|  |  |  |
| --- | --- | --- |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное автономное  образовательное учреждение высшего образования  «Пермский государственный национальный  исследовательский университет» | | |
|  | Институт компьютерных наук и технологий | |
| **ОТЧЁТ**  по индивидуальной работе №2  по дисциплине «Язык программирования C++»  Вариант 11 | | |
|  | | Работу выполнил  студент группы ИТ-11,12-2024 1 курса  Демин М.В  «24» Июня 2025 г. |
| Работу проверил  Кнутова Н.С  «26» Июня 2025 г. |
| Пермь 2025 | | |

**Содержание:**   
1.Постановка задачи ....................................................................................... 3

2.Алгоритм решения ....................................................................................... 4

3.Тестирования ................................................................................................ 5

4.Код программы .............................................................................................9

# Постановка задачи

Баржа. На барже располагается K грузовых отсеков. В каждый отсек можно поместить некоторое количество бочек с одним из 10 000 видов топлива. Причём извлечь бочку из отсека можно лишь в случае, если все бочки, помещённые в этот отсек после неё, уже были извлечены. Таким образом, в каждый момент времени в каждом непустом отсеке имеется ровно одна бочка, которую можно извлечь, не трогая остальных. Будем называть такие бочки крайними.

Изначально баржа пуста. Затем она последовательно проплывает через N доков, причём в каждом доке на баржу либо погружается бочка с некоторым видом топлива в некоторый отсек, либо выгружается крайняя бочка из некоторого отсека. Однако, если указанный отсек пуст, либо если выгруженная бочка содержит не тот вид топлива, который ожидалось, следует зафиксировать ошибку. Если на баржу оказывается погружено более P бочек или если после прохождения всех доков она не стала пуста, следует также зафиксировать ошибку. От вас требуется либо указать максимальное количество бочек, которые одновременно пребывали на барже либо зафиксировать ошибку.

Входные данные: в первой строке три целых числа N, K и P (1 ≤ N, K, P ≤ 100 000). Далее следует N строк с описанием действия, выполняемого в очередном доке. Если в нём происходит погрузка, то строка имеет вид «+ A B», где A — номер отсека, в который помещается бочка, а B — номер вида топлива в ней. Если же док занимается разгрузкой, то строка имеет вид «- A B», где A — номер отсека, из которого извлекается бочка, а B — номер ожидаемого вида топлива. Выходные данные: вывести либо одно число, равное искомому максимуму в случае безошибочного прохождения баржой маршрута, либо вывести слово «Error» в противном случае.

# Алгоритм решения

Шаг 1: Инициализация системы

-Чтение входных параметров:

Считываем три числа:

N - количество операций

K - количество отсеков

P - максимальное допустимое количество бочек

-Проверяем корректность ввода (числа в диапазоне 1-100 000)

-Создание модели баржи:

-Инициализируем массив стеков для K отсеков

-Устанавливаем начальные значения:

currentBarrels = 0 (текущее количество бочек)

maxBarrelsEver = 0 (максимальное количество бочек)

errorOccurred = false (флаг ошибки)

Шаг 2: Обработка операций

Для каждой из N операций последовательно выполняем:

Чтение операции:

-Считываем символ операции (+ или -)

-Считываем номер отсека A

-Считываем тип топлива B

Проверка типа операции для “+” и для “-”

Шаг 3: Проверка состояния после операций

После обработки всех операций выполняем финальные проверки:

-Проверяем флаг ошибки

-Проверяем пустоту баржи (currentBarrels == 0)

-Выводим результат:

Если ошибок нет и баржа пуста → maxBarrelsEver

В противном случае → "Error"

