

Лабораторна робота 5. АГРЕГАЦІЯ ТА КОМПОЗИЦІЯ

Тема. Класи. Агрегація. Композиція. Ключові слова *typedef* та *auto*.

Мета: отримати поняття агрегація та композиція; отримати знання про призначення ключових слів *typedef* та *auto*.

1. Завдання до роботи

Загальне завдання. Дослідити заздалегідь визначені типи даних з бібліотеки `<cstdlib>` / `<stdlib.h>`. Модернізувати розроблені у попередній роботі класи таким чином:

- замінити типи даних, що використовуються при індексуванні на типи з указаної бібліотеки;
- створити власний синонім типу, визначивши його необхідність;
- створити / оновити функцію сортування масиву, де крім поля, по якому виконується сортування, передається і вказівник на функцію, яка визначає напрям сортування;
- у базовий клас додати два поля, що мають кастомний тип даних (тип даних користувача) та які будуть відображати відношення «агрегація» та «композиція», при цьому оновити методи читання та запису об'єкта;
- ввести використання ключового слова `auto` як специфікатор зберігання типу змінної. Визначити плюси та мінуси цього використання.

2.1 Опис класів

Базовий клас: `C_Program`.

Клас, що має в собі масив базового класу та методи для роботи з ним: `CList`.

Клас, що відображає агрегативні відносини з базовим класом: `CAuthor`.

Клас, що відображає ком позитивні відносини з базовим класом: `CDate`.

2.2 Опис змінних

`int` `timeOfWork` – поле класу `C_Program`(час виконання програми).
`int` `size` – поле класу `C_Program`(розмір програми у мегабайтах).
`int` `amountOfLines` – поле класу `C_Program`(кількість рядків коду).
`int` `index` – поле класу `C_Program`(індентифікаційний номер).
`bool` `trojan` – поле класу `C_Program`(троян чи ні).
`string` `name` – поле класу `C_Program`(назва програми).
`CAuthor` `author` – поле класу `CAuthor`(автор програми).
`CDate` `date` – поле класу `CDate`(дата створення програми).
`int` `listSize` – поле класу `CList`(розмір масиву елементів класу `Program`).
`C_Program*` `list` – поле класу `C_Program`(масив елементів класу `Program`).
`CList` `list` – об'єкт класу `CList`.

C_Program newProgram, getProgram – змінні для програм, необхідні для роботи програми.

int chose = 1, value = 0, stop = 1 – змінні, необхідні для роботи функції меню.

string fileName – змінна, необхідна для запису назви файлу для роботи з ним.

2.3 Опис методів

void setListSize(int) – запис даних у змінну розміру масиву елементів класу Program (метод класу CList).

int getListSize() const – отримання даних змінної розміру масиву елементів класу Program (метод класу CList).

void createList(int) – створення масиву елементів і заповнення даними (метод класу CList).

void printAll() const – виведення даних елементів у консоль (метод класу CList).

void printOneEl(int) const – виведення даних одного елемента у консоль (метод класу CList).

void addEl(C_Program&) – додавання нового елемента в масив (метод класу CList).

void deleteEl(int) – видалення елемента з масиву (метод класу CList).

int task(int) – знаходження елементів за певним критерієм (метод класу CList).

C_Program getProgramID(int) const – отримання даних елемента по індексу (метод класу CList).

C_Program programs(int) – програми для заповнення списку (метод класу CList).

int linesInFile(string) – знаходження кількості рядків у файлі (метод класу CList).

void readFile(string) – читання даних з файлу (метод класу CList).

stringstream getOneEl(int) const – отримання строки даних (метод класу CList).

void saveToFile(string) – збереження даних у файл (метод класу CList).

void showOneEl(stringstream&) const – читання даних з рядка у консоль (метод класу CList).

void enterNewEl() – введення даних з клавіатури (метод класу CList).

void regexTask() – виведення елементів, назва яких містить 2 слова (метод класу CList).

void CList::sort(comp) – функція сортування списку елементів (метод класу CList).

~CList() – деструктор списку елементів (метод класу CList).

C_Program() – конструктор без параметра (метод класу C_Program)

C_Program(bool, int, int, int, int, string) – конструктор класу з параметрами (метод класу C_Program)

C_Program(const C_Program& other) – конструктор копіювання (метод класу C_Program)

~C_Program() – деструктор елемента (метод класу C_Program).

2.4 Опис функцій

void Menu() – функція меню.

