**Новосибирский Государственный Технический Университет**

***Факультет автоматики и вычислительной техники***

**Лабораторная работа № 4  
по курсу «Программирование»**

**«Наследование»**

**Вариант №2**

Студента 2-го курса  
факультета АВТФ группы АБ-521  
Рудковский А.А.

Крошаков Е.В.

Проверил Бычков М.И.

Новосибирск-2016г

**Цель работы**: Изучить механизм наследования и возможности порождения новых типов данных на основе уже существующих классов.

**Задание**:

1. Создать дочерние классы
   1. BinNumber
   2. BlogEntry
2. Для класса BinNumber
   1. Добавить методы
      1. Проверка бинарной строки
3. Для класса BlogEntry
   1. Добавить поля
      1. Логическое поле завершения
      2. Время начала и конца работы
   2. Добавить методы
      1. Методы создания запланированных дел
      2. Метод получение списка дел

**Ход работы**:

* 1. Объявим дочерние классы после объявления основного класса
  2. **class BinNumber: public String**
  3. **class BlogEntry: public String**
  4. Объявим в классе BinNumber
  5. Добавим методы
     1. BinNumber(int maxlen, char\* strarr);

Метод возвращает **объект класса**

* 1. Объявим в классе BlogEntry
  2. Добавим поле  
     bool complete;Которое будет отражать логическое завершение строки
  3. Добавим поле

time\_t start, end;

Которое будет отображать время создания и завершение строки

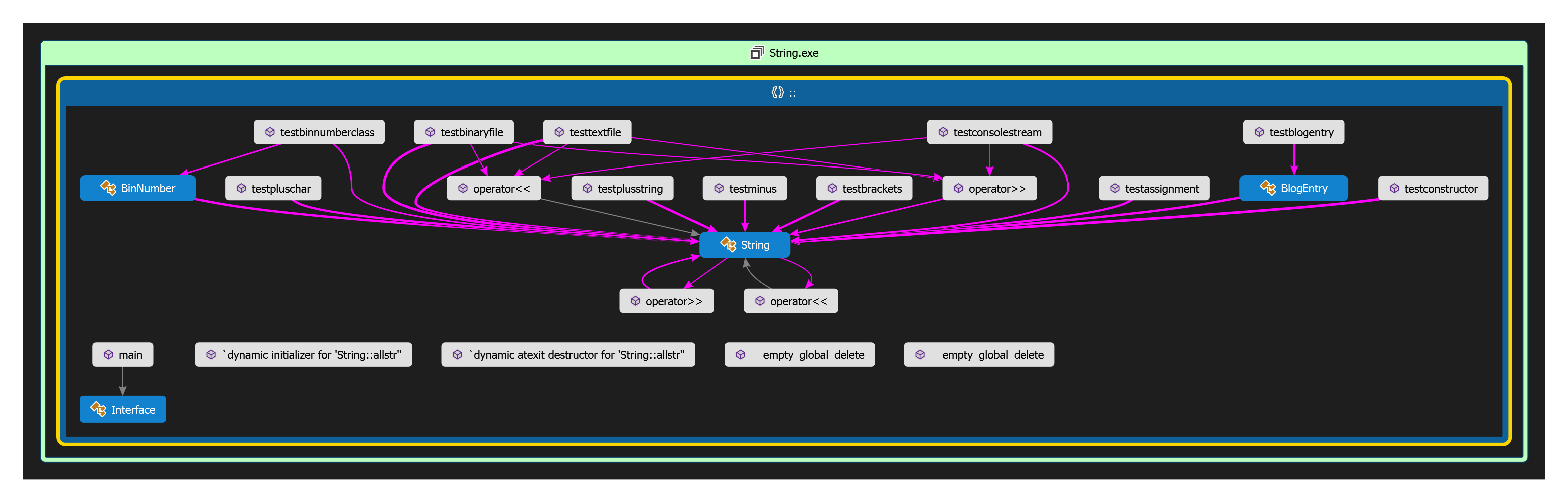
* 1. Добавим метод  
     void SetCompletionState(bool state);

Которое будет создавать условие выполнения

3.4 Добавим метод

bool GetCompletionState();

Который говорит что условие выполнилось

**Изображение иерархии классов **

**Вывод**: В ходе выполнения работы изучены принцип наследования. Получены практические навыки по созданию дочерних классов и перегрузке родительских методов.