Тестирование

1. Без ошибок

Входные данные:  
6 1 2

+ 1 1

+ 1 2

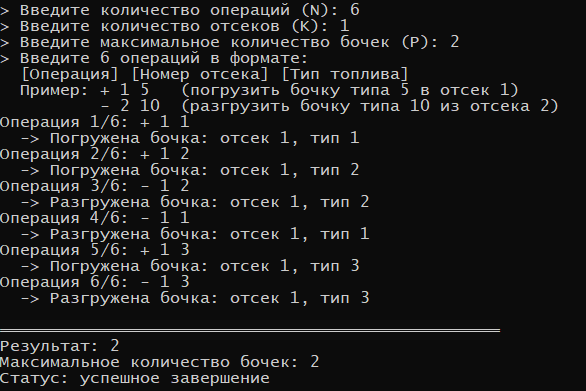
- 1 2

- 1 1

+ 1 3

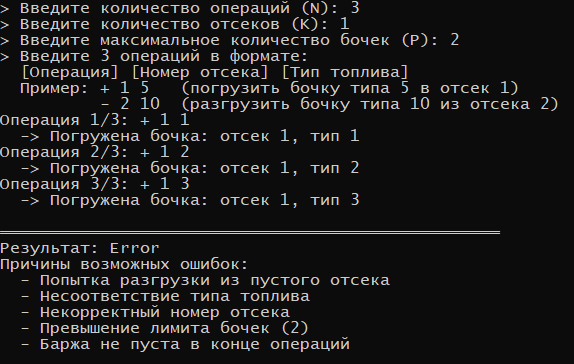
- 1 3

Ожидаемый результат : 2

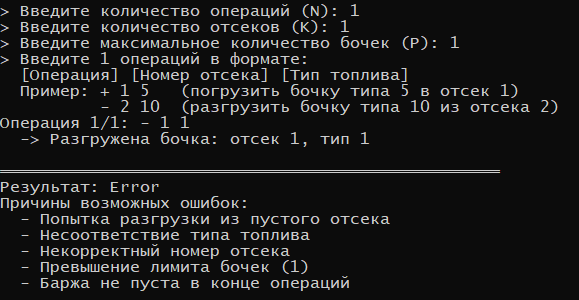


Всё верно.

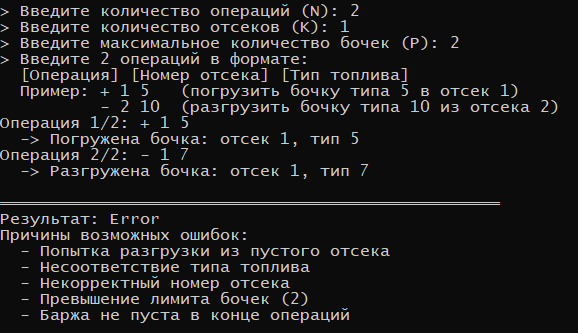
2.Ошибка (Количество бочек превышает лимит)



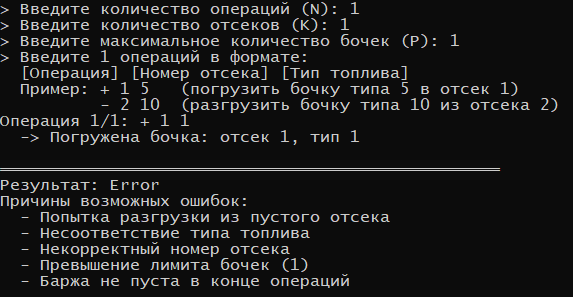
3.Ошибка (Убирают из пустого отсека)



4.Ошибка (Другой тип бочки)



5.Ошибка (Осталась 1 бочка)



6. Без ошибок с другими данными

Входные данные:

4 2 3

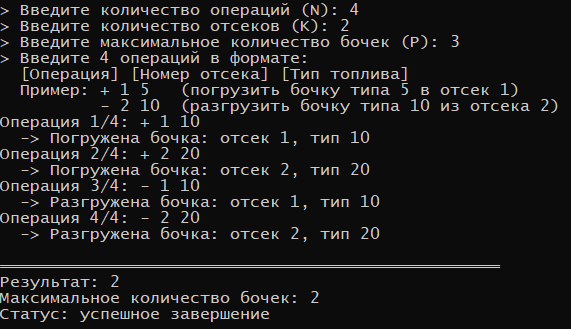
+ 1 10

+ 2 20

- 1 10

- 2 20

Ожидаемый результат: 2



Всё верно.