`void Test_GetProgramID(C_List&, int&)` – тест функції знаходження та повернення об'єкту по індексу.
`void Test_AddEl(C_List&)` – тест функції додавання об'єкта до масиву об'єктів.
`void Test_DelEl(C_List&)` – тест функції видалення об'єкта з масиву об'єктів.
`void Test_Task(C_List&, int&)` – тест функції знаходження елементів за певними критеріями(індивідуальне завдання).
`void Test_Stringstream(CList&)` – тест функції отримання даних зі строки.
`void Test_ReadFile(CList&)` – тест функції читання даних з файлу.
`void Test_RegexTask(CList&)` – тест функції отримання даних програм, які містять 2 слова.
`void Test_Sort(CList& list)`

3. Текст програми

test.cpp

```
#include "program.h"
#include "list.h"
#include "author.h"
#include "date.h"

void Test_GetProgramID(CList&);
void Test_AddEl(CList&);
void Test_DelEl(CList&);
void Test_Task(CList&);
void Test_Stringstream(CList&);
void Test_ReadFile(CList&);
void Test_RegexTask(CList&);
void Test_Sort(CList&);

int main() {
    setlocale(LC_ALL, "Rus");
    CList list;
    list.createList(5);

    Test_GetProgramID(list);
    Test_AddEl(list);
    Test_DelEl(list);
    Test_Task(list);
    Test_Stringstream(list);
    Test_ReadFile(list);
    Test_RegexTask(list);
    Test_Sort(list);

    if (_CrtDumpMemoryLeaks()) cout << "\n\nЕсть утечка памяти.\n\n";
    else cout << "\n\nУтечка памяти отсутствует.\n\n";

    return 0;
}

void Test_GetProgramID(CList& list)
{
    int expected = 5678;
    int real = list.list[2].getIndex();

    if(expected == real) cout << "Тест получения элемента по ID\t\t выполнен успешно.\n";
    else cout << "Тест получения элемента по ID\t\t не выполнен успешно.\n";
}

void Test_AddEl(CList& list)
{
    C_Program newProgram;
    int size = list.getListSize();
    list.addEl(newProgram);
}
```

```

    if (list.getListSize() > size) cout << "Тест добавления элемента в список\t выполнен успешно.\n";
    else cout << "Тест добавления элемента в список\t не выполнен успешно.\n";
}
void Test_DelEl(CList& list)
{
    int size = list.getListSize();
    list.deleteEl(3);

    if (size > list.getListSize()) cout << "Тест функции удаления\t\t выполнен успешно.\n\n";
    else cout << "Тест функции удаления\t\t не выполнен успешно.\n\n";
}
void Test_Task(CList& list)
{
    int expected = 2;
    int real = list.task(200);

    cout << endl;
    if (expected == real) cout << "Тест функции нахождения элементов по параметру\t\t выполнен успешно.\n";
    else cout << "Тест функции нахождения элементов по параметру\t\t не выполнен успешно.\n";
}
void Test_Stringstream(CList& list)
{
    string nameExpected = "1234";
    stringstream funcResult = list.getOneEl(1);
    string nameReal;
    funcResult >> nameReal;

    if (nameExpected == nameReal) cout << "Тест функции stringstream\t\t выполнен успешно." << endl;
    else cout << "Тест функции stringstream\t\t не выполнен успешно." << endl;
}
void Test_ReadFile(CList& list)
{
    string filename = "data.txt";
    list.readFile(filename);

    string expected = "Notepad ";
    string real = list.list[0].getName();

    if (expected == real) cout << "Тест функции чтения из файла\t\t выполнен успешно." << endl;
    else cout << "Тест функции чтения из файла\t\t не выполнен успешно." << endl;
}
void Test_RegexTask(CList& list)
{
    cout << "Здесь должны быть программы, содержащие в названии больше 2 слов:" << endl;
    list.regexTask();
}
void Test_Sort(CList& list)
{
    int beforeSorting = list.list[0].getLines();
    list.sort(list.sortDesc);
    int afterSorting = list.list[0].getLines();
    int expected = 666;

    if (beforeSorting != afterSorting && afterSorting == expected) cout << "Тест функции сортировки\t\t\t выполнен успешно." << endl;
    else cout << "Тест функции сортировки\t\t\t не выполнен успешно." << endl;
}
}
list.h
#pragma once
#include "Program.h"

class CList
{ private:
    int listSize;

```

```

public:
    C_Program* list;
    void setListSize(int);
    int getListSize() const;

    void createList(int);
    void printAll() const;
    void printOneEl(int) const;
    void addEl(C_Program&);
    void deleteEl(int);
    int task(int);
    int linesInFile(string);
    void readFile(string);
    void saveToFile(string);
    stringstream getOneEl(int) const;
    void showOneEl(stringstream&) const;
    void enterNewEl();
    void regexTask();

    C_Program getProgramID(int) const;
    C_Program programs(int);
    ~CList();
};

