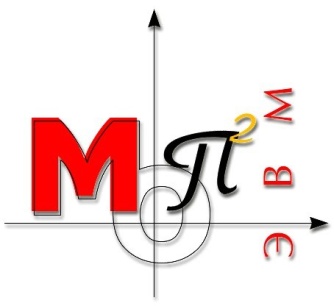
Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждения высшего образования   
«Южный федеральный университет»

Инженерно-технологическая академия

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**

Лабораторная работа №2

по дисциплине

"ООП"

на тему

"Наследование в С++"

**Выполнил:**

студент группы КТбо2-6

Пустовой А.В

**Проверил:**

Таганрог, 2020

**Вариант №16**

**Задание:**

Создать класс Pair (пара значений) с методами, реализующими арифметические операции сложения и вычитания. На его основе реализовать классы Money (деньги) и Complex (комплексное число). В классе Money денежная сумма представляется в виде двух целых, в которых хранятся рубли и копейки соответственно.

Спецификация классов:

class Pair

{

public:

Pair() :\_firstInt(0), \_secondInt(0) {}

Pair(const int firstInt, const int secondInt) :\_firstInt(firstInt), \_secondInt(secondInt) {}

void SetPair(const int firstInt, const int secondInt);

int GetFirstInt() const;

int GetSecondInt() const;

virtual Pair\* Plus(const Pair& source);

virtual Pair\* Minus(const Pair& source);

virtual string GetPair() const = 0;

protected:

int \_firstInt;

int \_secondInt;

};

class Money : public Pair

{

public:

Money() : Pair() {}

Money(int \_firstInt, int \_secondInt);

virtual Money\* Plus(const Pair& source) override;

virtual Money\* Minus(const Pair& source) override;

virtual string GetPair() const override;

};

class Complex : public Pair

{

public:

Complex() : Pair() {}

Complex(int \_firstInt, int \_secondInt) : Pair(\_firstInt, \_secondInt) {}

virtual string GetPair() const override;

};

#include "Pair.h"

#include "Complex.h"

#include "Money.h"

class ConsoleInteractor

{

public:

ConsoleInteractor();

~ConsoleInteractor();

void Init() const;

private:

int GetIndex(const string& name) const;

void Minus() const;

void Plus() const;

void Help() const;

void CreatePair() const;

void DeletePair(const string& name) const;

void PrintPair() const;

void DelPair() const;

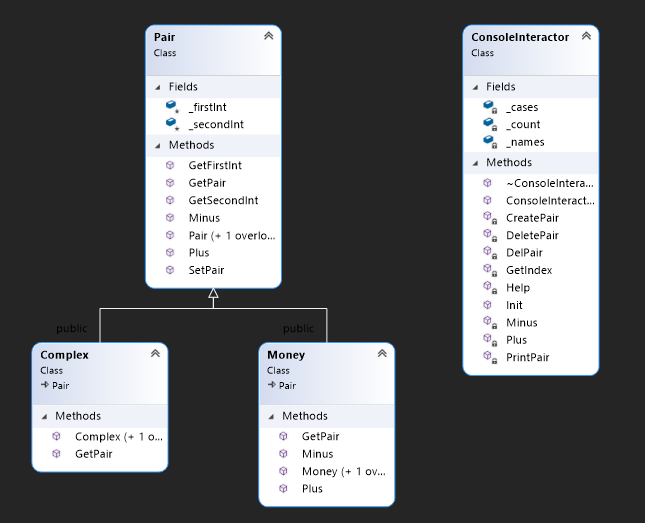
Pair\*\* \_cases = nullptr;

string\* \_names = nullptr;

int \_count = 0;

};

**Диаграмма классов**

****

**Листинг кода**

main.cpp

#include "ConsoleInteractor.h"

#include <iostream>

int main()

{

ConsoleInteractor init;

init.Init();

return 0;

}

Pair.h

#pragma once

#include <string>

using std::string;

class Pair

{

public:

