МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт компьютерных наук и кибербезопасности

Направление 02.03.01 Математика и компьютерные науки

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

«Статический анализ кода»

Обучающийся: _____ Шклярова Ксения Алексеевна

Руководитель: _____ Курочкин Михаил Александрович

«_____»_____20___г.

Санкт-Петербург, 2025

Содержание

Введение

Статическое тестирование - процесс анализа программного кода без его выполнения. Целью статического тестирования является выявление недочетов на ранних стадиях разработки, что позволяет сэкономить время и ресурсы.

В данной лабораторной работе требуется провести статический анализ кода программы «Калькулятор «большой» конечной арифметики».

Для этого необходимо:

1. Изучить инструмент статического анализа кода Cppcheck 2.17.1;

2. Провести статический анализ при помощи Cppcheck 2.17.1;

3. Проанализировать результаты статического анализа кода программы и произвести исправление ошибок.

1 Описание тестируемой программы

1.1 Калькулятор «большой» конечной арифметики

Алгебраические структуры

В курсовой работе рассматривается конечное коммутативное кольцо с единицей $<M; +, *>$, на котором определены операции сложения и умножения, а так же определено действие вычитания «-» - малая конечная арифметика

M - носитель малой конечной арифметики, в данном случае M = $\{a, b, c, d, e, f, g, h\}$. Операции «+» и «*» определяются правилом «+1».

Конечное коммутативное кольцо с единицей $<M_8^8; +, *>$, на котором также определено действие «-» - большая конечная арифметика для восьмиразрядных чисел. Операции «+» и «*» определяются посредством таблиц для этих операций в малой конечной арифметике. Вычитание определяется при помощи уже известного сложения и таблиц для малой конечной арифметики.

Свойства операций над алгебраическими структурами

Из определения малой конечной арифметики:

1. Ассоциативность «+» и «*»

$$\forall x, y, z \in M : x + (y + z) = (x + y) + z, x * (y * z) = (x * y) * z$$

2. Коммутативность «+» и «*»

$$\forall x, y \in M : x + y = y + x, x * y = y * x$$

3. Дистрибутивность «+» относительно «*»

$$\forall x, y, z \in M : x * (y + z) = x * y + x * z$$

4. Существование нейтрального элемента по «+»

$$\exists\ a \in M : \forall x \in M\ \ x + a = x$$

5. Существование нейтрального элемента по «*»

$$\exists\ b \in M : \forall x \in M\ \ x * b = x$$

Следствием из этого будет: $\forall x \in M : x * a = a$. Составим отношение порядка на множестве M, которое определяется правилом «+1»: a $\rightarrow b \rightarrow c \rightarrow g \rightarrow d \rightarrow h \rightarrow f \rightarrow e$.

Можно составить таблицы для сложения (см. табл. 1), умножения (см. табл. 3) и для переноса разряда (см. табл. 2 и табл. 4).

$+$	a	b	c	d	e	f	g	h
a	a	b	c	d	e	f	g	h
b	b	c	g	h	a	e	d	f
c	c	g	d	f	b	a	h	e
d	d	h	f	a	g	c	e	b
e	e	a	b	g	f	h	c	d
f	f	e	a	c	h	d	b	g
g	g	d	h	e	c	b	f	a
h	h	f	e	b	d	g	a	c

Таблица 1. Поразрядное сложение

$+_n$	a	b	c	d	e	f	g	h
a	a	a	a	a	a	a	a	a
b	a	a	a	a	b	a	a	a
c	a	a	a	a	b	b	a	a
d	a	a	a	a	b	b	a	b
e	a	b	b	b	b	b	b	b
f	a	a	b	b	b	b	b	b
g	a	a	a	a	b	b	a	b
h	a	a	a	b	b	b	b	b

Таблица 2. Перенос разряда при сложении

$*$	a	b	c	d	e	f	g	h
a	a	a	a	a	a	a	a	a
b	a	b	c	d	e	f	g	h
c	a	c	d	a	f	d	f	c
d	a	d	a	a	d	a	d	d
e	a	e	f	d	b	c	h	g
f	a	f	d	a	c	d	c	f
g	a	g	f	d	h	c	b	e
h	a	h	c	d	g	f	e	b

Таблица 3. Поразрядное умножение

$*_n$	a	b	c	d	e	f	g	h
a	a	a	a	a	a	a	a	a
b	a	a	a	a	b	a	a	a
c	a	a	a	b	b	b	a	b
d	a	a	b	c	g	g	b	c
e	a	a	b	g	f	h	c	d
f	a	a	b	g	h	d	c	g
g	a	a	a	b	c	c	b	b
h	a	a	b	c	d	g	b	g

Таблица 4. Перенос разряда при умножении

Примеры вычислений

1. a + b = b;
2. b + f = f + b = e;
3. c + f = c + 1 + h = b + 1 + f = a (при этом будет перенос раряда, поэтому полный ответ - ba);
4. a ∗ ff = a;
5. b ∗ dd = dd;
6. $c*d = c*(1+g) = c+c*(1+c) = c+c+c(1+b) = c+c+c+c = a$ (при этом будет перенос раряда, поэтому полный ответ - ba);

1.2 Формальное описание

Название: Калькулятор «большой» конечной арифметики

Дано: строка A, являющаяся первым числом, строка B, являющаяся вторым числом, length_A - длина строки A, length_B - длина строки B.