```

list.cpp

```
#include "list.h"
```

```

void CList::createList(int value)
{
    listSize = value;
    list = new C_Program[listSize];

    for (size_t i = 0; i < listSize; i++)
        list[i] = programs(i);
}

void CList::setListSize(int size) { listSize = size; }
int CList::getListSize() const { return listSize; }

void CList::printAll() const
{
    cout << endl << setiosflags(ios::left);
    cout << setw(12) << "    Время" << setw(12) << "Размер";
    cout << setw(10) << "Строки" << setw(11) << "Троян";
    cout << setw(14) << "Индекс" << setw(21) << "Название";
    cout << setw(16) << "Год выпуска" << setw(11) << "Автор";
    cout << endl;

    for (size_t i = 0; i < listSize; i++)
        printOneEl(i);

    cout << endl;
}

void CList::printOneEl(int number) const
{
    cout << setiosflags(ios::left) << setw(2) << number + 1 << " ";
    cout << setw(10) << list[number].getTime();
    cout << setw(12) << list[number].getSize();
    cout << setw(9) << list[number].getLines();
    cout << setw(12) << boolalpha << list[number].getTrojan();
    cout << setw(11) << list[number].getIndex();
    cout << setw(26) << list[number].getName();
    cout << setw(14) << list[number].getYear();
    cout << setw(12) << list[number].getAuthor() << endl;
}

void CList::addEl(C_Program& newProgram)

```

```

{
    C_Program* newList = new C_Program[listSize + 1];

    for (size_t i = 0; i < listSize; i++)
        newList[i] = list[i];
    newList[listSize++] = newProgram;
    delete[] list;

    list = new C_Program[listSize];
    for (size_t i = 0; i < listSize; i++)
        list[i] = newList[i];

    delete[] newList;
    cout << "Элемент добавлен." << endl;
}
void CList::deleteEl(int index)
{
    if (listSize == 0)
    {
        cout << "список программ пуст. возвращение с выбору действий." << endl;
        return;
    }
    if (index <= 0 || index > listSize)
    {
        cout << "ошибка. неверный номер элемента. возвращение." << endl;
        return;
    }

    C_Program* newList = new C_Program[listSize - 1];

    for (size_t i = 0; i < index - 1; i++)
        newList[i] = list[i];
    for (size_t i = index - 1, j = index; j < listSize; i++, j++)
        newList[i] = list[j];
    delete[] list;

    list = new C_Program[listSize--];
    for (size_t i = 0; i < listSize; i++)
        list[i] = newList[i];
    delete[] newList;

    return;
}
int CList::task(int minimalSize)
{
    int size = 0;

    for (size_t i = 0; i < listSize; i++)
        if (list[i].getSize() > minimalSize && list[i].getTrojan() == false)
        {
            printOneEl(i);
            size++;
        }

    return size;
}
int CList::linesInFile(string filename)
{
    int size = 0;
    string line;
    regex regular("([\\d]* [\\d]* [\\d]* [\\d]* [true|false]* [\\w]* [\\d]* [\\d]* [\\d]* [A-ZА-Я]+[\\wА-Яа-я.,;:-]* [\\wА-Яа-я.,;:-]*)");

    ifstream fin(filename);
    if (!fin.is_open())
    {
        cout << "Невозможно открыть файл. Возвращение в меню." << endl;
        return 0;
    }
}

```

```

while (getline(fin, line))
{
    if (regex_match(line, regular))    size++;
    else cout << "Строка в файле не прошла проверку." << endl;
}

fin.close();
return size;
}

void CList::readFile(string filename)
{
    ifstream fin(filename);
    if (!fin.is_open())
    {
        cout << "Неверное название файла. Повторите попытку." << endl;
        return;
    }

    string line, var;
    int size = CList::linesInFile(filename);
    regex regular("([\\d]* [\\d]* [\\d]* [\\d]* [true|false]* [\\w]* [\\d]* [\\d]* [\\d]* [A-ZА-Я]+[\\wА-Яа-я,.;:-]* [\\wА-Яа-я,.;:-]*)");
    int i = 0, a = 0;
    bool b;

    delete[] list;
    list = new C_Program[size];
    while (getline(fin, line) && i < size)
    {
        if (regex_match(line.c_str(), regular))
        {
            int time, size, lines, index;
            sint day, month, year;
            string author;
            bool trojan;
            string name, name2;
            string trueFalse;
            std::istringstream temp(line);

            temp >> index;
            temp >> time;
            temp >> size;
            temp >> lines;
            temp >> trueFalse;
            temp >> author;
            temp >> day;
            temp >> month;
            temp >> year;
            temp >> name;
            temp >> name2;

            if (trueFalse == "true") trojan = true;
            else trojan = false;

            if (name2 == "") name = name + " ";
            else(name = name + " " + name2);

            do {
                b = 0;

                a = name.find("--");
                if (a != -1)
                {
                    name.erase(a, 1);
                    b = 1;
                }

                a = name.find(" ");

