**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 2**

Тема: Перегрузка операций. Литералы

Студент: Бутырев Даниил Вячеславович

Группа: 80-206

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

<https://github.com/SteepturN/oop_exercise_02>

1. Постановка задачи

Создать класс BritishMoney для работы с денежными суммами в старой британской система.

Сумма денег должна быть представлено тремя полями:

-типа unsigned long long для фунтов стерлингов,

-типа unsigned char – для шиллингов,

-unsigned char – для пенсов (пенни).

Реализовать

сложение сумм,

вычитание,

деление сумм,

деление суммы на дробное число,

умножение на дробное число и

операции сравнения.

2. Описание программы

Программа состоит из 3 файлов:

1. main.cpp - здесь находится пользовательский интерпретатор, происходят манипуляции с объектами класса BritishMoney и их результаты отправляются на стандартный поток вывода,
2. Read\_input.cpp - здесь происходит ввод данных,
3. BritishMoney.cpp - здесь реализованы: класс BritishMoney, различные операции над объектами класса BritishMoney, например +, - и тд.

3. Набор тестов

a 12 3 3 1 3 4 - складываем 2 суммы британских денег

a 123421421 3 4 5 1 24 - складываем 2 суммы британских денег

a 123 4 5 123 14 15 - складываем 2 суммы британских денег

s 12342134 1 1 12344123 1 1 - вычитаем 2 суммы британских денег

s 0 1 1 124124124 0 0 - вычитаем 2 суммы британских денег

s 0 0 0 1 20 20 - вычитаем 2 суммы британских денег

d 123 1 1 1234 8 0 - делим 2 суммы британских денег

d 123 9 9 - проверка на реагирование на некорректный ввод

a -1 0 0 1 2 3 - складываем 2 суммы британских денег

a 123 1 1 0 0 0 - складываем 2 суммы британских денег

d 1 1 1 1 1 1 - делим 2 суммы британских денег

dbn 1 1 1 0 - делим 1 сумму британских денег на число

dbn 100 100 100 100 - делим 1 сумму британских денег на число

m 100 100 100 100 - умножаем 1 сумму британских денег на число

lth 100 100 100 100 100 100 - сравниваем 2 суммы британских денег

1. Результаты выполнения тестов

You can use

--add: a (funt1) (shilling1) (penny1) (funt2) (shilling2) (penny2)

--subtraction: s (funt1) (shilling1) (penny1) (funt2) (shilling2) (penny2)

--division of sums: d (funt1) (shilling1) (penny1) (funt2) (shilling2) (penny2)

--sum1 < sum2: lth (funt1) (shilling1) (penny1) (funt2) (shilling2) (penny2)

--divide by number: dbn (funt1) (shilling1) (penny1) (real number)

--multiplication by number: m (funt1) (shilling1) (penny1) (real number)

--exit

фунты = 12, шиллинги = 3, пенсы = 3 a фунты = 1, шиллинги = 3, пенсы = 4 = фунты = 13, шиллинги = 6, пенсы = 7

фунты = 123421421, шиллинги = 3, пенсы = 4 a фунты = 5, шиллинги = 3, пенсы = 0 = фунты = 123421426, шиллинги = 6, пенсы = 4

фунты = 123, шиллинги = 4, пенсы = 5 a фунты = 123, шиллинги = 15, пенсы = 3 = фунты = 246, шиллинги = 19, пенсы = 8

фунты = 12342134, шиллинги = 1, пенсы = 1 s фунты = 12344123, шиллинги = 1, пенсы = 1 = фунты = 0, шиллинги = 0, пенсы = 0

фунты = 0, шиллинги = 1, пенсы = 1 s фунты = 124124124, шиллинги = 0, пенсы = 0 = фунты = 0, шиллинги = 0, пенсы = 0

фунты = 0, шиллинги = 0, пенсы = 0 s фунты = 2, шиллинги = 1, пенсы = 8 = фунты = 0, шиллинги = 0, пенсы = 0

