МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ образовательное учреждениевысшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра вычислительной техники

**ОТЧЁТ по ПРАКТИЧЕСКОЙ работе**

**по дисциплине «ИНФОРМАТИКА»**

**«**Анализ программ, функции**»**

|  |  |
| --- | --- |
| Факультет: АВТ  Группа: ДТ-460а  Студент: Дроздов Иван Сергеевич | Преподаватель:  Копылова Оксана Андреевна |

Новосибирск, 2025 г.

Содержание

Задание 3

Теоретические сведения 3

Программа №13 3

Пример работы программы 3

Ошибки и неточности 3

Выводы 3

Список литературы 4

Приложение А. Исходный код программы 5

# Задание

Из заданий 13-58 выдается 5 номеров фрагментов кода. Понять, что происходит в программе, что является результатом ее работы, прокомментировать код, указав все стандартные решения (стандартные решения см. в разделе 2.4 cprog).

Каждый фрагмент оформить в виде трех функций. В первой функции – передача параметров по значению, во второй – по ссылке, в третьей – по указателю. Рядом с функцией в комментариях указать, каким способом передаются переменные и для чего эти переменные нужны. Передать в функции массив. В main() каждую функцию вызвать по 2 раза с разными входными фактическими параметрами.

В main одновременный вывод всех результатов в виде “F13\_val()” (по значению), “F13\_ref() ” (по ссылке), “F13\_pointer()”(по указателю) – номер функции\_номер вызова – результат.

//--------------------------------------------------------23

for (j=0,i=0; i<n ; i++){

for (m=2; m<A[i]; m++)

{ if (A[i]%m==0) break; }

if (m==A[i]) B[j++]=A[i];

}

B[j]=0;   
//--------------------------------------------------------26

for (j=0,i=0; i<n ; i++){

for (s=0,m=2; m<A[i]; m++)

if (A[i]%m==0) { s=1; break; }

if (s==0) {

for (j=i; j<n-1;j++) A[j]=A[j+1];

n--;

i--;

}}

//--------------------------------------------------------28

v=A[0]+1;

do {

v--;

for (i=0,s=0;i<n;i++)

if (A[i]%v!=0) { s=1; break; }

} while(s==1);

//--------------------------------------------------------34

for (k=0, m=1; m <= n; k++, m = m \* 2);

return k-1; }

//--------------------------------------------------------35

for (i=0,j=n-1; i < j; i++,j--)

{ k = c[i]; c[i] = c[j]; c[j] = k; }

# Теоретические сведения

Указатель (pointer) - переменная, которая хранит адрес сущностей (т.е. других переменных любого типа, будь то структура, или массив), и над которой возможно выполнять операцию разыменования (dereferencing). Адрес обычно выражен целым положительным числом. Диапазон адресов зависит от архитектуры компьютера.

Ссылка (reference) — это объект, который ссылается на другой объект без копирования самого объекта. Ссылки используются для предоставления альтернативного имени уже существующему объекту. В отличие от указателей, ссылка должна быть инициализирована сразу при объявлении и не может изменять своё значение после этого (ссылаться на другой объект).

**1. Передача по значению**

При передаче параметра по значению создается копия исходной переменной, которую функция получает. Любые изменения этой копии внутри функции никак не влияют на оригинальную переменную вне функции.

**2. Передача по ссылке**

Передача по ссылке позволяет функции получить доступ непосредственно к оригиналу переменной, передавая её адрес. Таким образом, любые изменения параметра внутри функции будут отражены и снаружи функции.

**3. Передача по указателю**

Указатели позволяют передавать адрес переменной в функцию. Внутри функции мы можем изменить саму переменную, используя операцию разыменования (\*). Указатели дают больше гибкости, позволяя обращаться к памяти напрямую, однако требуют аккуратного обращения.

