|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА ИУ8 “КОМПЬЮТЕРНАЯ БЕЗОПАСНОСТЬ”

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

Студент Лустин Павел Евгеньевич

*(*Фамилия И.О)

Группа\_\_ИУ8-24\_\_\_\_\_\_\_\_

Студент  **\_\_\_\_\_\_14.03\_\_\_\_\_\_\_\_\_**  Лустин \_\_ \_\_

*дата (*Фамилия И.О)

Преподаватель **\_\_\_\_\_\_14.03\_\_\_\_\_\_\_**  Барыкин Дмитрий

*дата (*Фамилия И.О)*.*

*2024 г.*

Цель работы

Цель работы состоит в овладении навыками разработки программ на языке Си++, использующих возможности наследования классов для решения различных задач.

Условие задачи(вариант 14)

Создать базовый класс – простой счет в банке. Компоненты класса: поля: ФИО владельца, начальная сумма счета, ставка вклада (проценты в год), время существования вклада в годах, статус доступа полей *protected*;

конструктор для инициализации полей;

функция для вычисления суммы на счете с учетом начисленных процентов за время существования вклада;

функция для печати параметров счета.

Создать производный класс – привилегированный счет.

Дополнительные поля: процент кредита предоставляемому по счету (проценты от доступной на счете суммы с учетом времени существования вклада).

Переопределенная функция для вычисления суммы на счете с учетом доступного кредита.

Переопределенная функция для печати параметров счета.

Внутри переопределенных функций вызывать соответствующие функции из базового класса.

Создать по 1 объекту каждого из классов. Показать вызов созданных функций. При переопределении функций обеспечить и продемонстрировать два варианта: статический полиморфизм и динамический полиморфизм.

Текст программы с комментариями

#include <iostream>

#include <string>

using namespace std;

class Bank //базовый класс

{

protected:

string FIO;

int sum;

int deposit\_rate;

int years;

public:

Bank() //пустой конструктор

{

FIO = "";

sum = 0;

deposit\_rate = 0;

years = 0;

}

Bank(string FIO, int sum, int deposit\_rate, int years) //конструктор с начальными значениями

{

this->FIO = FIO;

this->sum = sum;

this->deposit\_rate = deposit\_rate;

this->years = years;

}

double sum\_after\_a\_time() //функция подсчёта конечной суммы вклада

{

double temp = sum;

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

{

temp += temp \* (deposit\_rate \* 0.01); //начисление процентов

}

return temp;

}

void print() //функция печати

{

cout << "ФИО: " << FIO << endl;

cout << "Начальная сумма счёта: " << sum << endl;

cout << "Ставка вклада(в %): " << deposit\_rate << endl;

cout << "Время существования вклада(в годах): " << years << endl;

}

void virtual print1() //функция печати для демонстрации динамического полиморфизма

{

cout << "ФИО: " << FIO << endl;

cout << "Начальная сумма счёта: " << sum << endl;

cout << "Ставка вклада(в %): " << deposit\_rate << endl;

cout << "Время существования вклада(в годах): " << years << endl;

}

~Bank() {} //деструктор

};

class preferred\_Bank : public Bank //производный класс

{

protected:

int credit;

public:

preferred\_Bank(): Bank() //пустой конструктор производного класса

{

credit = 0;

}

preferred\_Bank(string FIO, int sum, int deposit\_rate, int years, int credit) : Bank(FIO, sum, deposit\_rate, years) //конструктор производного класса с начальными значениями

{

this->credit = credit;

}

double sum\_after\_a\_time() //функция подсчёта конечной суммы вклада с учётом процентов кредита

{

double temp = Bank::sum\_after\_a\_time(); //берём конечную сумму вклада

temp += temp \* (credit \* 0.01); //начисляем процент кредита

return temp;

}

void print() //функция печати производного класса

{

Bank::print();

cout << "Процент кредита(в %): " << credit << endl;

}

void print1() //функция печати для демонстрации динамического полиморфизма

{

Bank::print1();

cout << "Процент кредита(в %): " << credit << endl;

}

~preferred\_Bank() {} //деструктор производного класса

};

int main()

{

setlocale(LC\_ALL, "Rus");

Bank\* contrib = new Bank("FIO", 1000, 10, 3); //создание указателя на базовый класс с начальными значениями

double Sum1 = contrib->sum\_after\_a\_time(); //сумма вклада после начисления всех %

contrib->print(); //демонстрация статического полиморфизма

cout << "Сумма по окончании вклада (обычный счёт): " << Sum1 << endl << endl;

preferred\_Bank\* contrib\_pref = new preferred\_Bank("FIO", 1000, 10, 3, 20); //создание указателя на производный класс с начальными значениями

double Sum2 = contrib\_pref->sum\_after\_a\_time(); //сумма вклада с кредитом после начисления всех %

contrib\_pref->print1(); //демонстрация динамического полиморфизма

cout << "Сумма по окончании вклада (привилегированный счёт): " << Sum2 << endl;

return 0;

}

Вывод результатов работы программы

ФИО: FIO

Начальная сумма счёта: 1000

Ставка вклада(в %): 10

Время существования вклада(в годах): 3

Сумма по окончании вклада (обычный счёт): 1331

ФИО: FIO

Начальная сумма счёта: 1000

Ставка вклада(в %): 10

Время существования вклада(в годах): 3

Процент кредита(в %): 20

Сумма по окончании вклада (привилегированный счёт): 1597.2

*Вывод*

*Я* овладел навыками разработки программ на языке Си++, использующих возможности наследования классов для решения различных задач.