```

```

        if (a != -1)
        {
            name.erase(a, 1);
            b = 1;
        }

        a = name.find(",");
        if (a != -1)
        {
            name.erase(a, 1);
            b = 1;
        }

        a = name.find("::");
        if (a != -1)
        {
            name.erase(a, 1);
            b = 1;
        }

        a = name.find(";");
        if (a != -1)
        {
            name.erase(a, 1);
            b = 1;
        }

        a = name.find("_");
        if (a != -1)
        {
            name.erase(a, 1);
            b = 1;
        }
    } while (b == 1);

    C_Program newElement(trojan, time, size, lines, index, name, author, day,
month, year);
    list[i++] = newElement;
}

setListSize(size);
fin.close();
cout << endl << "Чтение из файла завершено." << endl;
}
void CList::saveToFile(string filename)
{
    std::ofstream fout(filename);

    fout.setf(ios::left);
    fout << setw(12) << "   Время" << setw(12) << "Размер";
    fout << setw(13) << "Строки" << setw(11) << "Троян";
    fout << setw(15) << "Индекс" << setw(24) << "Название";
    fout << setw(16) << "Год выпуска" << setw(11) << "Автор";
    fout << endl;

    for (size_t i = 0; i < getListSize(); i++)
    {
        fout << setiosflags(ios::left) << setw(2) << i + 1 << " ";
        fout << setw(10) << list[i].getTime();
        fout << setw(12) << list[i].getSize();
        fout << setw(12) << list[i].getLines();
        fout << setw(12) << boolalpha << list[i].getTrojan();
        fout << setw(15) << list[i].getIndex();
        fout << setw(26) << list[i].getName();
        fout << setw(14) << list[i].getYear();
        fout << setw(12) << list[i].getAuthor() << endl;
    }
}

```



```

        cout << "Запись в файл завершена." << endl;
        fout.close();
    }
    stringstream CList::getOneEl(int value) const
    {
        stringstream temp;

        temp << " " << list[value].getIndex() << " " << list[value].getTime() << " " <<
list[value].getSize()
        << " " << list[value].getLines() << " " << list[value].getTrojan() << " " <<
list[value].getAuthor()
        << " " << list[value].getDay() << " " << list[value].getMonth() << " " <<
list[value].getYear()
        << " " << list[value].getName();

        return temp;
    }
    void CList::showOneEl(stringstream& line) const
    {
        int time, size, lines, index;
        sint day, month, year;
        string author;
        bool trojan;
        string name, name2;
        string trueFalse;

        line >> index;
        line >> time;
        line >> size;
        line >> lines;
        line >> trueFalse;
        line >> author;
        line >> day;
        line >> month;
        line >> year;
        line >> name;
        line >> name2;

        if (trueFalse == "1") trojan = true;
        else trojan = false;

        if (name2 == "") name = name + " ";
        else(name = name + " " + name2);

        cout << endl << setiosflags(ios::left);
        cout << setw(12) << "   Время" << setw(12) << "Размер";
        cout << setw(10) << "Строки" << setw(11) << "Троян";
        cout << setw(14) << "Индекс" << setw(21) << "Название";
        cout << setw(16) << "Год выпуска" << setw(11) << "Автор";
        cout << endl << "   ";

        cout << setw(10) << time;
        cout << setw(12) << size;
        cout << setw(10) << lines;
        cout << setw(11) << boolalpha << trojan;
        cout << setw(10) << index;
        cout << setw(27) << name;
        cout << setw(12) << year;
        cout << setw(15) << author;
        cout << endl;
    }
    C_Program CList::getProgramID(int id) const
    {
        C_Program newProgram;

        for (size_t i = 0; i < listSize; i++)
            if (list[i].getIndex() == id)
            {

