Звіт

Автор: Момот Р.Є. КІТ-119а

Дата: 24 лютого 2020

Лабораторна робота 2. ПЕРЕВАНТАЖЕННЯ МЕТОДІВ

Тема. Класи. Конструктори та деструктори. Перевантаження методів.
Мета: отримати базові знання про класи, конструктори та деструктори. Дослідити механізм створення та видалення об'єктів.

1. Завдання до роботи

Загальне завдання. Поширити попередню лабораторну роботу таким чином:

1) в базовому класі необхідно додати:
- мінімум одне поле типу char*;
- конструктор за замовчуванням, копіювання та конструктор з аргументами;
- деструктор;

Індивідуальне завдання. Отримати список програм, розмір яких більше заданого розміру (наприклад, 100 К байт). Зі списку виключити «трояни».

2. Опис класів, змінних, методів та функцій

2.1 Опис класів

Базовий клас: C_Program
Клас, що має в собі масив базового класу та методи для роботи з ним: C_List

2.1 Опис змінних

`int TimeOfWork` – поле класу Program(час виконання програми).
`int size` – поле класу Program(розмір програми у мегабайтах).
`int AmountOfLines` – поле класу Program(кількість рядків коду).
`int index` – поле класу Program(індентифікаційний номер).
`const char* trojan` – поле класу Program(троян чи ні).
`int listSize` – поле класу C_List(розмір масиву елементів класу Program).
`C_Program* List` – поле класу C_Program(масив елементів класу Program).
`C_List List` – об'єкт класу C_List.
`C_Program newProgram` – змінні для нових програм, необхідні для роботи програми.

2.2 Опис методів

`void setListSize(int)` – запис даних у змінну розміру масиву елементів класу Program (метод класу C_List).
`int getListSize() const` – отримання даних змінної розміру масиву елементів класу Program (метод класу C_List).
`void CreateList(int)` – створення масиву елементів і заповнення даними (метод класу C_List).
`void PrintAll() const` – виведення даних елементів у консоль (метод класу C_List).
`void PrintOneEl(int) const` – виведення даних одного елементу у консоль (метод класу C_List).
`void AddEl(C_Program&)` – додавання нового елементу в масив (метод класу C_List).
`void DeleteEl(int)` – видалення елемента з масиву (метод класу C_List).
`void Task(int)` – знаходження елементів за певним критерієм (метод класу C_List).
`void GetProgramID(int)const` - отримання даних елемента по індексу (метод класу C_List).
`C_Program Programs(int)` – програми для заповнення списку (метод класу C_List).
`~C_List()` – деструктор списку елементів (метод класу C_List).
`C_Program()`– конструктор без параметра (метод класу C_Program)
`C_Program(char*, int, int, int, int)` – конструктор класса з параметрами (метод класу C_Program)
`C_Program(const C_Program& other)` – конструктор копіювання (метод класу C_Program)
`~C_Program()` – деструктор елементу (метод класу C_Program).

2.3 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).

3. Текст програми

test.cpp

```
#include "Program.h"
#include "List.h"

void Test_GetProgramID(C_List&, int&);
void Test_AddEl(C_List&);
void Test_DelEl(C_List&);
void Test_Task(C_List&, int&);

int main() {
    setlocale(LC_ALL, "Rus");
    C_List List;
    List.CreateList(5);

    int values[] = { 5678, 200 };
```

```
    Test_GetProgramID(List, values[0]);
    Test_AddEl(List);
    Test_DelEl(List);
    List.PrintAll();
    Test_Task(List, values[1]);

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

    return 0;
}
void Test_GetProgramID(C_List& list, int& value)
{
    printf("\n\nЗдесь должен быть элемент с идентификатором 5678:\n");
    list.GetProgramID(value);
}
void Test_AddEl(C_List& list)
{
    C_Program newProgram;
    int size = list.getListSize();
    list.AddEl(newProgram);

    if (list.getListSize() > size)
        printf("\n\nТест добавления элемента в список\t выполнен успешно.\n\n");
    else printf("\n\nТест добавления элемента в список\t не выполнен успешно.\n\n");
}
void Test_DelEl(C_List& list)
{
    int size = list.getListSize();
    list.DeleteEl(3);

    if (size > list.getListSize())
        printf("\n\nТест функции удаления\t\t выполнен успешно.\n\n");
    else printf("\n\nТест функции удаления\t\t не выполнен успешно.\n\n");
}
void Test_Task(C_List& list, int& value)
{
    printf("\n\nЗдесь должны быть элементы размером больше 200 и не трояны:\n");
    list.Task(value);
}
```

List.h

```
#pragma once
#include "Program.h"

class C_List {
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);
    void Task(int);
    void GetProgramID(int)const;
    C_Program Programs(int);
    ~C_List();
};
```