# Код программы

#include <iostream>  
#include <stdexcept>  
#include <limits>  
#include <cctype>  
using namespace std;

// Узел для реализации стека  
class StackNode {  
public:  
 int fuelType; // Тип топлива в бочке  
 StackNode\* next;// Указатель на следующий элемент в стеке

// Конструктор узла  
 StackNode(int type) : fuelType(type), next(nullptr) {}  
};

// Класс стека на основе связного списка  
class Stack {  
private:  
 StackNode\* topNode;

public:  
 // Конструктор: инициализация пустого стека  
 Stack() : topNode(nullptr) {}

// Деструктор: освобождение памяти всех элементов стека  
 ~Stack() {  
 while (!isEmpty()) {  
 pop();  
 }  
 }

// Проверка, пуст ли стек  
 bool isEmpty() const {  
 return topNode == nullptr;  
 }

// Добавление элемента на вершину стека  
 void push(int type) {  
 StackNode\* newNode = new StackNode(type);  
 newNode->next = topNode;  
 topNode = newNode;  
 }

//Удаление элемента с вершины стека  
 void pop() {  
 if (isEmpty()) throw runtime\_error("Попытка извлечения из пустого стека");  
 StackNode\* temp = topNode;  
 topNode = topNode->next;  
 delete temp;  
 }

// Получение значения верхнего элемента стека  
 int top() {  
 if (isEmpty()) throw runtime\_error("Попытка доступа к вершине пустого стека");  
 return topNode->fuelType;  
 }  
};

// Класс для моделирования баржи  
class Barge {  
private:  
 int numCompartments; // Количество отсеков (K)  
 int maxBarrels; // Максимальное допустимое количество бочек (P)  
 long long currentBarrels; // Текущее количество бочек на барже  
 long long maxBarrelsEver; // Максимальное количество бочек за все время  
 Stack\*\* compartments; // Массив стеков для отсеков  
 bool errorOccurred; // Флаг возникновения ошибки

// Объявление дружественной функции для валидации операций c использованием friend  
 friend bool validateOperation(const Barge& barge, int compartmentNum);

public:  
 // Конструктор баржи  
 Barge(int compartmentsCount, int maxCapacity) :  
 numCompartments(compartmentsCount),  
 maxBarrels(maxCapacity),  
 currentBarrels(0),  
 maxBarrelsEver(0),  
 compartments(nullptr),  
 errorOccurred(false)  
 {  
 try {  
 compartments = new Stack \* [numCompartments + 1];

for (int i = 0; i <= numCompartments; i++) {  
 compartments[i] = nullptr;  
 }

for (int i = 1; i <= numCompartments; i++) {  
 compartments[i] = new Stack();  
 }  
 }  
 catch (const bad\_alloc& e) {  
 errorOccurred = true;  
 cleanup();  
 throw runtime\_error("Ошибка выделения памяти");  
 }  
 }

// Деструктор: освобождение ресурсов  
 ~Barge() {  
 cleanup();  
 }

// Очистка ресурсов баржи  
 void cleanup() {  
 if (compartments) {  
 for (int i = 1; i <= numCompartments; i++) {  
 if (compartments[i]) {  
 delete compartments[i];  
 }  
 }  
 delete[] compartments;  
 compartments = nullptr;  
 }  
 }