```

```

        printOneEl(i);
        newProgram = list[i];
        return newProgram;
    }
    cout << "\nПрограммы с таким ID нету.\n" << endl;
    return newProgram;
}
C_Program CList::programs(int valueX)
{
    C_Program standartProgram;

    if (valueX == 1)
    {
        C_Program Program1(true, 222, 222, 222, 1234, "Skype", (CAuthor)"Microsoft", 2, 12,
2002);
        return Program1;

        return Program1;
    }
    else if (valueX == 2)
    {
        C_Program Program2(true, 333, 333, 666, 5678, "Standart Calculator",
(CAuthor)"Bethesda", 13, 3, 1993);
        return Program2;
    }
    else if (valueX == 3)
    {
        C_Program Program3(false, 444, 444, 444, 9532, "Domino Super", (CAuthor)"Aida", 14, 4,
1944);
        return Program3;
    }
    else if (valueX == 4)
    {
        C_Program Program4(false, 555, 555, 555, 4356, "Text editor", (CAuthor)"Google", 5, 5,
1995);
        return Program4;
    }
    return standartProgram;
}
void CList::enterNewEl()
{
    int time, size, lines, index;
    bool trojan;
    string author;
    sint day, month, year;
    string name, name2, trojan2, data;
    regex regular("([\\d]* [\\d]* [\\d]* [\\d]* [true|false]* [\\w]* [\\d]* [\\d]* [\\d]* [A-ZA-
Я]+[\\wA-Яa-я,.;:-]* [\\wA-Яa-я,.;:-]*)");

    cout << "Введите данные в линию в таком порядке:";
    cout << "Индекс Время Размер Строки Троян(true/false) Компания День Месяц Год Название" <<
endl;

    cin.ignore();
    getline(cin, data);

    data = data + " ";

    if (regex_match(data, regular))
    {
        std::istringstream temp(data);

        temp >> index;
        temp >> time;
        temp >> size;
        temp >> lines;
        temp >> trojan2;
        temp >> author;
        temp >> day;

```

```

        temp >> month;
        temp >> year;
        temp >> name;
        temp >> name2;

        if (name2 == "") name = name + " ";
        else(name = name + " " + name2);

        if (trojan2 == "true") trojan = true;
        else trojan = false;

        C_Program newProgram(trojan, time, size, lines, index, name, author, day, month,
year);
        addEl(newProgram);
    }
    else
    {
        cout << endl << "Было введено неправильное имя. Формат имени:" << endl;
        cout << "Первое слово в названии программы не должно начинаться с маленькой буквы." <<
endl;

        cout << "Не должно содержать символы." << endl;
        cout << "Завершение работы функции." << endl;
    }

    return;
}
void CList::regexTask()
{
    regex regular("(^[A-ZА-Я]+[\\wА-Яа-я.,;:-]* [\\wА-Яа-я.,;:-]+)");
    int listSize = getListSize();

    for (size_t i = 0; i < listSize; i++)
        if (regex_match(list[i].getName(), regular))
            printOneEl(i);

    cout << endl;
}
bool CList::sortAsc(const int& a, const int& b) { return a > b; }
bool CList::sortDesc(const int& a, const int& b) { return a < b; }
void CList::sort(comp condition)
{
    C_Program temp;
    int size = getListSize();
    bool pr;

    do {
        pr = 0;
        for (size_t i = 0; i < size-1; i++)
        {
            if (condition(list[i].getLines(), list[i+1].getLines()))
            {
                temp = list[i]; list[i]
                = list[i + 1]; list[i
                + 1] = temp;
                pr = 1;
            }
        }
    } while (pr == 1);
}
CList::~CList()
{
    //cout << "\nВызвался деструктор" << endl;
    delete[] list;
}

```

main.cpp

```
#include "program.h"
#include "list.h"
#include "author.h"
#include "date.h"

void menu();

int main()
{
    setlocale(LC_ALL, "Rus");
    menu();

    if (_CrtDumpMemoryLeaks()) cout << endl << "Есть утечка памяти." << endl;
    else cout << endl << "Утечка памяти отсутствует." << endl;

    return 0;
}

void menu()
{
    CList list; //список элементов
    CAuthor author; //переменная поля автор
    string* listAuthor;
    C_Program getProgram; //программа, которая вернётся при получении ID
    C_Program newProgram; //программа для добавления в список
    auto choise = 1, choise2 = 0;
    auto value = 0, stop = 1;
    string fileName; //название файла
    string::size_type n;
    stringstream str;
    int size; //количество элементов больше определённого размера
    listAuthor = author.createList(4);
    list.createList(4);

    cout << endl << "Выберите команду для работы со списком: ";
    while (stop != 0)
    {
        if (list.getListSize() == 0)
        {
            cout << "Список пуст. Что вы хотите сделать?" << endl;
            cout << "1) Добавить элемент вручную" << endl;
            cout << "2) Прочитать данные из файла" << endl;
            cout << "3) Завершение работы" << endl;
            cout << "======" << endl;
            cout << "Ваш выбор: ";
            cin >> choise;
            cout << endl;

            switch (choise)
            {
            case 1:
                list.enterNewEl();
                break;

            case 2:
                cout << "Введите название файла для чтения данных: ";
                cin >> fileName;
                cout << endl;

                n = fileName.find(".txt");
                if (n > 187) fileName += string(".txt");

                list.readFile(fileName);
                break;

            case 3:
                cout << "Завершение работы." << endl;
            }
        }
    }
}
```

```

        stop = 0;
        break;

    default:
        cout << "Неверный номер элемента. Повторите попытку." << endl;
        break;
    }
}
else
{
    cout << endl;
    cout << "1)Вывод на экран" << endl;
    cout << "2)Работа с файлами" << endl;
    cout << "3)Сортировка данных" << endl;
    cout << "4)Удаление элемента" << endl;
    cout << "5)Добавление элементов" << endl;
    cout << "6)Завершение работы" << endl;
    cout << "======" << endl;
    cout << "Ваш выбор: ";
    cin >> choise;
    cout << endl;
}

switch (choise)
{
    case 1:
        cout << "Выберите команду:" << endl;
        cout << "1) Вывести весь список на экран" << endl;
        cout << "2) Вывести один элемент на экран по номеру" << endl;
        cout << "3) Вывести список программ меньше определённого размера и не трояны"
<< endl;

        cout << "4) Вывести список программ, названия которых состоят из 2 слов" <<

        cout << "5) Получить строку с данными" << endl;
        cout << "6) Вывести программу по ID" << endl;
        cout << "7) Вернуться к выбору действий" << endl;
        cout << "======" << endl;
        cout << "Ваш выбор: ";
        cin >> choise2;
        cout << endl;

        switch (choise2)
        {
            case 1:
                list.printAll();
                break;

            case 2:
                cout << "Введите номер элемента, который надо вывести: ";
                cin >> value;
                cout << endl;

                if (value <= 0 || value > list.getListSize())
                {
                    cout << "Неверный номер элемента. Повторите попытку." << endl;
                    break;
                }

                list.printOneEl(value - 1);
                break;

            case 3:
                cout << "Введите минимальный размер программ: ";
                cin >> value;
                cout << endl;

                size = list.task(value);
                break;