List.cpp

```
#include "List.h"

void C_List::CreateList(int value)
{
    listSize = value;
    List = new C_Program[listSize];

    for (int i = 0; i < listSize; i++)
        List[i] = Programs(i);
}
void C_List::setListSize(int size)
{
    listSize = size;
}
int C_List::getListSize() const
{
    return listSize;
}
void C_List::PrintAll() const
{
    printf("  \nВремя\t\tРазмер\t\tСтроки\t\tИндекс\t\tТроян");
    for (int i = 0; i < listSize; i++)
        PrintOneEl(i);
}
void C_List::PrintOneEl(int number) const
{
    printf("\n%2i) %-10i\t %-10i\t ", number + 1, List[number].getTime(), List[number].getSize());
    printf("%-10i\t %-10i\t %-10s", List[number].getLines(), List[number].getIndex(),
List[number].getTrojan());
}
void C_List::AddEl(C_Program& newProgram)
{
    C_Program* newList = new C_Program[listSize + 1];

    for (int i = 0; i < listSize; i++)
        newList[i] = List[i];
    newList[listSize++] = newProgram;
    delete[] List;

    List = new C_Program[listSize];
    for (int i = 0; i < listSize; i++)
        List[i] = newList[i];

    printf("Элемент добавлен.\n");

    delete[] newList;
}
void C_List::DeleteEl(int index)
{
    if (listSize == 0)
    {
        printf("Список программ пуст. Возвращение с выбору действий.\n");
        return;
    }
    if (index <= 0 || index > listSize)
    {
        printf("Ошибка. Неверный номер элемента. Вовзвращение.\n");
        return;
    }

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

    for (int i = 0; i < index - 1; i++)
        newList[i] = List[i];
    for (int i = index - 1, j = index; j < listSize; i++, j++)
        newList[i] = List[j];
    delete[] List;
```

```
    List = new C_Program[listSize--];
    for (int i = 0; i < listSize; i++)
        List[i] = newList[i];
    delete[] newList;

    return;
}
void C_List::Task(int minimalSize)
{
    char b[] = "Нет";
    for (int i = 0; i < listSize; i++)
        if (List[i].getSize() > minimalSize&& strcmp(List[i].getTrojan(), b) == 0)
            PrintOneEl(i);

}
void C_List::GetProgramID(int id) const
{
    int newListSize = 0;

    for (int i = 0; i < listSize; i++)
        if (List[i].getIndex() == id)
        {
            PrintOneEl(i);
            newListSize++;
        }
}
C_Program C_List::Programs(int valueX)
{
    C_Program standartProgram;

    if (valueX == 1)
    {
        static char status[] = "Да";
        C_Program Program1(status, 222, 222, 222, 1234);
        return Program1;
    }
    else if (valueX == 2)
    {
        static char status[] = "Да";
        C_Program Program2(status, 333, 333, 666, 5678);
        return Program2;
    }
    else if (valueX == 3)
    {
        static char status[] = "Нет";
        C_Program Program3(status, 444, 444, 444, 9532);
        return Program3;
    }
    else if (valueX == 4)
    {
        static char status[] = "Нет";
        C_Program Program4(status, 555, 555, 555, 4356);
        return Program4;
    }
    return standartProgram;
}
C_List::~C_List()
{
    printf("\nВызвался деструктор");
    delete[] List;
}
```

