**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

## Факультет компьютерных наук

## Образовательная программа бакалавриата «Программная инженерия»

**ПРОГРАММА “ПЕРВАЯ ВОЕННАЯ ЗАДАЧА”**

**Пояснительная записка**

## Исполнитель студент группы БПИ196

## / Татаринов Н.А. /

## «19» ноября 2020 г.

**Москва 2020**

**Пояснительная записка Листов 15**

**Москва 2020**

**АННОТАЦИЯ**

## В данном программном документе приведена пояснительная записка к программе

## «Программа “Первая военная задача”».

## В разделе «Введение» указано наименование программы и краткая характеристика области её применения.

## В разделе «Технические характеристики» содержатся следующие подразделы:

## постановка задачи на разработку программы;

## описание алгоритма и функционирования программы с обоснованием выбора схемы алгоритма решения задачи и возможные взаимодействия программы с другими программами;

## описание и обоснование выбора метода организации входных и выходных данных.

## В разделе «Код программы» исходный текст программы на языке C++ с подробными комментариями необходимыми для понимания программы сторонним человеком.

**Содержание**

1. [ВВЕДЕНИЕ 4](#_bookmark0)
   1. [Наименование программы 4](#_bookmark1)
   2. [Краткая характеристика области применения 4](#_bookmark2)
2. [ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 5](#_bookmark3)
   1. [Постановка задачи на разработку программы 5](#_bookmark4)
   2. [Описание алгоритма и функционирования программы 5](#_bookmark5)
   3. [Описание и обоснование выбора метода организации входных и выходных данных 5](#_bookmark6)
3. [КОД ПРОГРАММЫ 6](#_bookmark7)
4. [ТЕСТИРОВАНИЕ 12](#_bookmark8)

[ПРИЛОЖЕНИЕ 1 ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ПЕРЕМЕННЫХ 14](#_bookmark9)

[ПРИЛОЖЕНИЕ 2 ОПИСАНИЕ И ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ПРОЦЕДУР 15](#_bookmark10)

# ВВЕДЕНИЕ

# Наименование программы

## Наименование программы – «Программа “Первая военная задача”».

# Краткая характеристика области применения

## Текст задания: «Темной-темной ночью прапорщики Иванов,Петров и Нечепорчук занимаются хищением военного имущества со складародной военной части. Будучи умными людьми и отличниками боевой истроевой подготовки, прапорщики ввели разделение труда: Иванов выноситимущество со склада, Петров грузит его в грузовик, а Нечепорчукподсчитывает рыночную стоимость добычи. Требуется составитьмногопоточное приложение, моделирующее деятельность прапорщиков. Прирешении использовать парадигму «производитель-потребитель»».

## Программа выполняется в рамках курса «Архитектура вычислительных систем» в соответствии с учебным планом подготовки бакалавров по направлению 09.03.04 «Программная инженерия» Национального исследовательского университета «Высшая школа экономики», факультет компьютерных наук, департамент программной инженерии.

# ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

# 2.1 Постановка задачи на разработку программы

## Программа должна создать файл с данными и многопотоковую работу с ними.

# 2.2 Описание алгоритма и функционирования программы

## В программе создаётся файл (склад), состоящий из наименований товаров (std::string) и их стоимостей (unsigned int). Далее, создаются 3 потока: поток-производитель, который считывает товары из файла-склада, сохраняет в 2 буфера данных их наименования и стоимости; поток-потребитель, получающий наименования товары из одного из буферов и сохраняющий их в другой файл (файл-грузовик); поток-потребитель, получающий стоимости товаров из другого буфера и сохраняющий их в третий файл (файл-прайслист).

