Министерство образования и науки Российской Федерации

МГТУ им .Н.Э Баумана

Факультет «Робототехника и комплексная автоматизация»

Кафедра «РК6»

**Отчет по лабораторной работе**

По курсу «Объектно-ориентированное программирование»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| Выполнил: |  |  | Студент | Журавлев Н.В. |
|  |  |  | Группа | РК6-12Б |
|  |  |  |  |  |
| Проверил: |  |  |  | Волосатова Т.М. |
|  |  |  | Дата | 22.12.2020 |
|  |  |  | Подпись | \_\_\_\_\_\_\_\_\_\_\_ |

**Практика №2**

Разработать ООП банкомата, которая обеспечивает выплату любой требуемой суммы денег минимальным количеством денежных знаков из фиксированного набора купюр различного достоинства с учётом имеющейся наличности. Информации о денежных ресурсах банкомата по числу купюр каждого достоинства должна быть сосредоточена в текстовом файле. Он должен соответствующим образом изменяться после выплаты денежных сумм по запросам из потока стандартного ввода. Число и достоинство денежных купюр каждой выплаты должно отображаться в потоке стандартного вывода. При разработке программы необходимо учесть возможность интерактивной модификации ресурсного файла банкомата независимо от запросов по выплате денежных сумм.

**Описание алгоритма**

Алгоритм снятия денег:

1. Пока сумма > 0. Если сумма больше 1 значения вектора и количество больше 0, то из суммы вычитается из 1 элемента вектора, а из 2 элемента вычитается единица.

Иначе переходим к следующую вектору.

3.1) Затем проверяется на то, можно ли следующими элементами заполнить сумму, а именно в цикле пока вычитается по 1 из сохранённого значения вектора money

1. Затем в случаях успешного выполнения обновляется ресурсный файл(метод update\_file).
2. Этот возвращает false, если вывод суммы невозможен. **Текст программы**

BANK.h

#ifndef UNTITLED1\_BANK\_H

#define UNTITLED1\_BANK\_H

#include <vector>

#include <fstream>

class BANK {

private:

std::string file\_path;

public:

explicit BANK(std::string path);

~BANK() = default;

bool atm\_request(int sum);

bool output();

private:

static bool foo\_sort(std::pair<int, int> one, std::pair<int, int> two);

bool fill\_from\_file(std::vector<std::pair<int, int>> &money);

bool update\_file(const std::vector<std::pair<int, int>> &money);

};

#endif //UNTITLED1\_BANK\_H

BANK.cpp

#include <algorithm>

#include <utility>

#include <vector>

#include <iostream>

#include "BANK.h"

BANK::BANK(std::string path): file\_path(std::move(path)) { }

bool BANK::atm\_request(int sum) {

std::vector<std::pair<int, int>> money, save\_money;

if (!fill\_from\_file(money)) {

return false;

}

save\_money = money;

size\_t it = 0;

size\_t size = money.size();

int save\_count, save\_sum = sum;

int sum\_2 = sum;

if (size == 0) {

return false;

}

while(sum > 0) {

if (sum >= money[it].first && money[it].second > 0) {

sum -= money[it].first;

--money[it].second;

} else {

++it;

if (it >= size) {

return false;

}

save\_count = money[it].second;

while (sum\_2 > 0){

if (sum\_2 >= money[it].first && save\_count > 0) {

sum\_2 -= money[it].first;

--save\_count;

} else {

break;

}

}

if (sum\_2 == 0) {

save\_money[it].second = save\_count;

return update\_file(save\_money);

} else{

sum\_2 = save\_sum;

}

}

}

return update\_file(money);

}

bool BANK::foo\_sort(std::pair<int, int> one, std::pair<int, int> two) {

return one.first > two.first;

}

bool BANK::fill\_from\_file(std::vector<std::pair<int, int>> &money) {

std::fstream file;

file.open(file\_path, std::ios::in);

if (!file.is\_open()) {

return false;

}

while (true) {

std::pair<int, int> data;

if (!(file >> data.first >> data.second)) {

break;

}

money.push\_back(data);

}

std::sort(money.begin(), money.end(), foo\_sort);

file.close();

return true;

}

bool BANK::update\_file(const std::vector<std::pair<int, int>> &money) {

std::fstream file;

file.open(file\_path, std::ios::out | std::ios::trunc);

if (!file.is\_open()) {

return false;

}

for (auto &it: money) {

if (it.second > 0) {

if (!(file << it.first << " " << it.second << std::endl)) {

file.close();

return false;

}

}

}

file.close();

return true;

}

bool BANK::output() {

std::fstream file;

file.open(file\_path, std::ios::in);

if (!file.is\_open()) {

return false;

}

std::string line;int i = 0;

while (!file.eof()) {i++;

getline(file, line);

std::cout << line << std::endl;

}

file.close();

if (i == 1) {

return false;

}

return true;

}

main.cpp

#include <iostream>

#include "BANK.h"

#define BANK\_FILE\_NAME "atm.txt"

using namespace std;

int main() {

BANK BANK(BANK\_FILE\_NAME);

int sum = 0;

std::cout << "ATM content: " << std::endl;

while (BANK.output()) {

std::cout << "Enter money request:";

if (!(std::cin >> sum)) {

std::cout << "Invalid input." << std::endl << std::endl;

cin.clear();

fflush(stdin);

continue;

}

if (!BANK.atm\_request(sum)) {

std::cout << "Error.Request failed." << std::endl << std::endl;

continue;

}

std::cout << "The money has been issued.\nNew ATM content:" << std::endl;

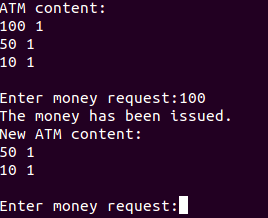
}

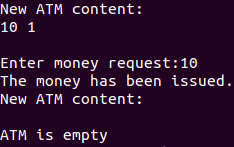
std::cout << "ATM is empty" << std::endl;

return 0;

}

**Тесты**





**Результат**

Если файл не пустой выводить содержимое, иначе выводить диагностическое сообщение и завершать программу.

**Списки литературы**

1)Волосатова Т. М., Родионов С. В. «Объектно-ориентированное программирование на С++» 2)http://bigor.bmstu.ru/?cnt/?doc=VU/base.cou