll();
                            break;

                        case 2:
                            Students.printNewStud();
                            break;

                        case 3:
                            cout << " Select student index:";
                            cin >> N;
                            Students.printStudIndex(N);
                            break;

                        case 4:

```

```

        cout << " Choose a place in the journal for a new student:";
        cin >> N;
        Students.addStud(N);
        cout << "\nResult:\n";
        Students.printAll();
        break;

    case 5:
        cout << " Select the student number of which you want to remove from the list";
        cin >> N;
        Students.removeStud(N - 1);
        cout << "\nResult:\n";
        Students.printAll();

        break;

    case 6:
        cout << " Do you want another new student ?\nOk it's him:\n ";
        cout << "Enter student's last name:";
        cin >> s;
        Students.setNewStud(s);
        Students.printNewStud();
        break;

    case 7:
        cout << " Select student index:";
        cin >> N;
        cout << "\nResult:\n";
        Students.infoWorks(N + 1);
        break;

    case 8:
        cout << " Surnames and names of students:\n";
        Students.printAll2();
        break;

    case 9:
        cout << " How to sort? Ascending (0) or descending (1)?\n";
        cin >> N;
        cout << endl;
        if (N == 1) {
            p = sort;
        }
        else if (N == 0) {
            p = sort2;
        }
        else {
            cout << "You enter false variant" << endl;
            break;
        }

        Students.sortMarkStud(p);
        break;

    case 10:
        Students.writeToFile();
        cout << "Completed\n";
        break;

    case 0:
        cout << "Work with the list is over, thanks for the work!";
        work = 0;
        break;

    }
    getchar(); getchar();
}
getchar();
}while (work);
}
}
catch (Exception& exception) {
    cout << "An error has occurred in working." << exception.what() << endl << " Error in
this function: " << exception.getFunc() << endl;
}
catch (exception& exception) {
    cout << "An error has occurred in working." << exception.what() << endl;
}

```

```

}
catch (...) {
    cout << "Unknown error!" << endl;
}

_CrtSetReportMode(_CRT_WARN, _CRTDBG_MODE_FILE);
_CrtSetReportFile(_CRT_WARN, _CRTDBG_FILE_STDERR);
_CrtSetReportMode(_CRT_ERROR, _CRTDBG_MODE_FILE);
_CrtSetReportFile(_CRT_ERROR, _CRTDBG_FILE_STDERR);
_CrtSetReportMode(_CRT_ASSERT, _CRTDBG_MODE_FILE);
_CrtSetReportFile(_CRT_ASSERT, _CRTDBG_FILE_STDERR);
_CrtDumpMemoryLeaks();

return _CrtDumpMemoryLeaks();
system("pause");

return 0;
}

```

## StudentsWorks.h

```

/**
 * @file StudentsWorks.h
 * File assignment | Header file with StudentsWorks class
 * @author | Kabak A. R.
 * @version 1.0
 * @date 2019.05.31
 */

#include "EconomicTI.h"
#include "MathTI.h"
#include "Exception.h"

/**
 * Class stores information about the list of students, a temporary student who wants to join the
 * group and the size of the current group
 */
class StudentsWorks
{
private:
    int SIZE; // Group size
    TestsInfo *newStudent; // New Student
    TestsInfo **list; // Group

public:
    /**
     * Constructor with parameters.
     * Used initialization lists.
     * @param s initializes StudentsWorks::SIZE.
     */
    StudentsWorks(string s);
    /**
     * InfoIndependentsWork class destructor.
     */
    ~StudentsWorks();

    /**
     * Class overload methods
     */
    friend ostream& operator<< (ostream& out, const StudentsWorks& o);
    friend istream& operator>> (istream& in, StudentsWorks& o);
    TestsInfo& operator[](int index);

    /**
     * Method that fills in information about a new student.
     */
    void setNewStud(string s);

    /**
     * Getter SIZE.
     */
    int getSIZE();

```

```

void writeToFile();

void infoWorks(const int num);

/**
 * Methods add/remove students in/from list.
 */
void addStud(const int num);
void removeStud(const int num);