фунты = 123, шиллинги = 1, пенсы = 1 d фунты = 1234, шиллинги = 8, пенсы = 0 = 0.0996929

wrong input

You can use

--add: a (funt1) (shilling1) (penny1) (funt2) (shilling2) (penny2)

--subtraction: s (funt1) (shilling1) (penny1) (funt2) (shilling2) (penny2)

--division of sums: d (funt1) (shilling1) (penny1) (funt2) (shilling2) (penny2)

--sum1 < sum2: lth (funt1) (shilling1) (penny1) (funt2) (shilling2) (penny2)

--divide by number: dbn (funt1) (shilling1) (penny1) (real number)

--multiplication by number: m (funt1) (shilling1) (penny1) (real number)

--exit

wrong input

wrong input

фунты = 123, шиллинги = 1, пенсы = 1 a фунты = 0, шиллинги = 0, пенсы = 0 = фунты = 123, шиллинги = 1, пенсы = 1

фунты = 1, шиллинги = 1, пенсы = 1 d фунты = 1, шиллинги = 1, пенсы = 1 = 1

фунты = 1, шиллинги = 1, пенсы = 1 dbn 0 = division by 0 is forbidden

фунты = 105, шиллинги = 8, пенсы = 4 dbn 100 = фунты = 1, шиллинги = 1, пенсы = 1

фунты = 105, шиллинги = 8, пенсы = 4 m 100 = фунты = 10568, шиллинги = 6, пенсы = 7

фунты = 105, шиллинги = 8, пенсы = 4 lth фунты = 105, шиллинги = 8, пенсы = 4 = 0

2. Листинг программы

main.cpp

#include <iostream>

#include <string>

#include "BritishMoney.cpp"

#include "Read\_input.cpp"

#include <cstdio>

#include <set>

int main(){

const char a[] = "You can use\n\

--add: a (funt1) (shilling1) (penny1) (funt2) (shilling2) (penny2)\n\

--subtraction: s (funt1) (shilling1) (penny1) (funt2) (shilling2) (penny2)\n\

--division of sums: d (funt1) (shilling1) (penny1) (funt2) (shilling2) (penny2)\n\

--sum1 < sum2: lth (funt1) (shilling1) (penny1) (funt2) (shilling2) (penny2)\n\n\

--divide by number: dbn (funt1) (shilling1) (penny1) (real number)\n\

--multiplication by number: m (funt1) (shilling1) (penny1) (real number)\n\n\

--exit\n";

long double input\_real\_number(0.0);

unsigned long long funt1n(0), funt2n(0); unsigned int shil1n(0), shil2n(0), penny1n(0), penny2n(0);

int conv\_penny(0), conv\_shil(0);

BritishMoney bm1(0,0,0);

BritishMoney bm2(0,0,0);

char ch(' ');

char command[20];

std::set<std::string> valid\_commands = {"a", "s", "d", "dbn", "m", "lth", "exit"};

std::cout << a;

do {

bool exit = false;

do{

read\_return\_t answer = get\_command(valid\_commands, command);

switch(answer) {

case END\_OF\_FILE: return 0;

case END\_OF\_LINE: continue;

case VALID\_INPUT: exit = true; break;

case INVALID\_INPUT:

do ch=getchar(); while((ch != EOF) && (ch != '\n'));

std::cout << "wrong input" << std::endl;

if(ch == EOF) return 0;

else break;

}

} while(!exit);

if(static\_cast<std::string>(command) == "exit") return 0;

if((get\_value<unsigned long long>(funt1n) != VALID\_INPUT) || (get\_value<unsigned int>(shil1n) != VALID\_INPUT) || (get\_value<unsigned int>(penny1n) != VALID\_INPUT)) {

do ch = getchar(); while((ch != EOF) && (ch != '\n'));

std::cout << "wrong input" << std::endl;

if(ch == EOF) return 0;

continue;

}

if((static\_cast<std::string>(command) == "dbn") || (static\_cast<std::string>(command) == "m")) {

if((get\_value<long double>(input\_real\_number) != VALID\_INPUT) || (input\_real\_number < 0)) {

do ch=getchar(); while((ch != EOF) && (ch != '\n'));

std::cout << "wrong input" << std::endl;

if(ch == EOF) return 0;

continue;