```

```

case 4:
    list.regexTask();
    break;

case 5:
    cout << "Введите номер элемента, который вы хотите получить: ";
    cin >> value;
    cout << endl;

    str = list.getOneEl(value - 1);
    list.showOneEl(str);
    break;

case 6:
    cout << "Введите id элемента, которого вы хотите получить: ";
    cin >> value;
    cout << endl;

    getProgram = list.getProgramID(value);
    break;

case 7:
    break;

default:
    cout << "Неверный символ. Повторите попытку." << endl;
    break;
}
break;
case 2:
    cout << "Выберите команду:" << endl;
    cout << "1) Сохранить данные в файл" << endl;
    cout << "2) Читать данные из файла" << endl;
    cout << "3) Вернуться к выбору" << endl;
    cout << "===== " << endl;
    cout << "Ваш выбор: ";
    cin >> choose2;
    cout << endl;

    switch (choose2)
    {
    case 1:
        cout << "Введите название файла для записи данных: ";
        cin >> fileName;
        cout << endl;

        n = fileName.find(".txt");
        if (n > 187) fileName += string(".txt");

        list.saveToFile(fileName);
        break;
    case 2:
        cout << "Введите название файла для чтения данных: ";
        cin >> fileName;
        cout << endl;

        n = fileName.find(".txt");
        if (n > 187) fileName += string(".txt");

        list.readFile(fileName);
        break;
    case 3:
        break;
    default:
        cout << "Неверный символ. Повторите попытку." << endl;
        break;
    }
    break;
case 3:

```

```

        cout << "Сортировать количество строк по:" << endl;
        cout << "1) Возрастаю" << endl;
        cout << "2) Убыванию" << endl;
        cout << "Ваш выбор: ";
        cin >> value;
        cout << endl;

        if (value == 1) list.sort(list.sortAsc);
        else if (value == 2) list.sort(list.sortDesc);
        else cout << "Ошибка. Неверный номер команды." << endl;

        break;

    case 4:
        cout << "Введите номер элемента, который хотите удалить: ";
        cin >> value;
        cout << endl;

        list.deleteEl(value);
        break;

    case 5:
        cout << "Выберите команду:" << endl;
        cout << "1) Добавить стандартную программу" << endl;
        cout << "2) Ввести данные с клавиатуры" << endl;
        cout << "3) Вернуться к выбору" << endl;
        cout << "===== " << endl;
        cout << "Ваш выбор: ";
        cin >> choose2;
        cout << endl;

        switch (choose2)
        {
            case 1:
                list.addEl(newProgram);
                break;

            case 2:
                list.enterNewEl();
                break;

            case 3:
                break;

            default:
                cout << "Неверный символ. Повторите попытку." << endl;
                break;
        }

        break;

    case 6:
        cout << "Завершение работы." << endl;
        stop = 0;
        break;

    default:
        cout << "Неверный символ. Повторите попытку." << endl;
        break;
}

author.deleteList();
return;
}