// Обработка операции погрузки бочки  
 void load(int compartmentNum, int fuelType) {  
 if (errorOccurred) return;

if (!validateOperation(\*this, compartmentNum)) {  
 errorOccurred = true;  
 return;  
 }

currentBarrels++;

if (currentBarrels > maxBarrels) {  
 errorOccurred = true;  
 return;  
 }

try {  
 compartments[compartmentNum]->push(fuelType);

if (currentBarrels > maxBarrelsEver) {  
 maxBarrelsEver = currentBarrels;  
 }  
 }  
 catch (...) {  
 errorOccurred = true;  
 }  
 }

// Обработка операции разгрузки бочки  
 void unload(int compartmentNum, int expectedFuelType) {  
 if (errorOccurred) return;

if (!validateOperation(\*this, compartmentNum)) {  
 errorOccurred = true;  
 return;  
 }

try {  
 if (compartments[compartmentNum]->isEmpty()) {  
 errorOccurred = true;  
 return;  
 }

int actualFuelType = compartments[compartmentNum]->top();  
 if (actualFuelType != expectedFuelType) {  
 errorOccurred = true;  
 return;  
 }

compartments[compartmentNum]->pop();  
 currentBarrels--;  
 }  
 catch (...) {  
 errorOccurred = true;  
 }  
 }

// Проверка наличия ошибок в работе баржи  
 bool hasError() const {  
 return errorOccurred || (currentBarrels != 0);  
 }

// Получение максимального количества бочек  
 long long getMaxBarrels() const {  
 return maxBarrelsEver;  
 }  
};

// Функция проверки корректности номера отсека  
bool validateOperation(const Barge& barge, int compartmentNum) {  
 return (compartmentNum >= 1 && compartmentNum <= barge.numCompartments);  
}

// Функция для безопасного ввода целого числа  
int safeInputInt(const string& prompt, int minVal, int maxVal) {  
 int value;  
 while (true) {  
 cout << prompt;  
 if (!(cin >> value)) {  
 // Очистка потока при некорректном вводе  
 cin.clear();  
 cin.ignore(numeric\_limits<streamsize>::max(), '\n');  
 cout << "Ошибка: введите целое число!" << endl;  
 continue;  
 }

// Проверка диапазона  
 if (value < minVal || value > maxVal) {  
 cout << "Ошибка: значение должно быть от "  
 << minVal << " до " << maxVal << "!" << endl;  
 continue;  
 }  
 break;  
 }  
 return value;  
}

// Функция для безопасного ввода операции (ИСПРАВЛЕНА)  
void safeInputOperation(char& op, int& compartmentNum, int& fuelType, int maxCompartments) {  
 string input;  
 while (true) {  
 // Ввод операции  
 cin >> input;  
 if (input.size() != 1 || (input[0] != '+' && input[0] != '-')) {  
 cin.clear();  
 cin.ignore(numeric\_limits<streamsize>::max(), '\n');  
 cout << "Ошибка: операция должна быть '+' или '-'!" << endl;  
 cout << "Повторите ввод операции: ";  
 continue;  
 }  
 op = input[0];  
 break;  
 }

// Ввод номера отсека  
 while (true) {  
 if (!(cin >> compartmentNum)) {  
 cin.clear();  
 cin.ignore(numeric\_limits<streamsize>::max(), '\n');  
 cout << "Ошибка: номер отсека должен быть числом!" << endl;  
 cout << "Повторите ввод номера отсека: ";  
 continue;  
 }  
 if (compartmentNum < 1 || compartmentNum > maxCompartments) {  
 cout << "Ошибка: номер отсека должен быть от 1 до "  
 << maxCompartments << "!" << endl;  
 cout << "Повторите ввод номера отсека: ";  
 continue;  
 }  
 break;  
 }

// Ввод типа топлива  
 while (true) {  
 if (!(cin >> fuelType)) {  
 cin.clear();  
 cin.ignore(numeric\_limits<streamsize>::max(), '\n');  
 cout << "Ошибка: тип топлива должен быть числом!" << endl;  
 cout << "Повторите ввод типа топлива: ";  
 continue;  
 }  
 break;  
 }  
}