Требуется: выполнить операции сложения, вычитания и умножения на основе заданной конечной арифметики.

Ограничения: A и B состоят из следующих символов: «a», «b», «c», «d», «e», «f», «g», «h»; максимальная длина строк A и B - 8 символов, минимальная длина A и B - 1 символ: $1 \leq length\_A \leq 8, 1 \leq length\_B \leq 8$.

Спецификация:

Входные данные	Выходные данные	Реакция программы
$length\_A = 0$ или $length\_B = 0$ или $length\_A > 8$ или $length\_B > 8$	Сообщение: Повторите ввод, размерность введенного значения не должна превышать 8 разрядов	Повторный запрос ввода от пользователя до тех пор, пока не будет введено корректное значение
A = abdfe, B = hcb, length_A = 5, length_B = 3	A + B = ccba, A - B = edf, B - A = -edf, A * B = baghhde	Завершение работы с кодом 0
A = bghcd, B = efffe, length_A = 5, length_B = 5	A + B = bbcdbg, A - B = -fgbdg, B - A = fgbdg, A * B = переполнение	Завершение программы с кодом 0
A = efabcd, B = gg, length_A = 6, length_B = 2	A + B = efabhe, A - B = efaaeb, B - A = -efaaeb, A * B = gcbcdggd	Завершение программы с кодом 0

Таблица 1. Результаты работы программы

2 Статический анализ кода

Статический анализ кода выполняется без запуска программы. Он анализирует исходный код, чтобы выявить потенциальные ошибки, уязвимости и нарушения стандартов кодирования. Этот вид анализа позволяет разработчикам обнаруживать проблемы на ранних стадиях разработки, что может существенно сократить затраты на исправление проблем в будущем.

Статический анализ также может включать проверку стиля кода, что помогает поддерживать единообразие в проекте. Это особенно важно в командах, где несколько разработчиков работают над одним проектом. Инструменты для статического анализа могут быть настроены для проверки соответствия кода определенным стандартам, что помогает избежать распространенных проблем и улучшить читаемость кода.

Многие инструменты для статического анализа могут быть интегрированы в процесс сборки, что позволяет автоматизировать проверку кода. Это обеспечивает постоянный контроль качества и помогает разработчикам сосредоточиться на написании кода, а не на его проверке.

Виды статического тестирования

Статический анализ кода (Static Code Analysis)

Этот вид статического тестирования включает в себя анализ исходного кода. Основная цель — выявление потенциальных проблем, неправильных практик, структурных аномалий и нарушений стандартов кодирования. Инструменты, такие как Lint, Pylint и ESLint, помогают автоматизировать этот процесс.

Обзоры кода (Code Reviews)

Этот вид статического тестирования включает в себя анализ кода членами команды разработки или экспертами. Обзоры кода позволяют выявлять проблемы и несоответствия стандартам. Они также способствуют обмену знаниями и опытом между членами команды.

Анализ архитектуры (Architecture Analysis)

При этом виде тестирования анализируется архитектура ПО, включая структуру, зависимости между компонентами и соответствие архитектурным принципам. Это позволяет выявить проблемы, связанные с проектированием системы.

3 Описание Cppcheck

Cppcheck — статический анализатор кода для языка C/C++, предназначенный для поиска ошибок, которые не обнаруживаются компиляторами. Цель состоит в том, чтобы обнаружить только реальные ошибки в коде и сгенерировать как можно меньше ложных срабатываний (ошибочных предупреждений).

Большинство проверок cppchck по умолчанию не включает. Среди них следующие категории проверок, каждая из которых может включаться/выключаться независимо:

• error - явные ошибки, которые анализатор считает критическими и обычно они приводят к багам (включено по умолчанию);

• warning — предупреждения, здесь даются сообщения о небезопасном коде;

• style — стилистические недочеты, сообщения появляются в случае неаккуратного кодирования;

• performance — проблемы производительности, здесь cppcheck предлагает варианты, как сделать код быстрее;

• portability — ошибки совместимости, обычно связано с различным поведением компиляторов или систем разной разрядности;

• information — информационные сообщения, возникающие в ходе проверки и не связанные с ошибками в коде;

• unusedFunction - попытка вычислить неиспользуемые функции, не умеет работать в многопоточном режиме;

• missingInclude — проверка на недостающий include (например, используем random, а подключить stdlib.h забыли).

Включаются проверки параметром - enable, список категорий проверок перечисляется через запятую. Либо можно использовать ключевое слово all, которое включает все перечисленные проверки. Внутри категорий выполняются отдельные правила проверки, каждое из которых отвечает за выявление определённых ошибок или недочетов в коде.

Рассмотрим аргументы в командной строке:

• -j — параметр, позволяющий запускать проверку в многопоточном режиме. В качестве параметра передаётся количество процессоров.

• -q — тихий режим. По умолчанию cppcheck выдаёт информационные сообщения о ходе проверки. Данный параметр полностью выключает информационные сообщения, остаются только сообщения об ошибках.

