

Отчет по л/р 1.

предмет «Языки и методы программирования»

Алешин Иван Сергеевич| ПИ 2-1 | 26.05.2021

# Предмет лабораторной работы

Создание программы, реализующей последовательный механизм наследования на основании пошагового расширения классов при помощи механизмов наследования в языке с++.

## Начальный Класс

В данной лабораторной работе у вас имеется начальный класс, описание которого задано первой часть вашего задания будет создать код реализующий этот класс и проверить его работоспособность на примере конструктора по умолчанию, когда вы убедитесь в работоспособности исходного класса, добавьте к публичным методам класса два метода-сеттера, пожалуйста. Эти методы должны задавать значения для h\_name и h\_age.

#include <iostream>

#include <string>

class Harbinger

{

private:

std::string h\_name;

int h\_age;

public:

Harbinger(std::string name = "", int age = 0)

: h\_name(name), h\_age(age)

{

}

std::string getName() const { return h\_name; }

int getAge() const { return h\_age; }

## };

После этого добавьте в класс Harbinger ещё один private-элемент типа std::string, назовите эту строку CatchPhrase. Напишите геттер и сеттер для неё, и добавьте новый конструктор с тремя аргументами, для каждого из которых будет значение по умолчанию.

Для отчета вставьте получившийся код ниже этой надписи, но до следующего раздела.

## Первое наследование

На втором этапе от класса Habringer необходимо унаследовать класс Prophet. Очень важно правильно выбрать модификатор наследования таким образом, чтобы все члены- переменные и методы класса предка не изменили своего модификатора доступа, будучи унаследованными в дочернем классе. (обратите внимание на то, как разные типы модификаторов наследования влияют на модификаторы доступа). В классе Prophet, в свою очередь, нужно создать массив из 3 строк, назвать его MagnumOpus , и написать геттер и сеттер для этого массива. Также у класса Phophet должны быть собственные члены-переменные (с геттерами и сеттерами) ph\_name(std::string), ph\_age(int) и ph\_call(std::string). В ph\_call всегда должна храниться самая длинная строка из массива MagnumOpus, и, как только какая-то строка из массива MagnumOpus после переопределения становится самой длинной, в ph\_call тут же должна записываться эта строка. Длину всех строк ограничить 25 символами, при вводе в сеттере проверять чтобы задаваемая строка не была слишком длинной. Проверьте при помощи создания объекта класса Phophet. (Код с комментариями вставить ниже для отчета):

## Второе наследование

Слово Harbinger переводится как «Предвестник», слово Prophet это «Пророк». Во время второго наследования мы создадим класс DeusEx («Бог из…»). Класс DeusEx будет публично наследовать класс Prophet. У нового класса появятся элементы-переменные gd\_name(std::string), gd\_year(int). Эти два элемента единожды задаются при помощи конструктора как значения по умолчанию. Переменная gd\_year хранит в себе четырёхзначное число, отражающая какой-то год в диапазоне от 0000 до 9999. В класс DeusEx нужно добавить публичный метод, который бы последовательно строка за строкой выводил в консоль:

(из класса Harbinger):

«*h\_name* в возрасте *h\_age* говорил, что *CatchPhrase.*»

(две строки кроме ph\_call из класса Prophet)

«Первое пророчество: *MagnumOpus[1].*»

«Второе пророчество: *MagnumOpus[2].*»

(ph\_call из класса Prophet)

«Главное пророчество: *ph\_call.*»

(ph\_name и ph\_age из класса Prophet)

«Так говорил *ph\_name* в возрасте *ph\_age.*»

(gd\_name и gd\_year из класса DeusEx)

«В год gd\_year пришёл gd\_name, которого предрекали h\_name и ph\_name».

(Приведите получившийся код ниже для отчета, пожалуйста)

## Последнее наследование

После того как мы создали предвестника, пророка с заповедями и бога перейдем к верующему. Создайте класс Believer, унаследуйте его приватно от DeusEx, добавьте члены-переменные bv\_name(std::string), bv\_age(int). Геттер и сеттер для них обоих. Конструктор без значений по умолчания, но с определением значения переменных в теле конструктора через два аргумента. И один публичный метод с выводом «Верующий *bv\_name* молится *gd\_name* именем *ph\_name*. C *gd\_year* и по сей день.»