Приложение 1. Заголовочный файл String.h

#pragma once

#include <list>

#include <ctime>

#include <iostream>

#define LEN 3 // выделяем три символа в начале строки для хранения реальной длины строки

using namespace std;

class String

{

protected:

char\* arr;

int maxlen;

time\_t create; // время создания строки

public:

static list<String\*> allstr; //Лист со всеми объектами класса в памяти

static void PrintAll(); //Вывод в консоль всех объектов

String();

String(int maxlength, char\* strarr);

String(const String& copy);

~String();

int RealLength();

int MaxLength();

char\* Substring(char\*); // нахождение подстроки

virtual void Print();

void RecalcLength();

//Перегрузка операций

String operator+(char\* concat);

String operator+(String concat);

String operator-(char\* substring);

String operator()(int position, int length);

String& operator=(String& assignment);

//Операторы ввода/вывода

friend ostream& operator<<(ostream& os, String& output);

friend istream& operator>>(istream& is, String& input);

friend ofstream& operator<<(ofstream& ofs, String& output);

friend ifstream& operator>>(ifstream& ifs, String& input);

void PrintToFile(char\* path, bool binary);

static String GetFromFile(char\* path, int length, bool binary);

};

Приложение 2. Файл исходного кода main.cpp

#include <iostream>

#include <conio.h>

#include <Windows.h>

#include "String.h"

#include "BinNumber.h"

#include "BlogEntry.h"

#include "Interface.h"

using namespace std;

//Демонстрация работы конструкторов, работа 1

void testconstructor()

{

char\* input;

String \*test, \*defaultconst, \*copy;

cin.ignore();

cout << "Enter string: ";

input = new char[60];

cin.getline(input, 60);

test = new String(60, input);

cout << "Real length: " << test->RealLength() << "\n";

cout << "Maximal length: " << test->MaxLength() << "\n";

cout << "Enter substring: ";

cin.getline(input, 60);

cout << "Substring by input: " << test->Substring(input) << "\n";

cout << "Full string: ";

test->Print();

defaultconst = new String();

copy = new String(\*test);

cout << "\n\n\nNow print all three strings: \n";

String::PrintAll();

}

//Функции демонстрации перегруженных операторов

void testpluschar()

{

char\* input;

cin.ignore();

cout << "Enter string 1: ";

input = new char[60];

cin.getline(input, 60);

String str = String(60, input);

cout << "Enter string 2: ";

cin.getline(input, 60);

str + input;

str.Print();

}

void testplusstring()

{

char\* input;

cin.ignore();

cout << "Enter string 1: ";

input = new char[60];

cin.getline(input, 60);

String str1 = String(60, input);

cout << "Enter string 2: ";

cin.getline(input, 60);

String str2 = String(60, input);

str1 + str2;

str1.Print();

}

void testminus()

{

char\* input;

cin.ignore();

cout << "Enter string 1: ";

input = new char[60];

cin.getline(input, 60);

String str = String(60, input);

cout << "Enter string 2: ";

cin.getline(input, 60);

cout << "Result: ";

str - input;

str.Print();

}

void testbrackets()

{

char \*temp = new char[256];

cout << "Enter string: ";

cin.ignore();

cin.getline(temp, 256);

String str(256, temp);

cout << "Substring (2, 2) = ";

String result = str(2, 2);

result.Print();

}

void testassignment()

{

char\*s = new char[256];

cout << "Enter first string: ";

cin.ignore();

cin.getline(s, 256);

String str(256, s);

cout << "Enter second string: ";

cin.getline(s, 256);

str = String(256, s);

cout << "First now is: ";

str.Print();

}

//тест работы

void testconsolestream()

{

cin.ignore();

String str(80, "");

cout << "Enter string: ";

cin >> str;

cout << "The string is: " << str;

}

void testtextfile()

{

cin.ignore();

String str(80, "");

cout << "Enter string: ";

cin >> str;

str.PrintToFile(".\\test.txt", false);

String fromfile;

fromfile = String::GetFromFile(".\\test.txt", str.MaxLength(), false);

cout << "The file now contains: " << fromfile;

}

void testbinaryfile()

{

cin.ignore();

String str(80, "");

cout << "Enter string: ";

cin >> str;

str.PrintToFile(".\\test.bin", true);

String fromfile;

fromfile = String::GetFromFile(".\\test.bin", str.MaxLength(), true);

cout << "The file now contains: " << fromfile;

}

void testbinnumberclass()

{

String\* bn = new BinNumber(5, "0101");

bn->Print();

String\* incorrectbn = new BinNumber(5, "0123");

incorrectbn->Print();

}

void testblogentry()

{

BlogEntry\* be = new BlogEntry(23, "Time to do something!");

be->Print();

be->SetCompletionState(true);

be->Print();

}

int main()

{

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

Interface items[] =

{

//id функция комментарий в консоли

{ 1, &testconstructor, "Test constructor" },

{ 2, &testpluschar, "Test operator+ with char\*" },

{ 3, &testplusstring, "Test operator+ with String" },

{ 4, &testminus, "Test operator- with char\*" },

{ 5, &testbrackets, "Test operator() with int,int" },

{ 6, &testassignment, "Test operator= with String\*" },

{ 7, &testconsolestream, "Test console I/O" },

{ 8, &testtextfile, "Test text file I/O" },

{ 9, &testbinaryfile, "Test binary file I/O" },

{ 10, &testbinnumberclass, "Test BinNumber class" },

{ 11, &testblogentry, "Test BlogEntry class" },

};

while (true)