• -f или -force — включить перебор всех вариантов директив ifdef.

• v — режим отладки — cppcheck выдаёт внутреннюю информацию о ходе проверки.

•-xml — выводить результат проверки в формате XML.

• template=gcc — выводить ошибки в формате предупреждений компилятора dcc (удобно для интеграции с IDE, поддерживающей такой компилятор).

• suppress — режим подавления ошибок с указанными идентификаторами.

• -h — выдаёт справку по всем параметрам на английском языке.

3.1 Группы правил

Cppcheck проверяет следующие группы правил:

1. 64-bit portability - некорректные операции с указателями и целыми числами на 64-битных системах.

2. Assert - побочные эффекты в assert, вызывающие различное поведение в debug release.

3. Auto Variables - ошибки, связанные с временем жизни локальных переменных и указателей на них.

4. Boolean - некорректные операции с boo1: инкремент, побитовые операции, сравнение с числами.

5. Bounds checking - выход за границы массива, переполнение указателей, ошибки strncat ().

6. Check function usage - отсутствие return в non-void функциях, игнорирование возвращаемого значения.

7. Class - ошибки в классах: неинициализированные члены, отсутствие конструктора копирования, использование memset на классах, неверное наследование.

8. Condition - ошибки в условиях: бессмысленные сравнения, лишние проверки,неправильные if/else.

9. Exception Safety - некорректная работа с исключениями: throw в деструкторах, неправильный catch, утечка исключения.

10. IO using format string - ошибки форматирования в printf/scanf, работа с файлами после закрытия.

11. Leaks (auto variables) - потерянные указатели на локальные переменные, двойное освобождение памяти.

12. Memory leaks (address not taken) - выделенная память не используется.

13. Memory leaks (class variables) - забыли освободить память в деструкторе класса.

14. Memory leaks (function variables) - забыли освободить память в функции перед выходом.

15. Memory leaks (struct members) - забыли освободить динамическую память в структуре.

16. Null pointer - разыменование nullptr, арифметика с nullptr.

17. Other - деление на 0, неопределенное поведение, некорректные побитовые операции, потерянные goto.

18. STL usage - ошибки работы с STL: использование невалидных итераторов, утечки памяти.

19. Sizeof - неправильное использование sizeof (), вычисления внутри sizeof.

20. String - некорректное использование строковых функций (sprintf, strncmp).

21. Type - ошибки приведения типов, переполнения, опасные преобразования знака.

22. Uninitialized variables - использование неинициализированных переменных.

23. Unused functions - функции, которые нигде не вызываются.

24. UnusedVar - переменные, которые не используются или неинициализированы.

25. Using postfix operators - предупреждение при использовании i++, если ++i было бы эффективнее.

26. Vaarg - ошибки работы с переменным числом аргументов: va_start, va_end, передача reference.

Для того, чтобы включить проверку правил, необходимо использовать комбинацию из параметров -enable и -supress, где - enable отвечает за включение всех правил из заданной категории, а -supress - за подавление заданных правил из категории.

Разбор CppCheck правил из группы «STL usage»

CppCheck предоставляет 13 проверок из группы «STL usage» для обнаружения неверного использования функций из стандартной библиотеки шаблонов. Рассмотрим следующие правила:

- **Rule:** `dereferencingAnErasedIterator`
  - **Severity:** Error
  - **Category:** STL usage
- **Rule:** `usingAutoPointer`
  - **Severity:** Warning
  - **Category:** STL usage

Правило «dereferencingAnErasedIterator»

В STL все действия с контейнерами как правило происходят при помощи итераторов (специальный proxy-объект, который предоставляет доступ к элементам контейнера и интерфейс для итерации по элементам). Однако итераторы не устойчивы, что означает, что если итератор был создан, а затем контейнер был модифицирован, то итератор может стать невалидным и такой итератор использовать нельзя. Для разных контейнеров это может работать по-разному. Например, для `std::vector` после вставки нового элемента все итераторы становятся невалидными, для `std::list` после вставки все итераторы остаются корректными.

Данное правило позволяет обнаружить проблемы, связанные с использованием итераторов после того, как контейнер был модифицирован.

Правило «usingAutoPointer»

STL предоставляет несколько классов для автоматического управления ресурсами, такие как `std::unique_ptr` (для уникального владения) и `std::shared_ptr` (для раздельного владения).

Однако оба эти класса появились начиная со стандарта C++11, до этого начиная со стандарта C++03 существовал только один подобный класс – `std::auto_ptr` (который предоставлял уникальное владение), однако он считается устаревшим, а его использование может привести к проблемам, связанным с управлением ресурсами, поэтому его использование считается нежелательным.

Данное правило отслеживает использование данного класса.

Разбор CppCheck правил из группы «Bounds checking»

Данная группа правил проверяет правильное использование массивов и строк. Мы рассмотрим следующие два правила:

- **Rule:** `arrayIndexOutOfBounds`
  - **Severity:** Error
  - **Category:** Bounds checking
- **Rule:** `negativeMemoryAllocationSize`
  - **Severity:** Error
  - **Category:** Bounds checking

Правило arrayIndexOutOfBounds

Данное правило работает для статических массивов либо для динамически аллоцированных массивов, у которых известен размер на этапе компиляции. Оно проверяет, что при использовании массива не происходит выход за его пределы.