# Программа

*int main() {*

vector<int> A1 = {10, 7, 15, 3, 9}; // Первый набор данных

vector<int> A2 = {20, 13, 25, 5, 17}; // Второй набор данных

int n1 = 5, n2 = 5; // Размеры массивов

int result2, result3; // Переменные для хранения результатов

// Числа для вызова функций

int num1 = 7, num2 = 10;

// ФУНКЦИЯ F23: Проверка на простоту

printf("---------------------------------------------------------\n");

printf("F23\_val(%d) - %d\n", num1, F23\_val(num1)); // Первый вызов

printf("F23\_val(%d) - %d\n", num2, F23\_val(num2)); // Второй вызов

F23\_ref(num1, result2);

printf("F23\_ref(%d) - %d\n", num1, result2); // Первый вызов

F23\_ref(num2, result2);

printf("F23\_ref(%d) - %d\n", num2, result2); // Второй вызов

F23\_pointer(&num1, &result3);

printf("F23\_pointer(%d) - %d\n", num1, result3); // Первый вызов

F23\_pointer(&num2, &result3);

printf("F23\_pointer(%d) - %d\n", num2, result3); // Второй вызов

// ФУНКЦИЯ F26: Удаление простых чисел

printf("---------------------------------------------------------\n");

printf("F26\_val(A1, %d):\n", n1);

F26\_val(A1, n1); // Первый вызов

print\_mass(A1);

printf("F26\_val(A2, %d):\n", n2);

F26\_val(A2, n2); // Второй вызов

print\_mass(A2);

printf("F26\_ref(A1, %d):\n", n1);

F26\_ref(A1, n1); // Первый вызов

print\_mass(A1);

printf("F26\_ref(A2, %d):\n", n2);

F26\_ref(A2, n2); // Второй вызов

print\_mass(A2);

vector<int> A3 = {10, 7, 15, 3, 9};

int n3 = 5;

printf("F26\_pointer(A3, %d):\n", n3);

F26\_pointer(&A3, &n3); // Первый вызов

print\_mass(A3);

vector<int> A4 = {20, 13, 25, 5, 17};

int n4 = 5;

printf("F26\_pointer(A4, %d):\n", n4);

F26\_pointer(&A4, &n4); // Второй вызов

print\_mass(A4);

// ФУНКЦИЯ F28: Нахождение НОД

printf("---------------------------------------------------------\n");

printf("F28\_val(A1, %d) - %d\n", n1, F28\_val(A1, n1)); // Первый вызов

printf("F28\_val(A2, %d) - %d\n", n2, F28\_val(A2, n2)); // Второй вызов

F28\_ref(A1, n1, result2);

printf("F28\_ref(A1, %d) - %d\n", n1, result2); // Первый вызов

F28\_ref(A2, n2, result2);

printf("F28\_ref(A2, %d) - %d\n", n2, result2); // Второй вызов

F28\_pointer(&A3, n3, &result3);

printf("F28\_pointer(A3, %d) - %d\n", n3, result3); // Первый вызов

F28\_pointer(&A4, n4, &result3);

printf("F28\_pointer(A4, %d) - %d\n", n4, result3); // Второй вызов

// ФУНКЦИЯ F34: Нахождение наибольшей степени двойки

printf("---------------------------------------------------------\n");

printf("F34\_val(%d) - %d\n", num1, F34\_val(num1)); // Первый вызов

printf("F34\_val(%d) - %d\n", num2, F34\_val(num2)); // Второй вызов

F34\_ref(num1, result2);

printf("F34\_ref(%d) - %d\n", num1, result2); // Первый вызов

F34\_ref(num2, result2);

printf("F34\_ref(%d) - %d\n", num2, result2); // Второй вызов

F34\_pointer(&num1, &result3);

printf("F34\_pointer(%d) - %d\n", num1, result3); // Первый вызов

F34\_pointer(&num2, &result3);

printf("F34\_pointer(%d) - %d\n", num2, result3); // Второй вызов

// ФУНКЦИЯ F35: Переворачивание массива

printf("---------------------------------------------------------\n");

printf("F35\_val(A1, %d):\n", n1);

F35\_val(A1, n1); // Первый вызов

print\_mass(A1);

printf("F35\_val(A2, %d):\n", n2);

F35\_val(A2, n2); // Второй вызов

print\_mass(A2);

printf("F35\_ref(A1, %d):\n", n1);