{

int choice;

SetConsoleTextAttribute(hConsoleHandle, 10 | 0);

for each (Interface i in items)

cout << i.id << ". " << i.comment << "\n";

SetConsoleTextAttribute(hConsoleHandle, 12 | 0);

cout << "Choose menu item: ";

cin >> choice;

SetConsoleTextAttribute(hConsoleHandle, 9 | 0);

if (choice < 1 || choice > sizeof(items) / sizeof(Interface))

return 0;

items[choice - 1].invoke();

\_getch();

system("cls");

}

\_getch();

return 0;

}

Файл с наследуемым классом BinNumber.h

#pragma once

#include "String.h"

class BinNumber : public String

{

public:

explicit BinNumber() : String() {};

BinNumber(int maxlen, char\* strarr);

};

#include "BinNumber.h"

BinNumber::BinNumber(int maxlength, char\* strarr)

{

for (int i = 0; i < maxlength-1; i++)

if (strarr[i] != '0' && strarr[i] != '1')

{

std::cout << strarr << " is incorrect binary number.";

return;

}

arr = new char[maxlength];

maxlen = maxlength;

strcpy\_s(arr + 3, maxlength, strarr);

RecalcLength();

create = time(NULL);

allstr.insert(allstr.end(), this);

}

Файл с наследуемым классом BlogEntry.h

#pragma once

#include <ctime>

#include "String.h"

class BlogEntry : public String

{

private:

bool complete;

time\_t start, end;

public:

explicit BlogEntry() : String() {};

BlogEntry(int maxlen, char\* entry);

void SetCompletionState(bool state);

bool GetCompletionState();

void Print();

};

#include "BlogEntry.h"

BlogEntry::BlogEntry(int maxlength, char\* entry)

{

arr = new char[maxlength];

maxlen = maxlength;

strcpy\_s(arr + 3, maxlength, entry);

RecalcLength();

create = time(NULL);

time(&start);

complete = false;

allstr.insert(allstr.end(), this);

}

void BlogEntry::SetCompletionState(bool state)

{

complete = state;

time(&end);

}

bool BlogEntry::GetCompletionState()

{

return complete;

}

void BlogEntry::Print()

{

struct tm\* strt = localtime(&start);

cout << "Started at " << strt->tm\_hour << ":" << strt->tm\_min << " \"" << arr + LEN << "\" and ";

if (complete)

{

struct tm\* endt = localtime(&end);

cout << "completed at " << endt->tm\_hour << ":" << endt->tm\_min << ".\n";

}

else

cout << "is still incomplete.\n";

}

**Защита лабораторной работы**

**1. Что такое наследование? Объясните механизм наследования в С++.**

**Наследование - концепция объектно-ориентированного программирования, процесс приведения к одному значению одного или нескольких свойств разных объектов. Механизм заключается в том, что класс, объявленный наследником некого базового, имеет доступ к тем полям и методам, для которых это предусмотрено.**

**2. Какое бывает наследование?**

**Простое (от одного базового класса) и множественное (от нескольких базовых классов).**

**3. Как осуществляется простое или множественное наследование?**

**Простое наследование осуществляется следующим образом:**

**class baza{};**

**class naslednik: private baza{};**

**Множественное наследование осуществляется следующим образом:**

**class baza1{};**

**class baza2{};**

**class baza3{};**

**class naslednik: private baza1, protected baza2, public baza3{};**

**4. Какой класс называется базовым, а какой производным?**

**Базовым называется класс, от которого строится наследование; производным – тот, который наследует.**

**5. Как определяется доступ к членам базового класса членов производного класса?**

**Доступ к членам базового класса определяется спецификаторами public, protected и private**

**6. Что такое защищенные члены класса?**

**Защищённые члены класса – это такие члены, которые доступны только классам-наследникам.**

**7. Как влияют спецификаторы public, protect, private на статус наследования?**

**public: доступа к private-членам не имеет, в остальном наследуются с прежними спецификаторами**

**protected: доступа к private-членам не имеет, в остальном наследуются со спецификаторами protected**

**private: доступа к private-членам не имеет, в остальном наследуются со спецификаторами private**

**8. Какие члены класса наследуются.**

**Наследуются члены класса, не помеченные спецификатором private.**

**9. Вызов конструкторов при наследовании.**

**При наследовании сначала вызывается конструктор базового класса, а затем конструкторы производных классов.**

**10. Написание конструкторов в производном классе.**

**Конструкторы производного класса могут использовать конструкторы базового класса:**

**naslednik(data d, data e, data f):baza(d,e);**

**11. Какой класс называется абстрактным классом? Могут ли существовать экземпляры абстрактного класса?**

**Абстрактный класс – класс, в котором имеется хотя бы одна чистая виртуальная функция. Экземпляры абстрактного класса существовать не могут.**