Правило negativeMemoryAllocationSize

Аллокация динамического и статического массива с отрицательным размером является проблемой. Однако компилятор может определить это только в том случае, если значение было явно указано при аллокации. В случае, если в качестве размера указана переменная, компилятор не сможет определить это как проблему. Но данное правило позволяет определить это в том случае, если значение переменной можно определить статически.

4 Статический анализ кода программы

Для проведения статического анализа кода программы необходимо запустить Cppcheck в командной строке:

cppcheck -q –enable=all –suppress=missingIncludeSystem –std=c++20 –check-level=exhaustive "/Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp"

Статический анализ выполнялся со следующими параметрами:

- -q - отключение вывода информационных сообщений;
- enable—all - активация всех возможных проверок;
- -suppress-missingIncludeSystem - игнорирование предупреждений о системных заголовках;
- std=c++20 - установление используемого стандарта C++;
- -level exhaustive - проведение полного анализа. По умолчанию уровень про-оверки - normal;
- "/Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp путь к тестируемой программе.

4.1 Результат первого запуска Cppcheck

В результате первого запуска статического анализатора Cppcheck были выявлены 2 стилистических замечания и 11 потенциальных проблем с производительностью, представленные в листинге 1.

```
1  /Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp:357:19: style:
       Condition 'a<=b' is always true [knownConditionTrueFalse]
2              if (a <= b){flag = false; break;}
3                    ^
4  /Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp:640:13: style:
       Consecutive return, break, continue, goto or throw statements are unnecessary. [
       duplicateBreak]
5              break;
6              ^
7  /Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp:181:19:
       performance: Function parameter 'c' should be passed by const reference. [passedByValue]
8  string sum(string c, string s){
9                    ^
10 /Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp:181:29:
       performance: Function parameter 's' should be passed by const reference. [passedByValue]
11 string sum(string c, string s){
12                             ^
13 /Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp:284:33:
       performance: Function parameter 's' should be passed by const reference. [passedByValue]
14 string minus__(string c, string s){
15                                  ^
16 /Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp:337:19:
       performance: Function parameter 'c' should be passed by const reference. [passedByValue]
17 bool more_(string c, string s){
```

```
                 ^
/Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp:337:29:
    performance: Function parameter 's' should be passed by const reference. [passedByValue]
bool more_(string c, string s){
                            ^
/Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp:364:18:
    performance: Function parameter 'c' should be passed by const reference. [passedByValue]
bool ravn(string c, string s){
                 ^
/Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp:364:28:
    performance: Function parameter 's' should be passed by const reference. [passedByValue]
bool ravn(string c, string s){
                            ^
/Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp:391:22:
    performance: Function parameter 'c' should be passed by const reference. [passedByValue]
string minus_(string c, string s){
                      ^
/Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp:391:32:
    performance: Function parameter 's' should be passed by const reference. [passedByValue]
string minus_(string c, string s){
                                ^
/Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp:445:21:
    performance: Function parameter 'c' should be passed by const reference. [passedByValue]
string umnoj(string c, string s){
                     ^
/Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp:445:31:
    performance: Function parameter 's' should be passed by const reference. [passedByValue]
string umnoj(string c, string s){
```

4.2 Анализ выявленных недочетов и их исправление

Недочет: Condition 'a<=b' is always true

Избыточное условие в функции more_. Условие a <= b всегда истинно после проверки a > b

Исправление:

```
//до исправления
if (a > b) { flag = true; break; }
if (a <= b) { flag = false; break; }  // Условие всегда true, если a > b ложно

//после исправления
if (a > b) { flag = true; break; }
if (a < b) { flag = false; break; }
```

Недочет: Consecutive return, break, continue, goto or throw statements are unnecessary

Лишний break в функции get_numb

Исправление: Удалить break

```
//до исправления
for (string str; ; getline(cin, str)){
```

```
    if ((isMyLetter(str)) && (str.size() <= 8)){
            return str;
            break;}

//после исправления
for (string str; ; getline(cin, str)){
    if ((isMyLetter(str)) && (str.size() <= 8)){
            return str;}
```

Недочет: Function parameter 'c' should be passed by const reference.

Передача string по значению 2 параметра в функции sum(), параметр s в minus__(), 2 параметра в more_(), 2 параметра в ravn(), minus_ и umnoj.

Исправление:

```
//до исправления
string sum(string c, string s)
string minus\_\_(string c, string s)
bool more_(string c, string s)
bool ravn(string c, string s)
string minus_(string c, string s)
string umnoj(string c, string s)

//после исправления
string sum(const string& c, const string& s)
string minus\_\_(string c, const string& s)
bool more_(const string& c, const string& s)
bool ravn(const string& c, const string& s)
string minus_(const string& c, const string& s)
string umnoj(const string& c, const string& s)
```

После исправления всех недочетов Cppcheck был запущен следующей командой:

cppcheck -q –enable=all –suppress=missingIncludeSystem –std=c++20 –check-level=exhaustive "/Users/kseniasklarova/Documents/matrix_opredelitel/matrix_opredelitel/main.cpp"&& echo 0 || echo 1

В результате было выведено сообщение «0», что означает, что в ходе проверки ошибок не было выявлено. Также не было выведено никаких предупреждений см. рис. 1.