/**
 * Sorting methods
 */
void sortMarkStud (bool(*comp)(int x, int y));
void sortNumStud (bool(*comp)(int x, int y));
void sortWorksStud(bool(*comp)(int x, int y));
void sortPageStud (bool(*comp)(int x, int y));

/**
 * Methods to display information on the screen.
 */
void printStudIndex(const int num)const; //Print info about student by ID
void printNewStud();
void printAll()const;
void printAll2()const; //display students who have 2 words in the
*surname* field
};

```

## StudentsWorks.cpp

```

/**
 * @file StudentsWorks.cpp
 * File assignment | Class methods
 * @author | Kabak A. R.
 * @version 1.0
 * @date 2019.05.31
 */
#include "StudentsWorks.h"

StudentsWorks::StudentsWorks(string s) :SIZE(0), newStudent(nullptr), list(nullptr) {
    int tmpSIZE;
    stringstream ss;
    ss << s;
    ss >> tmpSIZE;

    ifstream fin("G:\\Visual Studio (SAVES)\\Lab2.4\\Lab2.4\\TEST.txt");

    regex reg("([A-Z])([a-z]+)" //Surname
              "( ([A-Z])([a-z]+))?" //Name
              " ?([0-9])?" //mb id
    );
    if (fin.is_open()) {
        for (int i = 0; i < tmpSIZE; i++) {
            getline(fin, s);
            if (regex_match(s, reg)) {
                cout << "\nEnter information about " << s << endl;
                setNewStud(s);
                addStud(SIZE + 1);
            }
        }

        cout << "\nEnter information about new student " << s << endl;
        getline(fin, s);
        setNewStud(s);

        this->SIZE = tmpSIZE;
    }
    else { cout << "File not open."; }
    fin.close();
}

StudentsWorks::~StudentsWorks() {
    if (list) {

```

```

        for (int i = 0; i < SIZE; i++) {
            if(list[i])delete list[i];
        }
        delete[] list;
    }
    delete newStudent;
}

TestsInfo& StudentsWorks::operator[](int index){
    return *list[index];
}

ostream& operator<< (ostream& os, const StudentsWorks& o) {
    os << o.SIZE << endl;
    for (int i = 0; i < o.SIZE; i++) {
        os << o.list[i];
    }
    return os;
}

istream& operator>> (istream& is, StudentsWorks& o) {
    is >> o.SIZE;
    for (int i = 0; i < o.SIZE; i++) {
        is >> *o.list[i];
    }
    return is;
}

void StudentsWorks::setNewStud(string s) {

    regex reg("([A-Z])([a-z]+)"          //Surname
              "( ([A-Z])([a-z]+))?"    //Name
              " ?([0-9])?" );          //mb id

    if (regex_match(s, reg)) {

        int works, pages, a, b;

        int i = 2;
        while (i > 1) {
            cout << "Enter the number of student works(0-10):";
            cin >> works;
            if (works < 0 || works > 10) {
                cout << "You entered an invalid value, try again\n";
            }
            else { i = 1; }
        }
        while (i > 0) {
            cout << "Enter the number of pages in works(1-5):";
            cin >> pages;
            if (pages < 1 || pages > 5) {
                cout << "You entered an invalid value, try again\n";
            }
            else { i = 0; }
        }

        cout << "From which group is the student?(Math-0,Economic-1)\n";
        bool z;
        cin >> z;

        if (z) {
            i = 2;
            while (i > 1) {
                cout << "Enter the number of statistics(1-5):";
                cin >> a;
                if (a < 0 || a > 5) {
                    cout << "You entered an invalid value, try again\n";
                }
                else { i = 1; }
            }
            while (i > 0) {
                cout << "Enter the number of formulrs(1-3):";
                cin >> b;
                if (b < 1 || b > 3) {
                    cout << "You entered an invalid value, try again\n";
                }
                else { i = 0; }
            }
        }
    }
}

```



```

    }
    newStudent = new EconomicTI(SIZE + 1, works, pages, s, a, b);