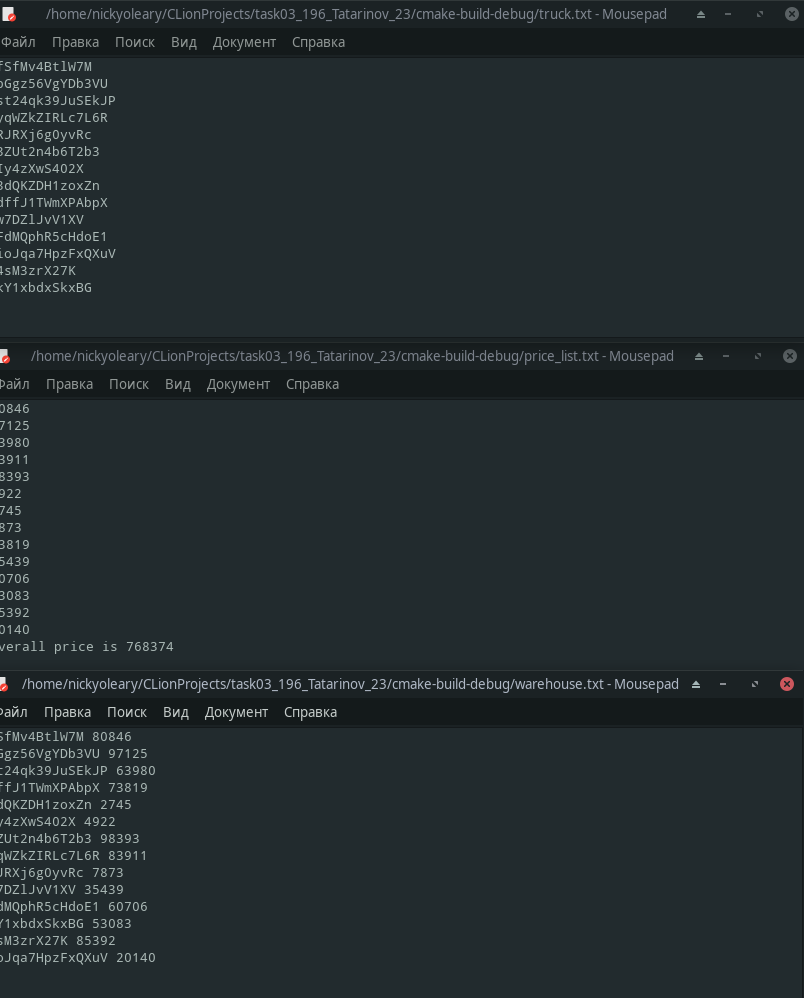
# 2.3 Описание и обоснование выбора метода организации входных и выходных данных

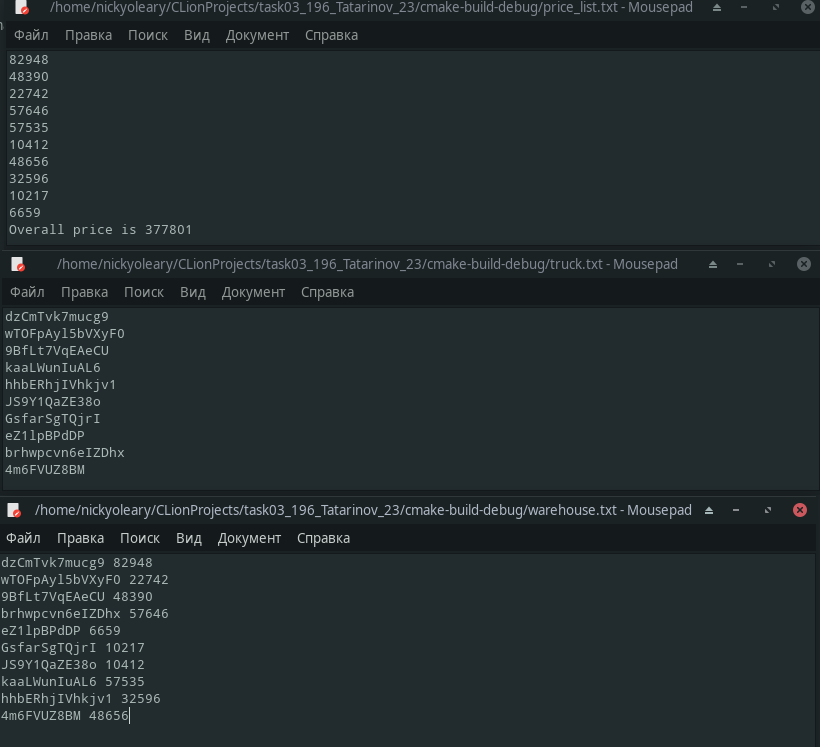
## Входные данные формируются в самой программе в виде файла “warehouse.txt”. Далее, из него читаются данные и выводятся в разделённом виде в 2 других файла. При этом, в консоль выводятся комментарии.