}

} else if((get\_value<unsigned long long>(funt2n) != VALID\_INPUT) || (get\_value<unsigned int>(shil2n) != VALID\_INPUT) || (get\_value<unsigned int>(penny2n) != VALID\_INPUT)) {

do ch = getchar(); while((ch != EOF) && (ch != '\n'));

std::cout << "wrong input" << std::endl;

if(ch == EOF) return 0;

continue;

}

conv\_penny = (penny1n)/12;

bm1.penny = (penny1n)%12;

conv\_shil = (shil1n+conv\_penny)/20;

bm1.shil = (shil1n+conv\_penny)%20;

bm1.funt = funt1n+conv\_shil;

conv\_penny = (penny2n)/12;

bm2.penny = (penny2n)%12;

conv\_shil = (shil2n+conv\_penny)/20;

bm2.shil = (shil2n+conv\_penny)%20;

bm2.funt = funt2n+conv\_shil;

bm1.print();

std::cout << '\t' << command << '\t';

if((static\_cast<std::string>(command) == "dbn") || (static\_cast<std::string>(command) == "m")) {

std::cout << input\_real\_number << '\t';

} else {

bm2.print();

}

std::cout << '\t' << '=' << '\t';

if((static\_cast<std::string>(command) == "dbn") || (static\_cast<std::string>(command) == "m")) {

if((input\_real\_number == 0) && (static\_cast<std::string>(command) == "dbn"))

std::cout << "division by 0 is forbidden";

else bm1.exec(command, input\_real\_number).print();

} else if (static\_cast<std::string>(command) == "lth") {

std::cout << (bm1<bm2);

} else if (static\_cast<std::string>(command) == "d") {

std::cout << (bm1/bm2);

} else {

bm1.exec(command, bm2).print();

}

std::cout << std::endl;

} while(true);

return 0;

}

BritishMoney.cpp

#include <cmath>

#include <iostream>

//1 фунт = 20 шиллингов, 1 шиллинг = 12 пенни.

class BritishMoney{

public:

BritishMoney(unsigned long long funt, unsigned char shil, unsigned char penny)

: funt(funt), shil(shil), penny(penny) {};

//1 фунт = 20 шиллингов, 1 шиллинг = 12 пенни

BritishMoney operator+(const BritishMoney& bm) const noexcept{

int conv\_penny = (penny+bm.penny)/12;

int \_penny = (penny+bm.penny)%12;

int conv\_shil = (shil+bm.shil+conv\_penny)/20;

int \_shil = (shil+bm.shil+conv\_penny)%20;

unsigned long long \_funt = funt+bm.funt+conv\_shil;

return BritishMoney(\_funt, \_shil, \_penny);

}

BritishMoney operator-(const BritishMoney& bm) const noexcept{

int k = bm < (\*this) ? 1 : 0;

unsigned long long \_funt = (funt-bm.funt)\*k;

int \_shil = (shil-bm.shil)\*k;

int \_penny = (penny-bm.penny)\*k;

if(\_penny < 0) {\_shil-=1; \_penny+=12;}

if(\_shil < 0) {\_funt-=1; \_shil+=20;}

return BritishMoney(\_funt, \_shil, \_penny);

}

double operator/(const BritishMoney& bm) const noexcept{

return (funt+shil/12.0+penny/240.0)/(bm.funt+bm.shil/12.0+bm.penny/240.0);

}

BritishMoney operator/(const long double& num) const noexcept{

if(num == 0) return BritishMoney(0, 0, 0);

long double res = (funt+shil/12.0L+penny/240.0L)/num;

unsigned long long \_funt = trunc(res);

unsigned char \_shil = static\_cast<unsigned char>(trunc((res-\_funt)\*20));

unsigned char \_penny = static\_cast<unsigned char>(trunc(((res-\_funt)\*20-\_shil)\*12));

return BritishMoney(\_funt, \_shil, \_penny);