```

program.cpp

```
#include "program.h"
```

```
int C_Program::getTime() const
{
    return timeOfWork;
}
int C_Program::getSize() const
{
    return size;
}
int C_Program::getLines() const
{
    return amountOfLines;
}
int C_Program::getIndex() const
{
    return index;
}
bool C_Program::getTrojan()const
{
    return trojan;
}
string C_Program::getName()const
{
    return name;
}
sint C_Program::getDay()const
{
    return date.getDay();
}
sint C_Program::getMonth()const
{
    return date.getMonth();
}
sint C_Program::getYear()const
{
    return date.getYear();
}
string C_Program::getAuthor() const
{
    return author.getAuthor();
}

void C_Program::setTime(const int valueTime)
{
    timeOfWork = valueTime;
}
void C_Program::setSize(const int valueSize)
{
    size = valueSize;
}
void C_Program::setLines(const int valueLines)
{
    amountOfLines = valueLines;
}
void C_Program::setTrojan(const bool trojanStatus)
{
    trojan = trojanStatus;
}
void C_Program::setIndex(const int valueIndex)
{
    index = valueIndex;
}
void C_Program::setName(const string valueName)
{
    name = valueName;
}
```



```

void C_Program::setDay(const sint valueDay)
{
    date.setDay(valueDay);
}
void C_Program::setMonth(const sint valueMonth)
{
    date.setMonth(valueMonth);
}
void C_Program::setYear(const sint valueYear)
{
    date.setYear(valueYear);
}
void C_Program::setAuthor(const string valueAuthor)
{
    author.setAuthor(valueAuthor);
}

C_Program::C_Program(bool trojan, int time, int size, int lines, int index, string name, CAuthor
author, sint day, sint month, sint year): trojan(trojan), timeOfWork(time), size(size),
amountOfLines(lines), index(index), name(name), author(author), date(day, month, year)
{
    //cout << "\nВызвался конструктор с параметрами";
}
C_Program::C_Program() : trojan(true), timeOfWork(0), size(0), amountOfLines(0), index(0101),
name("Basic"), date(1,1,111)
{
    //cout << "\nВызвался конструктор по умолчанию.";
}
C_Program::C_Program(const C_Program& other) : trojan(other.trojan), timeOfWork(other.timeOfWork),
size(other.size), amountOfLines(other.amountOfLines), index(other.index), name(other.name),
author(other.author), date(other.date)
{
    //cout << "\nВызвался конструктор копирования.";
}
C_Program::~C_Program()
{
    //cout << "\nВызвался деструктор";
}

```

program.h

```

#pragma once

#define _CRT_SECURE_NO_WARNINGS
#include <string.h>
#define CRTDBG_MAP_ALLOC
#include <crtdbg.h>
#define DEBUG_NEW new(_NORMAL_BLOCK, FILE, __LINE)

#include <string>
#include <iostream>
#include <iomanip>
#include <locale.h>
#include <fstream>
#include <sstream>
#include <regex>

using std::string;
using std::cin;
using std::cout;
using std::endl;
using std::setw;
using std::stringstream;
using std::boolalpha;
using std::regex;
using std::ifstream;
using std::regex_match;
using std::regex_search;

```

```

using std::cmatch;
using std::setiosflags;
using std::ios;

#include "author.h"
#include "date.h"

typedef bool (comp)(const int&, const int&);

class C_Program {
private:
    int timeOfWork;           //average time of program execution
    int size;                 //size of program
    int amountOfLines;        //number of lines in code
    int index;                //index
    bool trojan;              //trojan(yes or no)
    string name;              //name of program
    CAuthor author;           //creator of program
    CDate date;               //date of creating program
public:
    int getTime() const;
    int getSize() const;
    int getLines() const;
    int getIndex()const;
    bool getTrojan()const;
    string getName() const;
    sint getDay()const;
    sint getMonth()const;
    sint getYear()const;
    string getAuthor()const;

    void setTime(const int);
    void setSize(const int);
    void setLines(const int);
    void setIndex(const int);
    void setTrojan(const bool);
    void setName(const string);
    void setDay(const sint);
    void setMonth(const sint);
    void setYear(const sint);
    void setAuthor(const string);

    C_Program();
    C_Program(bool, int, int, int, int, string, CAuthor, sint, sint, sint);
    C_Program(const C_Program& other);
    ~C_Program();
};

```

author.h

```

#pragma once
#include <iostream>
#include <string>

using std::string;

class CAuthor
{ private:
    string companyName;
    int listSize;
    string* list;

public:
    void setAuthor(const string);
    string getAuthor()const;

    string* createList(int size);
    void deleteList();

```

```

    string authors(int value);

    CAuthor();
    CAuthor(string author);
    CAuthor(const CAuthor& other);
    ~CAuthor();
};