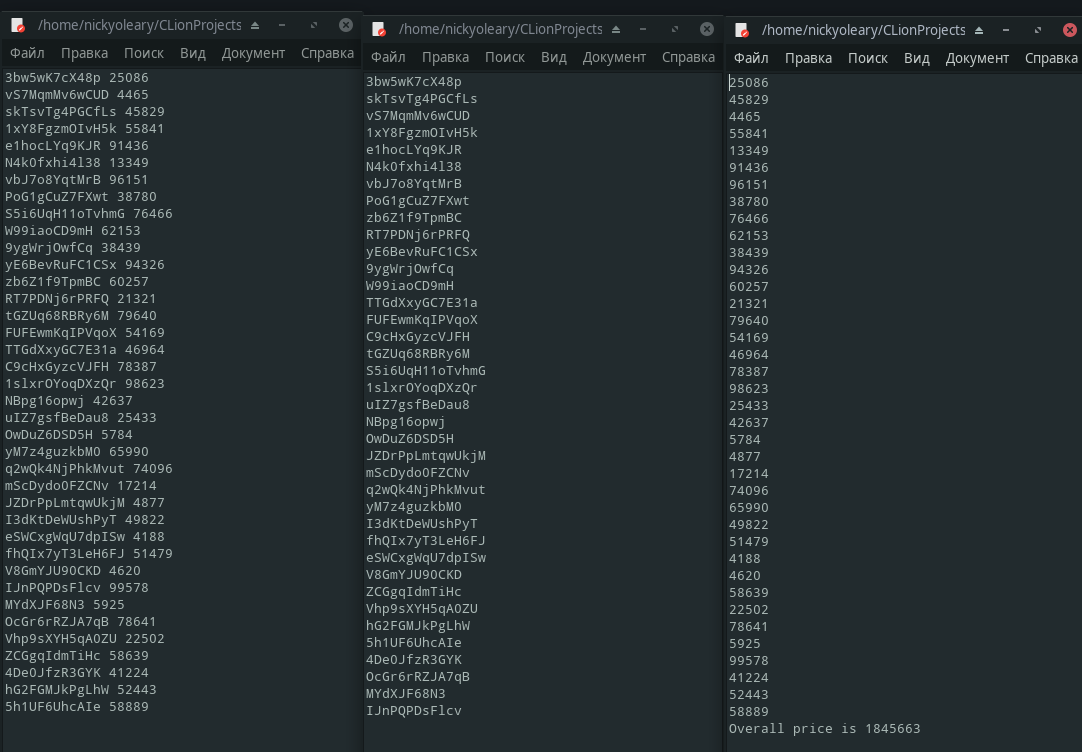
# КОД ПРОГРАММЫ

#include <iostream>  
#include <fstream>  
#include <thread>  
#include <stack>  
#include <mutex>  
#include <condition\_variable>  
#include <random>  
  
std::mutex buf\_mu, pr\_con\_mu;  
std::condition\_variable cond;  
bool file\_is\_empty = false;  
  
///Представляет собой класс для обмена данными между  
///производителями и потребителями.  
template<class T>  
class Buffer {  
private:  
 ///Максимальное количество элементов в буфере.  
 static const size\_t st\_max\_size = 5;  
 ///Контейнер для хранения элементов буфера.  
 std::stack<T> \*st;  
  
public:  
 Buffer();  
 ~Buffer();  
  