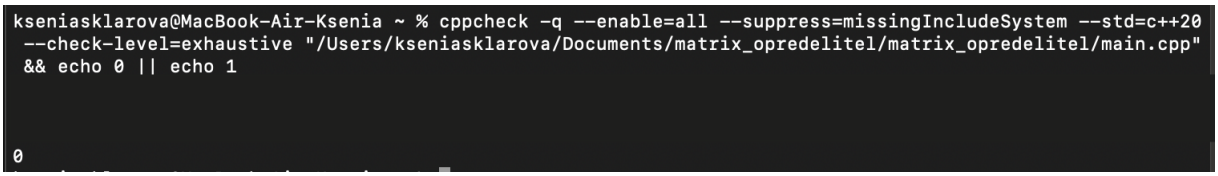

Рис. 1. Результат запуска Cppcheck после исправлений недочетов

Исправленный код представлен в Приложении В.

5 Заключение

В рамках лабораторной работы был исследован инструмент статического анализа кода Cppcheck (версия 2.17.1). Проведенный анализ кода программы «Калькулятор большой конечной арифметики» выявил 2 стилистических недочета и 11 потенциальных проблем с производительностью, связанных со следующими правилами:

• [knownConditionTrueFalse] - условие всегда истинно/ложно из-за логической ошибки;

• [duplicateBreak] - обнаружен недостижимый код после return или break, который следует удалить;

• [passedByValue] - объекты передаются по значению (с копированием), что менее эффективно, чем использование const&.

Ключевые недостатки включали избыточное условие (требующее замены на a < b), излишний оператор break, а также неоптимальную передачу строковых параметров по значению вместо константной ссылки в функциях sum(), minus__(), more_(), ravn(), minus_() и umnoj(), что приводило к ненужному копированию данных. Для повышения производительности было применено следующее решение: замена объявлений string param на const string& param в указанных функциях. Основными причинами выявленных недочетов стали логические ошибки в условиях и неэффективная передача объектов.

Все обнаруженные проблемы были успешно устранены, что подтверждено повторным запуском Cppcheck, не выявившим новых предупреждений.

В ходе работы было установлено, что статический анализ с помощью Cppcheck позволил обнаружить недочеты, пропущенные при ручной инспекции кода. В то же время, ручная инспекция помогла выявить проблемы, не обнаруженные Cppcheck (например, использование неинформативных имен переменных, снижающее читаемость кода).

На основании полученных результатов можно сделать вывод, что эти методы тестирования дополняют друг друга и будет эффективнее использовать их совместно.

Список литературы

[1] Майерс, Г. Искусство тестирования программ / Г. Майерс, Т. Баджетт, К. Санд-лер. — Изд. 3-е. — Санкт-Петербург : Диалектика, 2012. — 272 с.

[2] Cppcheck manual (Электронный ресурс).- URL: http://cppcheck.net/manual.pdf (дата обращения: 01.04.2025)

[3] Официальный сайт Cppcheck|Электронный ресурс)- URL: https://sourceforge.net/p/cppcheck/wiki/ListOfChecks/ (дата обращения: 01.04.2025)

Приложение А. Исходный код программы

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;

vector<string> my_vect{"a", "b", "c", "g", "d", "h", "f", "e"};
vector<char> my_vectt{'a', 'b', 'c', 'g', 'd', 'h', 'f', 'e'};

bool isMyLetter(string const& str){
    return !str.empty() && str.find_first_not_of("abcdefgh") == string::npos;
}

char plus_1(char c){
    char s = 0;
    for (int i = 0; i < 8; i++){
        if (c == my_vectt[i]){
            if (c == my_vectt[7]){
                s = my_vectt[0];
                break;
            }
            s = my_vectt[i+1];
            break;
        }}
    return s;
}

string sum(string c, string s){
    vector<char> my_v(16, 'a');
    string ss;
    if (c.size() == s.size()){
        for (int i = 0; i < max(c.size(), s.size()); i++){
            char b = my_v[i];
            my_v[i] = plus_razr(c[c.size()-1-i], my_v[i]);
            my_v[i] = plus_razr(s[s.size()-1-i], my_v[i]);
            if (razr3(c[c.size()-1-i], s[s.size()-1-i], b)){
                my_v[i+1] = plus_razr(my_vectt[1], my_vectt[0]);
            }}}
    if (c.size() > s.size()){
        for (int i = 0; i < s.size(); i++){
            char b = my_v[i];
            my_v[i] = plus_razr(c[c.size()-1-i], my_v[i]);
            my_v[i] = plus_razr(s[s.size()-1-i], my_v[i]);
            if (razr3(c[c.size()-1-i], s[s.size()-1-i], b)){
                my_v[i+1] = plus_razr(my_vectt[1], my_vectt[0]);
                }}
        for (int i = s.size(); i < c.size(); i++){
            char b = my_v[i];
            my_v[i] = plus_razr(c[c.size()-1-i], my_v[i]);
            if (razr(c[c.size()-1-i], b)){
```