}
else {
    i = 2;
    while (i > 1) {
        cout << "Enter the number of graphs (1-5):";
        cin >> a;
        if (a < 0 || a > 5) {
            cout << "You entered an invalid value, try again\n";
        }
        else { i = 1; }
    }
    while (i > 0) {
        cout << "Enter the number of arguments(1-3):";
        cin >> b;
        if (b < 1 || b > 3) {
            cout << "You entered an invalid value, try again\n";
        }
        else { i = 0; }
    }

    newStudent = new MathTI(SIZE + 1, works, pages, s, a, b);
}
else {
    cout << "Surname entered incorrectly\n";
}
}

int StudentsWorks::getSize() { return this->SIZE; }

void StudentsWorks::writeToFile(){
    ofstream fout;
    fout.open("InfoStud.txt");
    for (int i = 0; i < SIZE; i++) {
        list[i]->outputToFile(fout);
    }
    fout.close();
}

void StudentsWorks::infoWorks(const int num) {
    cout << "The number of finished student works is " << (*list[num]).getWorks() << "/10\n";
}

void StudentsWorks::addStud(const int num) {
    SIZE++;
    TestsInfo** tmpList = new TestsInfo*[SIZE];
    for (int i = 0; i < num - 1; i++)
        tmpList[i] = list[i];
    tmpList[num - 1] = newStudent;
    for (int i = num; i < SIZE; i++)
        tmpList[i] = list[i - 1];

    delete[] list;
    list = tmpList;
}

void StudentsWorks::removeStud(const int num) {
    if (SIZE > 1) {
        SIZE--;
        TestsInfo** tmpList = new TestsInfo * [SIZE];
        for (int i = 0; i < num; i++)
            tmpList[i] = list[i];
        for (int i = num; i < SIZE; i++)
            tmpList[i] = list[i + 1];

        delete list[num];
        delete[]list;

        list = tmpList;
    }
    else if (SIZE == 1) {
        SIZE--;
    }
}

```

```

        delete list[0];
        delete[] list;
        list = nullptr;
    }
    else{
        cout << "Mass havn't students :| \n";
    }
}

void StudentsWorks::sortMarkStud(bool(*comp)(int x, int y)) {
    TestsInfo* tmp;
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (comp((*list[i]).getMark(), (*list[j]).getMark())) {
                tmp = list[i];
                list[i] = list[j];
                list[j] = tmp;
            }
        }
    }
}

void StudentsWorks::sortNumStud(bool(*comp)(int x, int y)) {
    TestsInfo* tmp;
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (comp((*list[i]).getNum(), (*list[j]).getNum())) {
                tmp = list[i];
                list[i] = list[j];
                list[j] = tmp;
            }
        }
    }
}

void StudentsWorks::sortWorksStud(bool(*comp)(int x, int y)) {
    TestsInfo* tmp;
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (comp((*list[i]).getWorks(), (*list[j]).getWorks())) {
                tmp = list[i];
                list[i] = list[j];
                list[j] = tmp;
            }
        }
    }
}

void StudentsWorks::sortPageStud(bool(*comp)(int x, int y)) {
    TestsInfo* tmp;
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (comp((*list[i]).getPages(), (*list[j]).getPages())) {
                tmp = list[i];
                list[i] = list[j];
                list[j] = tmp;
            }
        }
    }
}

void StudentsWorks::printStudIndex(const int num)const {
    for (int i = 0; i < SIZE; i++) {
        if ((*list[i]).getNum() == num) {
            (*list[i]).printInfo();
            return;
        }
    }
    cout << "This student does not exist";
}

void StudentsWorks::printNewStud() {
    (*newStudent).printInfo();
}

void StudentsWorks::printAll()const {
    cout << "Current student journal:\n\n";
    for (int i = 0; i < SIZE; i++)
        (*list[i]).printInfo();
}

```

```
void StudentsWorks::printAll2()const {

    regex reg("([A-Z])([a-z]+)//Surname
              "([A-Z])([a-z]+)?" //Name
              " ?([0-9])?");      //mb id
    for (int i = 0; i < SIZE; i++) {
        if (regex_match((*list[i]).getSur(), reg)) {
            cout << (*list[i]).getSur();
        }
    }
}
```