F35\_ref(A1, n1); // Первый вызов

print\_mass(A1);

printf("F35\_ref(A2, %d):\n", n2);

F35\_ref(A2, n2); // Второй вызов

print\_mass(A2);

vector<int> A5 = {10, 7, 15, 3, 9};

int n5 = 5;

printf("F35\_pointer(A5, %d):\n", n5);

F35\_pointer(&A5, &n5); // Первый вызов

print\_mass(A5);

vector<int> A6 = {20, 13, 25, 5, 17};

int n6 = 5;

printf("F35\_pointer(A6, %d):\n", n6);

F35\_pointer(&A6, &n6); // Второй вызов

print\_mass(A6);

return 0;

}

# Пример работы программы

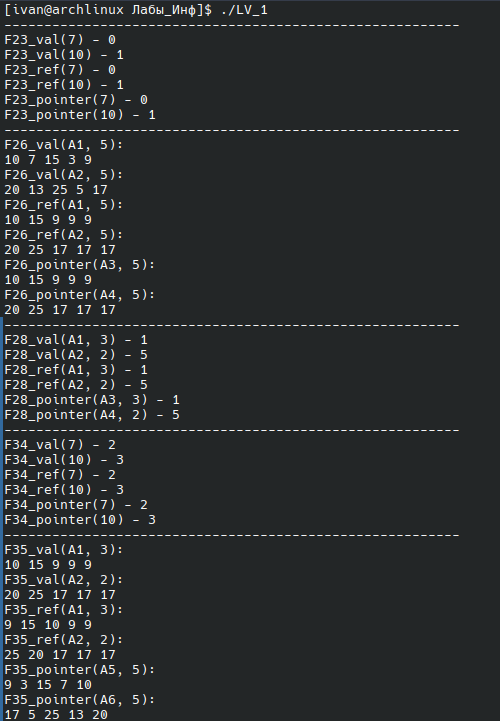


Рисунок 1 – Пример работы программы

# Ошибки и неточности

Есть некоторые комментарии по коду касающиеся довольно важных аспектов непосредственно самого языка. Во первых при передаче по значению создаются копии объектов, которые неплохо было бы удалять для более грамотным распоряжением памятью, так же не было замечено никаких проверок при передаче по указателю, что может являться проблемой. Проверка на NULL значения может улучшить стабильность программы. Проверки необходимы и алгоритмам, в которых есть смена знаков, если не проверять знаки в начале или конце массива, то это может негативно сказаться на работе всей программы. Последним, но не по важности замечанием будет отсутствие проверок на входные значения. При получении некорректных входных знаений программа может отработать с труднопрогнозируемым результатом, но так же может не отработать вовсе.

# Выводы

*В ходе выполнения лабораторной работы были рассмотрены и реализованы три способа передачи параметров в функции:*

1. **Передача по значению** – позволяет работать с копией данных, не изменяя оригинальные переменные, что обеспечивает безопасность исходных данных, но требует дополнительных затрат памяти.
2. ***Передача по ссылке*** *– предоставляет доступ к исходным данным и позволяет изменять их непосредственно, что экономит память и ускоряет работу программы.*
3. ***Передача по указателю*** *– аналогична передаче по ссылке, но требует явного разыменования, предоставляя гибкость при работе с динамическими структурами данных.*

*Реализованы и протестированы функции для решения различных задач, включая разложение числа на цифры, генерацию простых чисел, поиск элемента с равным числом больших и меньших значений, нахождение минимального положительного элемента и подсчёт переходов от отрицательных к положительным числам.*

# Список литературы

1. Подбельский В.В., Фомин С.С. Программирование на языке Си: Учеб.пособие. – 2-е доп. Изд. – М.: Финансы и статистика, 2004. – 600 с.

2. Романов Е. Л. Си/Си++. От дилетанта до профессионала. Электронное учебное пособие по дисциплинам "Информатика", "Программирование", "Технология программирования" для студентов 1–2 курсов направления 230100 : учеб. пособие / Е. Л. Романов. – Новосибирский государственный технический университет, № гос регистрации 0321000528, 2010. - 581 с.