Main.cpp

```
#include "Program.h"
#include "List.h"

void Menu();

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

    if (_CrtDumpMemoryLeaks()) printf("\nЕсть утечка памяти.\n");
    else printf("\nНет утечки памяти.\n");

    return 0;
}

void Menu()
{
    C_List List;
    C_Program newProgram;
    int choise = 1, value = 0, stop = 1;
    List.CreateList(4);
    printf("\nВыберите команду для работы со списком:\n");
    while (stop != 0)
    {
        if (List.getListSize() == 0)
        {
            printf("Список пуст. Добавить элемент(1) или закончить работу(0): ");
            scanf("%i\n", &choise);

            if (choise == 1) choise = 3;
            else if (choise == 0) choise = 5;
            else(printf("Неверный символ.\n"));
        }
        else
        {
            printf("\n\n1)Вывести всё на экран\n2)Вывести 1 элемент на экран\n");
            printf("3)Добавить элемент(в конец)\n4)Удалить 1 элемент\n5)Завершение работы\n");
            printf("6)Вывести программы по индексу\n7)Получить список программ больше ");
            printf("определённого размера и не трояны\n============================\nВаш выбор: ");
            scanf("%i", &choise);
        }

        switch (choise)
        {
        case 1:
            List.PrintAll();
            break;
        case 2:
            printf("Введите номер элемента, который надо вывести: ");
            scanf("%i", &value);
            if (value <= 0 || value > List.getListSize())
            {
                printf("Неверный номер элемента. Повторите попытку.\n");
                break;
            }
            List.PrintOneEl(value - 1);
            break;
        case 3:
            List.AddEl(newProgram);
            break;
        case 4:
            printf("Введите номер элемента, который хотите удалить: ");
            scanf("%i", &value);
            List.DeleteEl(value);
            break;
        case 5:
```

```
            printf("Завершение работы.\n");
            stop = 0;
            break;
        case 6:
            printf("Введите индекс элемента, которого вы хотите получить: ");
            scanf("%i", &value);
            List.GetProgramID(value);
            break;
        case 7:
            printf("Введите минимальный размер программ: ");
            scanf("%i", &value);
            List.Task(value);
            break;
        default:
            printf("Неверный символ. Повторите попытку.\n");
            break;
        }
    }
    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;
}
const char* C_Program::getTrojan()const
{
    return trojan;
}
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 char* trojanStatus)
{
    trojan = trojanStatus;
}
void C_Program::setIndex(int valueIndex)
{
    index = valueIndex;
}

C_Program::C_Program(char* trojan, int time, int size, int lines, int index)
{
```

```
    printf("\nВызвался конструктор с параметрами");
    this->trojan = trojan;
    TimeOfWork = time;
    this->size = size;
    this->index = index;
    AmountOfLines = lines;
}
C_Program::C_Program()    //конструктор по умолчанию
{
    printf("\nВызвался конструктор по умолчанию.");
    trojan = "Да";
    TimeOfWork = 0000;
    size = 0000;
    index = 0000;
    AmountOfLines = 0000;
}
C_Program::~C_Program()  //деструктор
{
    printf("\nВызвался деструктор");
}
C_Program::C_Program(const C_Program& other)  //конструктор копирования
{
    printf("\nВызвался конструктор копирования.");
    this->trojan = other.trojan;
    this->TimeOfWork = other.TimeOfWork;
    this->size = other.size;
    this->AmountOfLines = other.AmountOfLines;
    this->index = other.index;
}
```

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 <stdio.h>
#include <locale.h>

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
    const char* trojan; //trojan(yes or no)
public:
    int getTime() const;
    int getSize() const;
    int getLines() const;
    int getIndex()const;
    const char* getTrojan()const;

    void setTime(const int);
    void setSize(const int);
    void setLines(const int);
    void setIndex(const int);
    void setTrojan(const char*);

    C_Program();
    C_Program(char*, int, int, int, int);
    C_Program(const C_Program& other);
    ~C_Program();
};
```

```
1)Вывести всё на экран
2)Вывести 1 элемент на экран
3)Добавить элемент(в конец)
4)Удалить 1 элемент
5)Завершение работы
6)Вывести программы по индексу
7)Получить список программ больше определённого размера и не трояны
============================
Ваш выбор: 1

Время           Размер          Строки          Индекс          Троян
 1) 0            0               0               0               Да
 2) 222          222             222             1234            Да
 3) 333          333             666             5678            Да
 4) 444          444             444             9532            Нет
```

```
Время           Размер          Строки          Индекс          Троян
 1) 0            0               0               0               Да
 2) 222          222             222             1234            Да
 3) 333          333             666             5678            Да
 4) 444          444             444             9532            Нет
 5) 0            0               0               0               Да

1)Вывести всё на экран
2)Вывести 1 элемент на экран
3)Добавить элемент(в конец)
4)Удалить 1 элемент
5)Завершение работы
6)Вывести программы по индексу
7)Получить список программ больше определённого размера и не трояны
============================
Ваш выбор: 6
Введите индекс элемента, которого вы хотите получить: 1234

 2) 222          222             222             1234            Да
```

```
Время           Размер          Строки          Индекс          Троян
 1) 0            0               0               0               Да
 2) 222          222             222             1234            Да
 3) 333          333             666             5678            Да
 4) 444          444             444             9532            Нет
 5) 0            0               0               0               Да

1)Вывести всё на экран
2)Вывести 1 элемент на экран
3)Добавить элемент(в конец)
4)Удалить 1 элемент
5)Завершение работы
6)Вывести программы по индексу
7)Получить список программ больше определённого размера и не трояны
============================
Ваш выбор: 2
Введите номер элемента, который надо вывести: 4

 4) 444          444             444             9532            Нет
```

```
Время           Размер          Строки          Индекс          Троян
 1) 0            0               0               0               Да
 2) 222          222             222             1234            Да
 3) 333          333             666             5678            Да
 4) 444          444             444             9532            Нет
 5) 0            0               0               0               Да

1)Вывести всё на экран
2)Вывести 1 элемент на экран
3)Добавить элемент(в конец)
4)Удалить 1 элемент
5)Завершение работы
6)Вывести программы по индексу
7)Получить список программ больше определённого размера и не трояны
============================
Ваш выбор: 7
Введите минимальный размер программ: 300

 4) 444          444             444             9532            Нет
```

4. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з перевантаженням методів.

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

5. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з класами та їх специфікаторами доступу, інкапсуляцією, константами.
Програма протестована, витоків пам’яті немає, виконується без помилок