```

author.cpp

```

#include "author.h"

string CAuthor::getAuthor() const
{
    return companyName;
}

void CAuthor::setAuthor(string name)
{
    companyName = name;
}

string* CAuthor::createList(int size)
{
    listSize = size;
    list = new string[size];

    for (size_t i = 0; i < size; i++)
    {
        list[i] = authors(i);
    }

    return list;
}

void CAuthor::deleteList()
{
    delete[] list;
}

string CAuthor::authors(int value)
{
    string author;

    switch (value)
    {
    case 1:
        author = { "Microsoft" };
        return author;
    case 2:
        author = { "Lambda" };
        return author;
    case 3:
        author = { "AMD" };
        return author;
    case 4:
        author = { "Logitech" };
        return author;
    default:
        return author;
    }
}

CAuthor::CAuthor() : companyName("IBM") {}
CAuthor::CAuthor(string author) : companyName(author) {}
CAuthor::CAuthor(const CAuthor& other) : companyName(other.companyName) {}
CAuthor::~CAuthor() {}

```

date.h

```
#pragma once
typedef short sint;

class CDate
{
private:
    sint day;
    sint month;
    sint year;
public:
    void setDay(sint day);
    void setMonth(sint month);
    void setYear(sint year);

    sint getDay() const;
    sint getMonth() const;
    sint getYear() const;

    CDate();
    CDate(sint, sint, sint);
    CDate(const CDate& other);
    ~CDate();
};
```

date.cpp

```
#include "date.h"

void CDate::setDay(sint day)
{
    this->day = day;
}
void CDate::setMonth(sint month)
{
    this->month = month;
}
void CDate::setYear(sint year)
{
    this->year = year;
}

sint CDate::getDay() const
{
    return day;
}
sint CDate::getMonth() const
{
    return month;
}
sint CDate::getYear() const
{
    return year;
}

CDate::CDate() : day(1), month(1), year(1999) {}
CDate::CDate(sint day, sint month, sint year) : day(day), month(month), year(year) {}
CDate::CDate(const CDate& other) : day(other.day), month(other.month), year(other.year) {}
CDate::~CDate() {}
```

4. Результати роботи програми

```
1) Вывести весь список на экран
2) Вывести один элемент на экран по номеру
3) Вывести список программ меньше определенного размера и не трояны
4) Вывести список программ, названия которых состоят из 2 слов
5) Получить строку с данными
6) Вывести программу по ID
7) Вернуться к выбору действий
=====
Ваш выбор: 1

  Время   Размер   Строки   Троян   Индекс   Название           Год выпуска   Автор
1 )0      0          0        true    65       Basic              111          IBM
2 )222    222        222      true    1234     Skype             2002         Microsoft
3 )333    333        666      true    5678     Standart Calculator 1993         Bethesda
4 )444    444        444      false   9532     Domino Super       1944         Aida

1)Вывод на экран
2)Работа с файлами
3)Сортировка данных
4)Удаление элемента
5)Добавление элементов
6)Завершение работы
=====
Ваш выбор: 3

Сортировать количество строк по:
1) Возрастанию
2) Убыванию
Ваш выбор: 2

1)Вывод на экран
2)Работа с файлами
3)Сортировка данных
4)Удаление элемента
5)Добавление элементов
6)Завершение работы
=====
Ваш выбор: 1

Выберите команду:
1) Вывести весь список на экран
2) Вывести один элемент на экран по номеру
3) Вывести список программ меньше определенного размера и не трояны
4) Вывести список программ, названия которых состоят из 2 слов
5) Получить строку с данными
6) Вывести программу по ID
7) Вернуться к выбору действий
=====
Ваш выбор: 1

  Время   Размер   Строки   Троян   Индекс   Название           Год выпуска   Автор
1 )333    333        666      true    5678     Standart Calculator 1993         Bethesda
2 )444    444        444      false   9532     Domino Super       1944         Aida
3 )222    222        222      true    1234     Skype             2002         Microsoft
4 )0      0          0        true    65       Basic              111          IBM
```

```
Тест получения элемента по ID      выполнен успешно.
Элемент добавлен.
Тест добавления элемента в список  выполнен успешно.
Тест функции удаления              выполнен успешно.

3 )444    444        444      false   9532     Domino Super       1944         Aida
4 )555    555        555      false   4356     Text editor        1995         Google

Тест функции нахождения элементов по параметру выполнен успешно.
Тест функции stringstream          выполнен успешно.
Строка в файле не прошла проверку.

Чтение из файла завершено.
Тест функции чтения из файла      выполнен успешно.
Здесь должны быть программы, содержащие в названии больше 2 слов:
2 )666    666        666      true    666     Photo-shop; CPlus54 2012         Abra

Тест функции сортировки           выполнен успешно.

Утечка памяти отсутствует.
```

5. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з агрегативними та композитивними типами зв'язків класів, ключовими словами *typedef* та *auto*.

Програма протестована, витоків пам'яті немає, виконується без помилок.