3. Си/Си++ от дилетанта до профессионала [Электронный ресурс]. URL: http://ermak.cs.nstu.ru/cprog/HTML/index.htm (дата обращения 01.10.2022).

# Приложение А. Исходный код программы

#include <iostream>

#include <vector>

using namespace std;

// Функция для вывода массива

void print\_mass(const vector<int>& mass) {

// Проходимся по массиву и выводим каждый элемент через пробел

for (vector<int>::size\_type i = 0; i < mass.size(); i++) {

printf("%d ", mass[i]);

}

printf("\n"); // Переход на новую строку после вывода всех элементов

}

// Проверка, является ли число простым

// Передача параметров по значению

int F23\_val(int a) {

int s, n;

// Пробегаем от 2 до числа 'a' и проверяем, делится ли оно нацело

for (s = 0, n = 2; n < a; n++) {

if (a % n == 0) { // Если есть делители кроме самого числа и единицы

s = 1; // Флаг простого числа устанавливается в 1 (не простое)

break; // Выходим из цикла, поскольку нашли делитель

}

}

return s; // Возвращаем результат проверки простоты числа

}

// Проверка, является ли число простым

// Передача параметров по ссылке

void F23\_ref(const int &a, int &result) {

int s, n;

// Аналогично функции F23\_val, но результат записывается по ссылке

for (s = 0, n = 2; n < a; n++) {

if (a % n == 0) {

s = 1;

break;

}

}

result = s; // Запись результата в переменную, переданную по ссылке

}

// Проверка, является ли число простым

// Передача параметров по указателю

void F23\_pointer(const int \*a, int \*result) {

int s, n;

// Здесь работаем с числом по указателю '\*a'

for (s = 0, n = 2; n < \*a; n++) {

if (\*a % n == 0) {

s = 1;

break;

}

}

\*result = s; // Результат сохраняется по указателю

}

// Удаление простых чисел из массива

// Передача параметров по значению

void F26\_val(vector<int> A, int n) {

int i, j, s, m;

// Проходим по каждому элементу массива и проверяем его простоту

for (i = 0; i < n; i++) {

for (s = 0, m = 2; m < A[i]; m++) {

if (A[i] % m == 0) {

s = 1;

break;

}

}

if (s == 0) { // Если число простое, удаляем его из массива

for (j = i; j < n - 1; j++) {

A[j] = A[j + 1]; // Сдвигаем элементы вправо

}

n--; // Уменьшаем размер массива

i--; // Начинаем проверку заново с предыдущего элемента

}

}

}

// Удаление простых чисел из массива

// Передача параметров по ссылке

void F26\_ref(vector<int> &A, int &n) {

int i, j, s, m;

// Аналогично F26\_val, но изменения происходят непосредственно в переданном массиве

for (i = 0; i < n; i++) {

for (s = 0, m = 2; m < A[i]; m++) {

if (A[i] % m == 0) {

s = 1;

break;

}

}

if (s == 0) {

for (j = i; j < n - 1; j++) {

A[j] = A[j + 1];

}

n--;

i--;

}

}

}

// Удаление простых чисел из массива

// Передача параметров по указателю

void F26\_pointer(vector<int> \*A, int \*n) {

int i, j, s, m;

// Работаем с массивом через указатель

for (i = 0; i < \*n; i++) {

for (s = 0, m = 2; m < (\*A)[i]; m++) {

if ((\*A)[i] % m == 0) {

s = 1;

break;

}

}

if (s == 0) {

for (j = i; j < \*n - 1; j++) {

(\*A)[j] = (\*A)[j + 1];

}

(\*n)--;

i--;

}

}

}

// Нахождение наибольшего общего делителя элементов массива

// Передача параметров по значению

int F28\_val(const vector<int> &A, int n) {

int v, i, s;

v = A[0] + 1; // Начальное значение НОД больше первого элемента массива

do {

v--; // Уменьшаем возможное значение НОД

for (i = 0, s = 0; i < n; i++) {

if (A[i] % v != 0) { // Если хотя бы одно число не делится на v

s = 1; // Обозначаем, что v не подходит

break; // Прерываем цикл

}

}

} while (s == 1); // Продолжаем искать меньшее значение, пока не найдем подходящий НОД

return v; // Возвращаем найденный НОД

}

// Нахождение наибольшего общего делителя элементов массива

// Передача параметров по ссылке

void F28\_ref(const vector<int> &A, int n, int &result) {

int v, i, s;

v = A[0] + 1;

do {

v--;

for (i = 0, s = 0; i < n; i++) {

if (A[i] % v != 0) {

s = 1;

break;