 ///Добавляет новый элемент в буфер.  
 void add(const T& newElement);  
  
 ///Удалаяет верхний элемент из буфера.  
 T remove();  
  
 ///Возвращает true, если буфер пуст.  
 [[nodiscard]] bool empty() const;  
};  
  
///Представляет собой класс производителя.  
class Producer{  
private:  
 ///Буфер наименований товаров.  
 Buffer<std::string> \*units;  
 ///Буфер цен товаров.  
 Buffer<unsigned int> \*prices;  
 ///Файл, откуда считываются товары.  
 std::ifstream fin;  
 ///Предоставляем классу Consumer доступ к  
 ///приватным полям и методам.  
 template <class>  
 friend class Consumer;  
  
public:  
 Producer(Buffer<std::string> \*\_units,  
 Buffer<unsigned int> \*\_prices,  
 const std::string& path);  
 ~Producer();  
  
 ///Метод работы с данными.  
 void run();  
};  
  
///Представляет собой класс потребителя.  
template<class T>  
class Consumer{  
private:  
 ///Буфер товаров (наименований или стоимостей).  
 Buffer<T> \*objects;  
 ///Файл, в который выводятся товары.  
 std::ofstream fout;  
 ///Суммарная стоимость товаров (если потребитель  
 ///хранит стоимости).  
 unsigned long long overall\_price;  
  
public:  
 explicit Consumer(const std::string& path);  
 ~Consumer();  
  
 ///Метод работы с товарами.  
 void run();  
  
 ///Получает от потребителя буфер для работы.  
 void get\_buffer(const Producer \*pr);  
};  
  
///Метод генерации случайной строки из латинских букв и цифр  
///длиной от 10 до 15 символов.  
static std::string generate\_random\_string();  
  
///Метод создания файла с товарами.  
static void create\_input\_file();  
  
int main() {  
 create\_input\_file();  
 auto \*units = new Buffer<std::string>();  
 auto \*prices = new Buffer<unsigned int>();  
 auto Ivanov = new Producer(units, prices, "warehouse.txt");  
 auto Petrov = new Consumer<std::string>("truck.txt");  
 auto Necheporchuk = new Consumer<unsigned int>("price\_list.txt");  
 Petrov->get\_buffer(Ivanov);  
 Necheporchuk->get\_buffer(Ivanov);  
  
 std::thread Ivanov\_thread(&Producer::run, Ivanov);  
 std::thread Petrov\_thread(&Consumer<std::string>::run, Petrov);  
 std::thread Necheporchuk\_thread(&Consumer<unsigned int>::run, Necheporchuk);  
  
 Ivanov\_thread.join();  
 file\_is\_empty = true;  
 Petrov\_thread.join();  
 Necheporchuk\_thread.join();  
 delete Necheporchuk;  
 delete Petrov;  
 delete Ivanov;  
 delete prices;  
 delete units;  
 return 0;  
}  
  
static std::string generate\_random\_string(){  
 thread\_local std::random\_device rd;  
 thread\_local std::mt19937 engine(rd());  
 thread\_local std::uniform\_int\_distribution<unsigned char> dist1(10, 15);  
 thread\_local std::uniform\_int\_distribution<unsigned char> dist2(0, 61);  
 std::string chars\_to\_use = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";  
 std::string result;  
 unsigned char str\_length = dist1(engine);  
 for(size\_t i = 0; i < str\_length; i++){  
 result += chars\_to\_use[dist2(engine)];  
 }  
 return result;  
}  
  
static void create\_input\_file(){  
 std::ofstream fout;  
 fout.open("warehouse.txt");  
 thread\_local std::random\_device rd;  
 thread\_local std::mt19937 engine(rd());  
 thread\_local std::uniform\_int\_distribution<unsigned char> dist1(10, 50);  
 thread\_local std::uniform\_int\_distribution<unsigned int> dist2(1000, 100000);  
 unsigned char amount\_of\_units = dist1(engine);  
 for(unsigned char i = 0; i < amount\_of\_units; i++){  
 fout << generate\_random\_string() << " " << dist2(engine) << "\n";  
 }  
 fout.close();  
}  
  