```
                my_v[i+1] = plus_razr(my_vectt[1], my_vectt[0]);
            }}}
    if (c.size() < s.size()){
        for (int i = 0; i < c.size(); i++){
            char b = my_v[i];
            my_v[i] = plus_razr(c[c.size()-1-i], my_v[i]);
            my_v[i] = plus_razr(s[s.size()-1-i], my_v[i]);
            if (razr3(c[c.size()-1-i], s[s.size()-1-i], b)){
                my_v[i+1] = plus_razr(my_vectt[1], my_vectt[0]);
            }}
        for (int i = c.size(); i < s.size(); i++){
            char b = my_v[i];
            my_v[i] = plus_razr(s[s.size()-1-i], my_v[i]);
            if (razr(s[s.size()-1-i], b)){
                my_v[i+1] = plus_razr(my_vectt[1], my_vectt[0]);
            }}}
    int f = 9;
    for (int i = 0; i < my_v.size(); i++){
        if (my_v[i] != 'a'){
            f = i;}}
    if (f == 9){
        ss.push_back('a');}
    else{
        for (int i = f; i >= 0; i--){

            ss.push_back(my_v[i]);}}
    return ss;
}

string minus__(string c, string s){
    vector<char> vect(c.size(), 'a');
    int a = 0, b = 0;
    for (int i = 0; i < s.size(); i++){
        for (int j = 0; j < 8; j++){
            if (c[c.size()-1-i] == my_vectt[j]){
                a = j;
            }
            if (s[s.size()-1-i] == my_vectt[j]){
                b = j;
            }}
        if (a >= b){
            vect[i] = minus_razr(c[c.size()-1-i], s[s.size()-1-i]);
        }
        else{
            char d = my_vectt[7];//перенос разряда
            vect[i] = plus_1(plus_razr(minus_razr(d, s[s.size()-1-i]), c[c.size()-1-i]));
            for (int j = c.size()-i-2; j >= 0; j--){
                if (c[j] != my_vectt[0]){
                    c[j] = minus_1(c[j]);
                    break;
                }
```

```
                else{
                    c[j] = my_vectt[7];
                }}}}
    for (int i = s.size(); i < c.size(); i++){
        vect[i] = c[c.size()-1-i];
    }
    string ss;
    int f = 0;
    for (int i = 0; i < vect.size(); i++){
        if (vect[i] != 'a'){
            f = i;
        }}
    for (int i = f; i >= 0; i--){
        ss.push_back(vect[i]);
    }
    return ss;
}

bool more_(string c, string s){
    bool flag = false;
    int size_ = c.size() - s.size();
    if (size_ < 0){
        flag = false;}
    if (size_ > 0){
        flag = true;}
    if (size_ == 0){
        for (int i = 0; i < c.size(); i++){
            int a = 0, b = 0;
            for (int j = 0; j < 8; j++){
                if (c[i] == my_vectt[j]){ a = j;}
                if (s[i] == my_vectt[j]){ b = j;}
            }
            if (a > b){
                flag = true;
                break;
            }
            if (a <= b){flag = false; break;}
        }}
    return flag;
}

bool ravn(string c, string s){
    bool flag = true;
    int size_ = c.size()-s.size();
    make_polozh(s).size();
    if (size_ < 0){
        flag = false;}
    if (size_ > 0){
        flag = false;}
    if (size_ == 0){
        for (int i = 0; i < c.size(); i++){
```

```
152             int a = 0, b = 0;
153             for (int j = 0; j < 8; j++){
154                 if (c[i] == my_vectt[j]){ a = j;}
155                 if (s[i] == my_vectt[j]){ b = j;}
156             }
157             if (c.size() == s.size() == 1 && a == b){flag = true; break;}
158             if (a > b){
159                 flag = false;
160                 break;
161             }
162             if (a < b){flag = false; break;}
163         }}
164     return flag;
165 }
166 
167 string minus_(string c, string s){
168     string str = "a";
169     if (ravn(c, s)){
170         str = sum(str, my_vect[0]);
171     }
172     if (more_(c, s)){
173         str = minus__(c, s);
174     }
175     if (more_(s, c)){
176         string strr;
177         strr = minus__(s, c);
178         str = "-";
179         for (int i = 1; i < strr.size()+1; i++){
180             str.push_back(strr[i-1]);
181         }}
182     return str;
183 }
184 
185 string umnoj(string c, string s){
186     if (c == my_vect[0] || s == my_vect[0]){
187         return my_vect[0];
188     }
189     else{
190         vector<string> vect;
191         for (int i = 0; i < s.size(); i++){
192             vector<char> v(16, 'a');
193             for (int j = 0; j < c.size(); j++){
194                 int t = 0;
195                 for (int h = 0; h < 8; h++){
196                     if (v[j+i] == my_vectt[h]){
197                         t = h;
198                         break;
199                     }}
200                 string ff = sum(umnoj_razr(s[s.size()-1-i], c[c.size()-1-j]), my_vect[t]);
201                 if (ff.size() == 1){
202                     v[j+i] = ff[0];
```

