**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСТИЕТ**

**Лабораторная работа №2 по дисциплине**

**«Программирование»**

Переопределение операций

Вариант – 14

Группа: **АВТ-907**

Студент: **Смоленский И.В.**

Преподаватель: **Булатов А. Д.**

НОВОСИБИРСК 2020

**Задание****:**

**Цель работы:** ознакомиться с особенностями использования дружественных классов и функций, а также возможностью получения законченного нового типа данных, определив для него допустимые операции с помощью перегрузки операторов.

**Задания на лабораторную работу**

Для класса из лаб. работы №1 реализовать набор операций для работы с объектами класса: по заданию. Изменить демонстрационную программу, продемонстрировав все перегруженные операции. Операции перегрузить методами класса и дружественными функциями.

Для класса Bankomat перегрузить следующие операции:

* операция вычитания числа для снятия суммы денег;
* операция сложения с числом для загрузки текущей суммы денег в банкомате;
* операции сравнения с числом для проверки текущей суммы денег в банкомате;
* операция вызова функции - получение строки-чека операции;
* операцию присваивания.

**Теоретические сведения:**

Абстрактный тип данных (АТД) тип данных, определяемый только через операции, которые могут выполняться над соответствующими объектами безотносительно к способу представления этих объектов. Для реализации АТД необходимо, во-первых, выбрать представление памяти для объектов и, во-вторых, реализовать операции в терминах выбранного представления.

Примером абстрактного типа данных является класс в языке С++. В С есть возможность использовать знаки стандартных операций для записи выражений как для встроенных, так и для АТД.

Перегрузка операций

В языке С++ для перегрузки операций используется ключевое слово operator, с помощью которого определяется специальная операция-функция (operator function).

Формат операции-функции:

тип\_возвр\_значения operator знак\_операции (специф\_параметров)

{операторы\_тела\_функции}

Перегрузка унарных операций

- Любая унарная операция  может быть определена двумя способами: либо как компонентная функция без параметров, либо как глобальная (возможно дружественная) функция с одним параметром. В первом случае выражение  Z означает вызов Z.operator  (), во втором  вызов operator (Z).

- Унарные операции, перегружаемые в рамках определенного класса, могут перегружаться только через нестатическую компонентную функцию без параметров. Вызываемый объект класса автоматически воспринимается как операнд.

- Унарные операции, перегружаемые вне области класса (как глобальные функции), должны иметь один параметр типа класса. Передаваемый через этот параметр объект воспринимается как операнд.

class ZZ {

public:

friend ZZ operator-(ZZa);

ZZ operator-();

};

Перегрузка бинарных операций

- Любая бинарная операция  может быть определена двумя способами: либо как компонентная функция с одним параметром, либо как глобальная (возможно дружественная) функция с двумя параметрами. В первом случае x y означает вызов x.operator(y), во втором – вызов operator (x,y).

- Операции, перегружаемые внутри класса, могут перегружаться только нестатическими компонентными функциями с параметрами. Вызываемый объект класса автоматически воспринимается в качестве первого операнда.

- Операции, перегружаемые вне области класса, должны иметь два операнда, один из которых должен иметь тип класса.

Перегрузка операторов дает возможность добавлять к встроенным типам данных в С++ новые типы. В зависимости от назначения перегрузка операторов объявляется так же, как и обычная дружественная функция или функция-член класса.

class ZZ {

public:

friend ZZ operator+(ZZa, ZZb);

ZZ operator-( ZZb);

};

Имена функций - operator-, operator+ состоят из ключевого слова operator и символа операции.

Список операций, которые могут быть перегружены

\* / + - % ^ & :

- ! , = < > <= >=

++ -- << >> == != && ||

\*= /\* %= ^= &= |= += -=

<<= >>= -> ->\* [] () new delete

Не перегружаются операции:

- :: - расширение области видимости,

- \* - раз именование указателя,

- . - точка,

- sizeof() – определение размера памяти

- # .

Перегрузка не вносит в С++ ничего нового, только лишь позволяет использовать в выражениях объекты классов вместо того, чтобы передавать их в функции.

**Date\_time.cpp**

#include "Bankomat.h"

#include <iostream>

#include <string>

#include <exception>

using namespace std;

Bankomat operator-(Bankomat Sub, unsigned subtract) // try catch

{

try // Try catch block works properly

{

if ((subtract > Sub.sumMaxGet) || (subtract > Sub.sumBank))

{

throw invalid\_argument("Invalid arg");

}

if (subtract < 0)

{

throw invalid\_argument("Invalid arg");

}

}

catch (const std::invalid\_argument & err)

{

cerr << " Invalid argument: " << err.what() << endl;

}

Sub.sumBank -= subtract;

return Sub;

}

Bankomat operator+(Bankomat Sub, unsigned addition) // try catch

{

try // Try catch block works properly

{

if (addition < 0)

{

throw std::invalid\_argument("Invalid arg");

}

}

catch (const std::invalid\_argument & err)

{

cerr << " Invalid argument: " << err.what() << endl;

}

Sub.sumBank += addition;

return Sub;

}

bool Bankomat::operator>(Bankomat sub)

{

return (strcmp(sub.getId(), id)) > 0;

}

bool Bankomat::operator<(Bankomat sub)

{

return (strcmp(sub.getId(), id)) < 0;

}

bool Bankomat::operator==(Bankomat sub)

{

return (strcmp(sub.getId(), id)) == 0;

}

bool Bankomat::operator<(char\* sub)

{

return (strcmp(sub, id)) == 0;

}

bool Bankomat::operator>(char\* sub)

{

return (strcmp(sub, id)) == 0;

}

bool Bankomat::operator==(char\* sub)

{

return (strcmp(sub, id)) == 0;

}

char\* Bankomat::operator()()

{

return (toString());

}

Bankomat Bankomat::operator=(const Bankomat &assignment)

{

id = assignment.id;

sumBank = assignment.sumBank;

sumMaxGet = assignment.sumMaxGet;

return (\*this);

}

std::ostream& operator<<(std::ostream& os,const Bankomat &p)

{

os << "ID of ATM: " << p.id << "\nsum in ATM: " << p.sumBank << "\nMax sum of Get: " << p.sumMaxGet << endl;

return os;

}

std::istream& operator>>(std::istream& is, Bankomat& p)

{

is >> p.id >> p.sumBank >> p.sumMaxGet;

return is;

}

**Вывод:**

В ходе проделанной работы мной были перегружены операторы присваивания, сложения и вычитания, а также тип данных char\*.