template<class T>  
Buffer<T>::Buffer() {  
 st = new std::stack<T>();  
}  
  
template<class T>  
Buffer<T>::~Buffer(){  
 delete st;  
}  
  
template<class T>  
void Buffer<T>::add(const T& newElement) {  
 while(true) {  
 std::unique\_lock<std::mutex> locker(buf\_mu);  
 cond.wait(locker, [this]() { return st->size() < st\_max\_size; });  
 st->push(newElement);  
 locker.unlock();  
 cond.notify\_all();  
 return;  
 }  
}  
  
template<class T>  
T Buffer<T>::remove() {  
 while(true) {  
 std::unique\_lock<std::mutex> locker(buf\_mu);  
 cond.wait(locker, [this]() { return !st->empty(); });  
 T top = st->top();  
 st->pop();  
 locker.unlock();  
 cond.notify\_all();  
 return top;  
 }  
}  
  
template<class T>  
bool Buffer<T>::empty() const{  
 return st->empty();  
}  
  
Producer::Producer(Buffer<std::string> \*\_units,  
 Buffer<unsigned int> \*\_prices,  
 const std::string& path) : units(\_units), prices(\_prices){  
 fin.open(path);  
}  
  
Producer::~Producer() {  
 units = nullptr;  
 prices = nullptr;  
 fin.close();  
}  
  
void Producer::run() {  
 std::string unit;  
 unsigned int price;  
 while(fin >> unit >> price){  
 units->add(unit);  
 prices->add(price);  
 std::unique\_lock<std::mutex> locker(pr\_con\_mu);  
 std::cout << "Unit \"" << unit << "\" of price " << price <<  
 " is taken out of the warehouse\n";  
 std::this\_thread::sleep\_for(std::chrono::milliseconds(50));  
 locker.unlock();  
 }  
}  
  
template<class T>  
Consumer<T>::Consumer(const std::string& path) : overall\_price(0){  
 fout.open(path);  
}  
  
template<class T>  
Consumer<T>::~Consumer() {  
 if(typeid(T).name() == typeid(unsigned int).name()) {  
 std::cout << "Overall price is " << overall\_price << "\n";  
 fout << "Overall price is " << overall\_price << "\n";  
 }  
 objects = nullptr;  
 fout.close();  
}  
  
template<class T>  
void Consumer<T>::run() {  
 while(!file\_is\_empty || !objects->empty()){  
 if(!objects->empty()) {  
 T unit = objects->remove();  
 std::unique\_lock<std::mutex> locker(pr\_con\_mu);  
 if (typeid(T) == typeid(std::string)) {  
 fout << unit << "\n";  
 std::cout << "Unit \"" << unit << "\" is entrained in the truck\n";  
 std::this\_thread::sleep\_for(std::chrono::milliseconds(50));  
 } else if (typeid(T) == typeid(unsigned int)) {  
 overall\_price += \*reinterpret\_cast<unsigned int\*>(&unit);  
 fout << unit << "\n";  
 std::cout << "+" << unit << " in price list\n";  
 std::this\_thread::sleep\_for(std::chrono::milliseconds(50));  
 }  
 locker.unlock();  
 }  
 }  
}  
  
template<class T>  
void Consumer<T>::get\_buffer(const Producer \*pr){  
 if(typeid(T).name() == typeid(std::string).name()){  
 objects = reinterpret\_cast<Buffer<T> \*>(pr->units);  
 } else if (typeid(T).name() == typeid(unsigned int).name()){  
 objects = reinterpret\_cast<Buffer<T> \*>(pr->prices);  
 }  
}

# ТЕСТИРОВАНИЕ







**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

## CODE REVIEW [Электронные ресурс]// URL: https://codereview.stackexchange.com/questions/235651/a-multi-thread-producer-consumer-where-a-consumer-has-multiple-producers-c17 (режим доступа: свободный, дата обращения: ноябрь 2020)