```
203                 }
204                 else{
205                     v[j+i] = ff[1];
206                     v[j+1+i] = ff[0];
207                 }}
208             string ss;
209             for (int j = v.size()-1; j >= 0; j--){
210                 ss.push_back(v[j]);
211             }
212             vect.push_back(ss);
213         }
214         string str = "a";
215         for (int i = 0; i < vect.size(); i++){
216             str = sum(str, vect[i]);
217         }
218         return str;
219     }}
220
221 string get_numb(){
222     for (string str; ; getline(cin, str)){
223         if ((isMyLetter(str)) && (str.size() <= 8)){
224             return str;
225             break;
226         }
227         else {
228             cout << "Введите значение, его размерность не должна превышать 8 разрядов" << endl
                    ;
229         }}}
230
231
232 int main() {
233     string A;
234     string B;
235     cout << "Выберете значение A" << endl;
236     A = get_numb();
237     cout << "Выберете значение B" << endl;
238     B = get_numb();
239     cout << "A + B = ";
240     if (sum(A,B).size() > 8){
241         cout << "переполнение" << endl;
242     }
243     else{
244         cout << sum(A, B) << endl;}
245     cout << "A - B = ";
246     if (minus_(A,B).size() > 8){
247         cout << "переполнение" << endl;
248     }
249     else{
250         cout << minus_(A, B) << endl;}
251     cout << "B - A = ";
252     if (minus_(B,A).size() > 8){
```