// Функция для вывода инструкций  
void printInstructions() {  
 cout << "╔══════════════════════════════════════════════════════════╗" << endl;  
 cout << "║ МОДЕЛИРОВАНИЕ РАБОТЫ БАРЖИ ║" << endl;  
 cout << "╠══════════════════════════════════════════════════════════╣" << endl;  
 cout << "║ Программа моделирует работу баржи с отсеками, где каждая ║" << endl;  
 cout << "║ операция - это погрузка или разгрузка бочек с топливом. ║" << endl;  
 cout << "║ Отсеки работают по принципу стека (LIFO): ║" << endl;  
 cout << "║ - Можно разгружать только последнюю добавленную бочку ║" << endl;  
 cout << "║ ║" << endl;  
 cout << "║ Требования: ║" << endl;  
 cout << "║ 1. Количество отсеков: 1-100000 ║" << endl;  
 cout << "║ 2. Максимальное количество бочек: 1-100000 ║" << endl;  
 cout << "║ 3. Тип топлива - целое число ║" << endl;  
 cout << "╚══════════════════════════════════════════════════════════╝" << endl;  
}

int main() {  
 try {  
 printInstructions();

// Ввод основных параметров с проверкой  
 int numOperations = safeInputInt(  
 "> Введите количество операций (N): ",  
 1, 100000  
 );

int numCompartments = safeInputInt(  
 "> Введите количество отсеков (K): ",  
 1, 100000  
 );

int maxCapacity = safeInputInt(  
 "> Введите максимальное количество бочек (P): ",  
 1, 100000  
 );

// Создание объекта баржи  
 Barge barge(numCompartments, maxCapacity);

cout << "> Введите " << numOperations << " операций в формате:" << endl;  
 cout << " [Операция] [Номер отсека] [Тип топлива]" << endl;  
 cout << " Пример: + 1 5 (погрузить бочку типа 5 в отсек 1)" << endl;  
 cout << " - 2 10 (разгрузить бочку типа 10 из отсека 2)" << endl;

// Обработка всех операций  
 for (int i = 0; i < numOperations; i++) {  
 char op;  
 int compartmentNum, fuelType;

cout << "Операция " << (i + 1) << "/" << numOperations << ": ";  
 safeInputOperation(op, compartmentNum, fuelType, numCompartments);

if (op == '+') {  
 barge.load(compartmentNum, fuelType);  
 cout << " -> Погружена бочка: отсек " << compartmentNum  
 << ", тип " << fuelType << endl;  
 }  
 else if (op == '-') {  
 barge.unload(compartmentNum, fuelType);  
 cout << " -> Разгружена бочка: отсек " << compartmentNum  
 << ", тип " << fuelType << endl;  
 }  
 }

// Проверка состояния и вывод результата  
 if (barge.hasError()) {  
 cout << endl << "══════════════════════════════════════════════════" << endl;  
 cout << "Результат: Error" << endl;

// Детализация ошибки  
 cout << "Причины возможных ошибок:" << endl;  
 cout << " - Попытка разгрузки из пустого отсека" << endl;  
 cout << " - Несоответствие типа топлива" << endl;  
 cout << " - Некорректный номер отсека" << endl;  
 cout << " - Превышение лимита бочек (" << maxCapacity << ")" << endl;  
 cout << " - Баржа не пуста в конце операций" << endl;  
 }  
 else {  
 cout << endl << "══════════════════════════════════════════════════" << endl;  
 cout << "Результат: " << barge.getMaxBarrels() << endl;  
 cout << "Максимальное количество бочек: " << barge.getMaxBarrels() << endl;  
 cout << "Статус: успешное завершение" << endl;  
 }  
 }  
 // Обработка всех исключений  
 catch (const exception& e) {  
 cout << endl << "══════════════════════════════════════════════════" << endl;  
 cout << "Критическая ошибка: " << e.what() << endl;  
 cout << "Программа завершена" << endl;  
 }  
 return 0;  
}