}

}

} while (s == 1);

result = v; // Сохраняем результат в переменной, переданной по ссылке

}

// Нахождение наибольшего общего делителя элементов массива

// Передача параметров по указателю

void F28\_pointer(const vector<int> \*A, int n, int \*result) {

int v, i, s;

v = (\*A)[0] + 1;

do {

v--;

for (i = 0, s = 0; i < n; i++) {

if ((\*A)[i] % v != 0) {

s = 1;

break;

}

}

} while (s == 1);

\*result = v; // Сохраняем результат по указателю

}

// Нахождение наибольшей степени двойки, меньшей или равной n

// Передача параметров по значению

int F34\_val(int n) {

int k, m;

// Подсчитываем степень двойки, которая меньше или равна числу 'n'

for (k = 0, m = 1; m <= n; k++, m = m \* 2);

return k - 1; // Возвращаем степень двойки

}

// Нахождение наибольшей степени двойки, меньшей или равной n

// Передача параметров по ссылке

void F34\_ref(const int &n, int &result) {

int k, m;

for (k = 0, m = 1; m <= n; k++, m = m \* 2);

result = k - 1; // Сохраняем результат в переменной, переданной по ссылке

}

// Нахождение наибольшей степени двойки, меньшей или равной n

// Передача параметров по указателю

void F34\_pointer(const int \*n, int \*result) {

int k, m;

for (k = 0, m = 1; m <= \*n; k++, m = m \* 2);

\*result = k - 1; // Сохраняем результат по указателю

}

// Переворачивание массива

// Передача параметров по значению

void F35\_val(vector<int> c, int n) {

int i, j, k;

// Меняем местами первый и последний элементы, второй и предпоследний и т.д.

for (i = 0, j = n - 1; i < j; i++, j--) {

k = c[i];

c[i] = c[j];

c[j] = k;

}

}

// Переворачивание массива

// Передача параметров по ссылке

void F35\_ref(vector<int> &c, int &n) {

int i, j, k;

// То же самое, но изменения производятся непосредственно в переданном массиве

for (i = 0, j = n - 1; i < j; i++, j--) {

k = c[i];

c[i] = c[j];

c[j] = k;

}

}

// Переворачивание массива

// Передача параметров по указателю

void F35\_pointer(vector<int> \*c, int \*n) {

int i, j, k;

// Работа с массивом через указатели

for (i = 0, j = \*n - 1; i < j; i++, j--) {

k = (\*c)[i];

(\*c)[i] = (\*c)[j];

(\*c)[j] = k;