(Ниже приведите полный код со всеми классами и функцией main, которая позволит продемонстрировать функциональность созданных классов.)

И, пожалуйста, не забывайте про комментарии.

#include <iostream>

#include <string>

using namespace std;

class Harbinger

{

private:

string h\_name;

int h\_age;

string CatchPhrase;

public:

Harbinger(string name = "Иисус", int age = 2000)

: h\_name(name), h\_age(age)

{

}

Harbinger(string name, int age, string phrase) : h\_name(name), h\_age(age), CatchPhrase(phrase) {}

string getName() const { return h\_name; }

int getAge() const {

return h\_age;

}

void setAge(int age) {

h\_age = age;

}

void setName(string name) {

h\_name = name;

}

void setPhrase(string phrase) {

CatchPhrase = phrase;

}

string getPhrase() const { return CatchPhrase; }

};

class Prophet : public Harbinger {

private:

string\* MagnumOpus = new string[3];

string ph\_name;

int ph\_age;

string ph\_call;

public:

Prophet(string name = "Александр", int age = 20, string call = "") : ph\_name(name), ph\_age(age), ph\_call(call) {

setMagnum();

}

void setMagnum(string\* arr = NULL) {

if (arr) {

for (int i = 0; i < 3; i++)

{

if (arr[i].length() > 25) cout << "Ошибка длины строки!";

return;

}

MagnumOpus = arr;

setCall();

}

else {

MagnumOpus[0] = "АБВ";

MagnumOpus[1] = "АБ";

MagnumOpus[2] = "А";

setCall();

}

}

string\* getMagnum() {

return MagnumOpus;

}

void setCall() {

string max = MagnumOpus[0];

if (MagnumOpus[1].length() > max.length())

max = MagnumOpus[1];

if (MagnumOpus[2].length() > max.length())

max = MagnumOpus[2];

if (ph\_call.length() < max.length())

ph\_call = max;

}

string getCall() { return ph\_call; }

void setAge(int age) { ph\_age = age; }

int getAge() { return ph\_age; }

void setName(string name) { ph\_name = name; }

string getName() { return ph\_name; }

};

class DeusEx : public Prophet {

private:

string gd\_name;

int gd\_year;

Harbinger hb;

Prophet pr;

public:

DeusEx() {

gd\_year = 2021;

gd\_name = "Бог";

}

void print() {

cout << hb.getName() << " в возрасте " << hb.getAge() << " говорил, что " << hb.getPhrase() << endl;

cout << endl;

cout << "Первое пророчество: " << (pr.getMagnum())[1] << endl;

cout << "Второе пророчество: " << (pr.getMagnum())[2] << endl;

cout << "Главное пророчество: " << (pr.getCall()) << endl;

cout << endl;

cout << "Так говорил " << pr.getName() << " в возрасте " << pr.getAge() << endl;

cout << "В год " << gd\_year << " пришел " << gd\_name << ", которого предрекали " << hb.getName() << " и " << pr.getName() << endl;

}

string getName() { return gd\_name; }

int getYear() { return gd\_year; }

};

class Believer : private DeusEx {

private:

string bv\_name;

int bv\_age;

DeusEx de;

Prophet ph;

public:

Believer(string name, int age) {

bv\_name = name;

bv\_age = age;

}

void setName(string name) { bv\_name = name; }

void setAge(int age) { bv\_age = age; }

string getName() { return bv\_name; }

int getAge() { return bv\_age; }

void print() {

cout << endl;

cout << "Верующий " << bv\_name << " молится " << de.getName() << " именем " << ph.getName() <<

" с " << de.getYear() << endl;

}

};

int main() {

setlocale(LC\_ALL, "Russian");

const string name = "Кирилл";

const int age = 200;

DeusEx obj;

obj.print();

Believer obj2(name, age);

obj2.print();

return 0;

## }