}

BritishMoney operator\*(const long double& num) const noexcept{

long double res = (funt+shil/12.0L+penny/240.0L)\*num;

unsigned long long \_funt = trunc(res);

unsigned char \_shil = static\_cast<unsigned char>(trunc((res-\_funt)\*20));

unsigned char \_penny = static\_cast<unsigned char>(trunc(((res-\_funt)\*20-\_shil)\*12));

return BritishMoney(\_funt, \_shil, \_penny);

}

bool operator<(const BritishMoney& bm) const noexcept{

if(funt < bm.funt) return true;

else if(funt == bm.funt){

if(shil < bm.shil) return true;

else if(shil == bm.shil)

return penny < bm.penny;

else return false;

}

else return false;

}

void print(){

std::cerr << "фунты = " << funt << ", шиллинги = " << static\_cast<int>(shil) << ", пенсы = " << static\_cast<int>(penny);

}

BritishMoney exec(const char\* command, const BritishMoney& bm) const noexcept {

if(static\_cast<std::string>(command) == "a") return \*this+bm;

else if(static\_cast<std::string>(command) == "s") return \*this-bm;

else return BritishMoney(0,0,0);

}

BritishMoney exec(const char\* command, const double koef) const noexcept {

if(static\_cast<std::string>(command) == "dbn") return \*this/koef;

else if(static\_cast<std::string>(command) == "m") return \*this\*koef;

else return BritishMoney(0,0,0);

}

unsigned long long funt;

unsigned char shil;

unsigned char penny;

};

Read\_input.cpp

#include <iostream>

#include <string>

#include <cstdio>

#include <set>

typedef enum {

END\_OF\_FILE,

VALID\_INPUT,

INVALID\_INPUT,

END\_OF\_LINE,

} read\_return\_t;

read\_return\_t get\_command(std::set<std::string>& valid\_commands, char\* command)

{

char ch(' ');

while((ch == '\t') || (ch == ' ')) {

std::cin >> ch;

if(std::cin.eof()) return END\_OF\_FILE;

if(ch == '\n') return END\_OF\_LINE;

}

std::cin.unget();

std::cin >> command;

if(std::cin.fail()) {

std::cin.clear();

return INVALID\_INPUT;

}

if(valid\_commands.count(command)) return VALID\_INPUT;

else return INVALID\_INPUT;

}

template <typename T>

read\_return\_t get\_value(T& d)

{

char ch(' ');

while((ch == '\t') || (ch == ' ')) {

std::cin >> ch;

if(std::cin.eof()) return END\_OF\_FILE;

if(ch == '\n') return END\_OF\_LINE;

}

std::cin.unget();

std::cin >> d;

if(std::cin.fail()) {

std::cin.clear();

return INVALID\_INPUT;

}

return VALID\_INPUT;

}

template <>

read\_return\_t get\_value (unsigned int& d) {

char ch(' ');

while((ch == '\t') || (ch == ' ')) {

std::cin >> ch;

if(std::cin.eof()) return END\_OF\_FILE;

if(ch == '\n') return END\_OF\_LINE;

}

std::cin.unget();

if(ch == '-') return INVALID\_INPUT;

std::cin >> d;

if(std::cin.fail()) {

std::cin.clear();

return INVALID\_INPUT;

}

return VALID\_INPUT;

}

template <>

read\_return\_t get\_value (unsigned long long& d) {

char ch(' ');

while((ch == '\t') || (ch == ' ')) {

std::cin >> ch;

if(std::cin.eof()) return END\_OF\_FILE;

if(ch == '\n') return END\_OF\_LINE;

}

std::cin.unget();

if(ch == '-') return INVALID\_INPUT;

std::cin >> d;

if(std::cin.fail()) {

std::cin.clear();

return INVALID\_INPUT;

}

return VALID\_INPUT;

}

Вывод

Язык С++ даёт возможность переопределить операции для различных классов, что повышает его лаконичность и объектоориентированность.

ЛИТЕРАТУРА

1. Б. Страуструп "Язык программирования С++", четвёртое издание, 2013.