Pair() :\_firstInt(0), \_secondInt(0) {}

Pair(const int firstInt, const int secondInt) :\_firstInt(firstInt), \_secondInt(secondInt) {}

void SetPair(const int firstInt, const int secondInt);

int GetFirstInt() const;

int GetSecondInt() const;

virtual Pair\* Plus(const Pair& source);

virtual Pair\* Minus(const Pair& source);

virtual string GetPair() const = 0;

protected:

int \_firstInt;

int \_secondInt;

};

Pair.cpp

#include "Pair.h"

void Pair::SetPair(const int firstInt, const int secondInt)

{

\_firstInt = firstInt;

\_secondInt = secondInt;

}

int Pair::GetFirstInt() const

{

return \_firstInt;

}

int Pair::GetSecondInt() const

{

return \_secondInt;

}

Pair\* Pair::Plus(const Pair& source)

{

\_firstInt += source.\_firstInt;

\_secondInt += source.\_secondInt;

return this;

}

Pair\* Pair::Minus(const Pair& source)

{

\_firstInt += source.\_firstInt;

\_secondInt += source.\_secondInt;

return this;

}

Money.h

#pragma once

#include "Pair.h"

class Money : public Pair

{

public:

Money() : Pair() {}

Money(int \_firstInt, int \_secondInt);

virtual Money\* Plus(const Pair& source) override;

virtual Money\* Minus(const Pair& source) override;

virtual string GetPair() const override;

};

Money.cpp

#include "Money.h"

const int MAX\_SECOND = 100;

Money::Money(int \_firstInt, int \_secondInt) : Pair(\_firstInt, \_secondInt)

{

if (\_secondInt > MAX\_SECOND)

{

\_firstInt += \_secondInt / MAX\_SECOND;

\_secondInt = \_secondInt % MAX\_SECOND;

}

if (\_firstInt < 0 || \_secondInt < 0)

{

throw "LESS\_THAN\_ZERO";

}

}

Money\* Money::Plus(const Pair& source)

{

int firstInt = GetFirstInt();

int secondInt = source.GetSecondInt();

\_firstInt += firstInt + (\_secondInt + secondInt) / MAX\_SECOND;

\_secondInt = (\_secondInt + secondInt) % MAX\_SECOND;

return this;

}

Money\* Money::Minus(const Pair& source)

{

int firstInt = source.GetFirstInt();

int secondInt = source.GetSecondInt();

\_firstInt = (\_firstInt > firstInt) ? \_firstInt - firstInt : 0;

if (\_secondInt >= secondInt)

{

\_secondInt -= secondInt;

}

else if (\_firstInt)

{

\_firstInt--;

\_secondInt = \_secondInt - secondInt + MAX\_SECOND;

}

else

{

\_secondInt = 0;

}

return this;

}

string Money::GetPair() const

{

string result = std::to\_string(\_firstInt) + "." + std::to\_string(\_secondInt) + " rub";

return result;

}

Complex.h

#pragma once

#include "Pair.h"

class Complex : public Pair

{

public:

Complex() : Pair() {}

Complex(int \_firstInt, int \_secondInt) : Pair(\_firstInt, \_secondInt) {}

virtual string GetPair() const override;

};

Complex.cpp

#include "Complex.h"

string Complex::GetPair() const

{

return std::to\_string(\_firstInt) + " + " + std::to\_string(\_secondInt) + "i";

}

ConsoleInteractor.h

#include "Pair.h"

#include "Complex.h"

#include "Money.h"

class ConsoleInteractor

{

public:

ConsoleInteractor();

~ConsoleInteractor();

void Init() const;

private:

int GetIndex(const string& name) const;

void Minus() const;

void Plus() const;

void Help() const;

void CreatePair() const;

void DeletePair(const string& name) const;

void PrintPair() const;

void DelPair() const;

Pair\*\* \_cases = nullptr;

string\* \_names = nullptr;

int \_count = 0;

};

ConsoleInteractor.cpp

#include <iostream>

#include "ConsoleInteractor.h"

using std::cin;

using std::cout;

ConsoleInteractor::~ConsoleInteractor()

{

delete[] \_names;

for (int i = 0; i < \_count; i++)

{

delete \_cases[i];

}

delete[] \_cases;

}

ConsoleInteractor::ConsoleInteractor()

{

cout << "Enter space for Pairs\n>> ";

cin >> \_count;