}

}

# int main() {

# vector<int> A1 = {10, 7, 15, 3, 9}; // Первый набор данных

# vector<int> A2 = {20, 13, 25, 5, 17}; // Второй набор данных

# int n1 = 5, n2 = 5; // Размеры массивов

# int result2, result3; // Переменные для хранения результатов

# // Числа для вызова функций

# int num1 = 7, num2 = 10;

# // ФУНКЦИЯ F23: Проверка на простоту

# printf("---------------------------------------------------------\n");

# printf("F23\_val(%d) - %d\n", num1, F23\_val(num1)); // Первый вызов

# printf("F23\_val(%d) - %d\n", num2, F23\_val(num2)); // Второй вызов

# F23\_ref(num1, result2);

# printf("F23\_ref(%d) - %d\n", num1, result2); // Первый вызов

# F23\_ref(num2, result2);

# printf("F23\_ref(%d) - %d\n", num2, result2); // Второй вызов

# F23\_pointer(&num1, &result3);

# printf("F23\_pointer(%d) - %d\n", num1, result3); // Первый вызов

# F23\_pointer(&num2, &result3);

# printf("F23\_pointer(%d) - %d\n", num2, result3); // Второй вызов

# // ФУНКЦИЯ F26: Удаление простых чисел

# printf("---------------------------------------------------------\n");

# printf("F26\_val(A1, %d):\n", n1);

# F26\_val(A1, n1); // Первый вызов

# print\_mass(A1);

# printf("F26\_val(A2, %d):\n", n2);

# F26\_val(A2, n2); // Второй вызов

# print\_mass(A2);

# printf("F26\_ref(A1, %d):\n", n1);

# F26\_ref(A1, n1); // Первый вызов

# print\_mass(A1);

# printf("F26\_ref(A2, %d):\n", n2);

# F26\_ref(A2, n2); // Второй вызов

# print\_mass(A2);

# vector<int> A3 = {10, 7, 15, 3, 9};

# int n3 = 5;

# printf("F26\_pointer(A3, %d):\n", n3);

# F26\_pointer(&A3, &n3); // Первый вызов

# print\_mass(A3);

# vector<int> A4 = {20, 13, 25, 5, 17};

# int n4 = 5;

# printf("F26\_pointer(A4, %d):\n", n4);

# F26\_pointer(&A4, &n4); // Второй вызов

# print\_mass(A4);

# // ФУНКЦИЯ F28: Нахождение НОД

# printf("---------------------------------------------------------\n");

# printf("F28\_val(A1, %d) - %d\n", n1, F28\_val(A1, n1)); // Первый вызов

# printf("F28\_val(A2, %d) - %d\n", n2, F28\_val(A2, n2)); // Второй вызов

# F28\_ref(A1, n1, result2);

# printf("F28\_ref(A1, %d) - %d\n", n1, result2); // Первый вызов

# F28\_ref(A2, n2, result2);

# printf("F28\_ref(A2, %d) - %d\n", n2, result2); // Второй вызов

# F28\_pointer(&A3, n3, &result3);

# printf("F28\_pointer(A3, %d) - %d\n", n3, result3); // Первый вызов

# F28\_pointer(&A4, n4, &result3);

# printf("F28\_pointer(A4, %d) - %d\n", n4, result3); // Второй вызов

# // ФУНКЦИЯ F34: Нахождение наибольшей степени двойки

# printf("---------------------------------------------------------\n");

# printf("F34\_val(%d) - %d\n", num1, F34\_val(num1)); // Первый вызов

# printf("F34\_val(%d) - %d\n", num2, F34\_val(num2)); // Второй вызов

# F34\_ref(num1, result2);

# printf("F34\_ref(%d) - %d\n", num1, result2); // Первый вызов

# F34\_ref(num2, result2);

# printf("F34\_ref(%d) - %d\n", num2, result2); // Второй вызов

# F34\_pointer(&num1, &result3);

# printf("F34\_pointer(%d) - %d\n", num1, result3); // Первый вызов

# F34\_pointer(&num2, &result3);

# printf("F34\_pointer(%d) - %d\n", num2, result3); // Второй вызов

# // ФУНКЦИЯ F35: Переворачивание массива

# printf("---------------------------------------------------------\n");

# printf("F35\_val(A1, %d):\n", n1);

# F35\_val(A1, n1); // Первый вызов

# print\_mass(A1);

# printf("F35\_val(A2, %d):\n", n2);

# F35\_val(A2, n2); // Второй вызов

# print\_mass(A2);

# printf("F35\_ref(A1, %d):\n", n1);

# F35\_ref(A1, n1); // Первый вызов

# print\_mass(A1);

# printf("F35\_ref(A2, %d):\n", n2);

# F35\_ref(A2, n2); // Второй вызов

# print\_mass(A2);

# vector<int> A5 = {10, 7, 15, 3, 9};

# int n5 = 5;

# printf("F35\_pointer(A5, %d):\n", n5);

# F35\_pointer(&A5, &n5); // Первый вызов

# print\_mass(A5);

# vector<int> A6 = {20, 13, 25, 5, 17};

# int n6 = 5;

# printf("F35\_pointer(A6, %d):\n", n6);

# F35\_pointer(&A6, &n6); // Второй вызов

# print\_mass(A6);

# return 0;

# }