```
253         cout << "переполнение" << endl;
254     }
255      else{
256          cout << minus_(B, A) << endl;}
257     cout << "A * B = ";
258     if (umnoj(A,B).size() > 8){
259         cout << "переполнение" << endl;
260      }
261     else{
262         cout << umnoj(A, B) << endl;}
263     return 0;
264 }
```

Приложение В. Исправленный код программы

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;

const vector<string> my_vect{"a", "b", "c", "g", "d", "h", "f", "e"};
const vector<char> my_vectt{'a', 'b', 'c', 'g', 'd', 'h', 'f', 'e'};

bool isMyLetter(const string& str) {
    return !str.empty() && str.find_first_not_of("abcdefgh") == string::npos;
}

char plus_1(char c) {
    char s = 0;
    for (int i = 0; i < 8; i++) {
        if (c == my_vectt[i]) {
            if (c == my_vectt[7]) {
                s = my_vectt[0];
                break;
            }
            s = my_vectt[i+1];
            break;
        }}
    return s;
}

string sum(const string& c, const string& s) {
    vector<char> my_v(16, 'a');
    string ss;
    if (c.size() == s.size()) {
        for (int i = 0; i < max(c.size(), s.size()); i++) {
            char b = my_v[i];
            my_v[i] = plus_razr(c[c.size()-1-i], my_v[i]);
            my_v[i] = plus_razr(s[s.size()-1-i], my_v[i]);
            if (razr3(c[c.size()-1-i], s[s.size()-1-i], b)) {
                my_v[i+1] = plus_razr(my_vectt[1], my_vectt[0]);
            }}}
    if (c.size() > s.size()) {
        for (int i = 0; i < s.size(); i++) {
            char b = my_v[i];
            my_v[i] = plus_razr(c[c.size()-1-i], my_v[i]);
            my_v[i] = plus_razr(s[s.size()-1-i], my_v[i]);
            if (razr3(c[c.size()-1-i], s[s.size()-1-i], b)) {
                my_v[i+1] = plus_razr(my_vectt[1], my_vectt[0]);
            }}
        for (int i = s.size(); i < c.size(); i++) {
            char b = my_v[i];
            my_v[i] = plus_razr(c[c.size()-1-i], my_v[i]);
            if (razr(c[c.size()-1-i], b)) {
```

```
                my_v[i+1] = plus_razr(my_vectt[1], my_vectt[0]);
            }}}
    if (c.size() < s.size()) {
        for (int i = 0; i < c.size(); i++) {
            char b = my_v[i];
            my_v[i] = plus_razr(c[c.size()-1-i], my_v[i]);
            my_v[i] = plus_razr(s[s.size()-1-i], my_v[i]);
            if (razr3(c[c.size()-1-i], s[s.size()-1-i], b)) {
                my_v[i+1] = plus_razr(my_vectt[1], my_vectt[0]);
            }}
        for (int i = c.size(); i < s.size(); i++) {
            char b = my_v[i];
            my_v[i] = plus_razr(s[s.size()-1-i], my_v[i]);
            if (razr(s[s.size()-1-i], b)) {
                my_v[i+1] = plus_razr(my_vectt[1], my_vectt[0]);
            }}}
    int f = 9;
    for (int i = 0; i < my_v.size(); i++) {
        if (my_v[i] != 'a') {
            f = i;
        }}
    if (f == 9) {
        ss.push_back('a');
    }
    else {
        for (int i = f; i >= 0; i--) {
            ss.push_back(my_v[i]);
        }}
    return ss;
}

string minus__(const string& c, const string& s) {
    vector<char> vect(c.size(), 'a');
    int a = 0, b = 0;
    for (int i = 0; i < s.size(); i++) {
        for (int j = 0; j < 8; j++) {
            if (c[c.size()-1-i] == my_vectt[j]) {
                a = j;
            }
            if (s[s.size()-1-i] == my_vectt[j]) {
                b = j;
            }}
        if (a >= b) {
            vect[i] = minus_razr(c[c.size()-1-i], s[s.size()-1-i]);
        }
        else {
            char d = my_vectt[7];
            vect[i] = plus_1(plus_razr(minus_razr(d, s[s.size()-1-i]), c[c.size()-1-i]));
            for (int j = c.size()-i-2; j >= 0; j--) {
                if (c[j] != my_vectt[0]) {
                    c[j] = minus_1(c[j]);
```

```
                    break;
                }
                else {
                    c[j] = my_vectt[7];
                }}}}
    for (int i = s.size(); i < c.size(); i++) {
        vect[i] = c[c.size()-1-i];
    }
    string ss;
    int f = 0;
    for (int i = 0; i < vect.size(); i++) {
        if (vect[i] != 'a') {
            f = i;
        }}
    for (int i = f; i >= 0; i--) {
        ss.push_back(vect[i]);
    }
    return ss;
}

bool more_(const string& c, const string& s) {
    bool flag = false;
    int size_ = c.size() - s.size();
    if (size_ < 0) {
        flag = false;
    }
    if (size_ > 0) {
        flag = true;
    }
    if (size_ == 0) {
        for (int i = 0; i < c.size(); i++) {
            int a = 0, b = 0;
            for (int j = 0; j < 8; j++) {
                if (c[i] == my_vectt[j]) { a = j; }
                if (s[i] == my_vectt[j]) { b = j; }
            }
            if (a > b) {
                flag = true;
                break;
            }
            if (a < b) {
                flag = false;
                break;
            }}}
    return flag;
}

bool ravn(const string& c, const string& s) {
    bool flag = true;
    int size_ = c.size()-s.size();
    if (size_ != 0) {
```

```
        flag = false;
    }
    else {
        for (int i = 0; i < c.size(); i++) {
            int a = 0, b = 0;
            for (int j = 0; j < 8; j++) {
                if (c[i] == my_vectt[j]) { a = j; }
                if (s[i] == my_vectt[j]) { b = j; }
            }
            if (a != b) {
                flag = false;
                break;
            }}}
    return flag;
}

string minus_(const string& c, const string& s) {
    string str = "a";
    if (ravn(c, s)) {
        str = sum(str, my_vect[0]);
    }
    else if (more_(c, s)) {
        str = minus__(c, s);
    }
    else {
        string strr = minus__(s, c);
        str = "-";
        for (int i = 1; i < strr.size()+1; i++) {
            str.push_back(strr[i-1]);
        }}
    return str;
}

string umnoj(const string& c, const string& s) {
    if (c == my_vect[0] || s == my_vect[0]) {
        return my_vect[0];
    }
    else {
        vector<string> vect;
        for (int i = 0; i < s.size(); i++) {
            vector<char> v(16, 'a');
            for (int j = 0; j < c.size(); j++) {
                int t = 0;
                for (int h = 0; h < 8; h++) {
                    if (v[j+i] == my_vectt[h]) {
                        t = h;
                        break;
                    }}
                string ff = sum(umnoj_razr(s[s.size()-1-i], c[c.size()-1-j]), my_vect[t]);
                if (ff.size() == 1) {
                    v[j+i] = ff[0];
```

```
                }
                else {
                    v[j+i] = ff[1];
                    v[j+1+i] = ff[0];
                }}
            string ss;

            for (int j = v.size()-1; j >= 0; j--) {
                ss.push_back(v[j]);
            }
            vect.push_back(ss);
        }
        string str = "a";
        for (int i = 0; i < vect.size(); i++) {
            str = sum(str, vect[i]);
        }
        return str;
    }}

string get_numb() {
    for (string str; ; getline(cin, str)) {
        if ((isMyLetter(str)) && (str.size() <= 8)) {
            return str;
        }
        else {
            cout << "Введите значение, его размерность не должна превышать 8 разрядов" << endl
                ;
        }}}

int main() {
    string A;
    string B;
    cout << "Выберете значение A" << endl;
    A = get_numb();
    cout << "Выберете значение B" << endl;
    B = get_numb();
    cout << "A + B = ";
    if (sum(A,B).size() > 8) {
        cout << "переполнение" << endl;
    }
    else {
        cout << sum(A, B) << endl;
    }
    cout << "A - B = ";
    if (minus_(A,B).size() > 8) {
        cout << "переполнение" << endl;
    }
    else {
        cout << minus_(A, B) << endl;
    }
    cout << "B - A = ";
```

```
253     if (minus_(B,A).size() > 8) {
254         cout << "переполнение" << endl;
255     }
256     else {
257         cout << minus_(B, A) << endl;
258     }
259     cout << "A * B = ";
260     if (umnoj(A,B).size() > 8) {
261         cout << "переполнение" << endl;
262     }
263     else {
264         cout << umnoj(A, B) << endl;
265     }
266     return 0;
267 }
```