\_names = new string[\_count];

\_cases = new Pair\*[\_count];

}

int ConsoleInteractor::GetIndex(const string& name) const

{

for (int i = 0; i < \_count; i++)

{

if (name == \_names[i])

{

return i;

}

}

return -1;

}

void ConsoleInteractor::Minus() const

{

string name, name2;

cout << "Enter name of first pair\n>> ";

cin >> name;

cout << "Enter name of second pair\n>> ";

cin >> name2;

int indexName1 = GetIndex(name);

int indexName2 = GetIndex(name2);

if (indexName1 != -1 && indexName2 != -1)

{

cout << \_cases[indexName1]->GetPair() << "\n-\n";

cout << \_cases[indexName2]->GetPair() << "\n";

cout << (\_cases[indexName1]->Minus(\*\_cases[indexName2]))->GetPair() << "\n";

}

else

{

cout << "Wrond name\n";

}

}

void ConsoleInteractor::Plus() const

{

string name, name2;

cout << "Enter name of first pair\n>> ";

cin >> name;

cout << "Enter name of second pair\n>> ";

cin >> name2;

int indexName1 = GetIndex(name);

int indexName2 = GetIndex(name2);

if (indexName1 != -1 && indexName2 != -1)

{

cout << \_cases[indexName1]->GetPair() << "\n+\n";

cout << \_cases[indexName2]->GetPair() << "\n";

cout << (\_cases[indexName1]->Minus(\*\_cases[indexName2]))->GetPair() << "\n";

}

else

{

cout << "Wrond name\n";

}

}

void ConsoleInteractor::Help() const

{

cout << "\"1\" - Create new pair\n\"2\" - Delete pair\n\"3\" - Help\n\"4\" - Plus\n\"5\" - Minus\n\

\"6\" - Print pair\n\"7\" - Exit\n";

}

void ConsoleInteractor::CreatePair() const

{

int firstNum, secondNum;

cout << "Enter first num:\n>> ";

cin >> firstNum;

cout << "Enter second num:\n>> ";

cin >> secondNum;

cout << "Enter name of Pair:\n>> ";

string name;

cin >> name;

int type;

cout << "Is it Complex or Money (1 or 2):\n>> ";

cin >> type;

bool isEnoughtSpace = false;

int index;

for (int i = 0; i < \_count && !isEnoughtSpace; i++)

{

if (\_names[i].empty())

{

isEnoughtSpace = true;

index = i;

}

}

if (isEnoughtSpace)

{

Pair\* newElement = nullptr;

if (type == 1)

{

newElement = new Complex(firstNum, secondNum);

}

else if (type == 2)

{

try

{

newElement = new Money(firstNum, secondNum);

}

catch (const char\*)

{

cout << "Some part of pair was less than 0\n";

}

}

\_names[index] = name;

\_cases[index] = newElement;

}

else

{

cout << "You don't have enought space, delete something\n";

}

}

void ConsoleInteractor::DeletePair(const string& name) const

{

int index = GetIndex(name);

if (index == -1)

{

cout << "Can't find pair with this name\n";

}

else

{

\_names[index] = "";

delete \_cases[index];

\_cases[index] = nullptr;

}

}

void ConsoleInteractor::PrintPair() const

{

string name;

cout << "Enter name of pair to print\n>> ";

cin >> name;

int index = GetIndex(name);

if (index != -1)

{

cout << \_cases[index]->GetPair() << "\n";

}

else

{

cout << "Wrong name\n";

}

}

void ConsoleInteractor::DelPair() const

{

string name;

cout << "Enter name of pair to delete\n>> ";

cin >> name;

DeletePair(name);

cout << "Complete\n";

}

void ConsoleInteractor::Init() const

{

int code, arg; string name, name2;

while (true)

{

cout << "Enter \"3\" - Help\n";

cout << ">> ";

cin >> code;

if (code == 1)

{

CreatePair();

}

else if (code == 2)

{

DelPair();

}

else if (code == 3)

{

Help();

}

else if (code == 4)

{

Plus();

}

else if (code == 5)

{

Minus();

}

else if (code == 6)

{

PrintPair();

}

else if (code == 7)

{

break;

}

}

}