Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Белгородский государственный технологический университет им. В.Г. Шухова»

(БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №6

По дисциплине: «Основы программирования»

Тема: «Введение в функции»

Выполнил: студент группы ВТ-231

Борченко Александр Сергеевич

Проверили:

Черников Сергей Викторович

Новожен Никита Викторович

Белгород 2023

Цель работы: получение навыков написания функций при решении простых задач. Закрепление навыков разработки алгоритмов разветвляющейся и циклической структуры. Получение навыков формулирования спецификаций к разрабатываемым функциям.

Содержание работы:

Задача №1: Напишите функцию abs для вычисления модуля вещественного числа x

Код задачи:

```c
#include <stdio.h>

float abs(float x) {
    if (x < 0) {
        return -x;
    }
    else {
        return x;
    }
}

int main() {
    float n;
    scanf("%f", &n);

    printf("%f", abs(n));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные	Примечание
-7.5	7.5	Вывод модуля вещественного числа

Задача №2: Напишите функцию $max2$, которая возвращает максимальное значение из двух целочисленных переменных типа int

Код задачи:

```c
#include <stdio.h>

int max2(int a, int b) {
    return a > b ? a : b;
}

int main()
{
    int n, m;
    scanf("%d %d", &n, &m);

    printf("%d", max2(n,m));

}
```

Тестовые данные:

Входные данные	Выходные данные	Примечание
77 50	50	Вывод максимального значения
-569 0	0	Проверка с отрицательными значениями

Задача №3: Напишите функцию $max3$, которая возвращает максимальное значение из трёх целочисленных переменных типа int**.**

Код задачи:

```c
#include <stdio.h>

int max2(int a, int b) {
    return (a > b) ? a : b;
}

int max3(int a, int b, int c) {
    return max2(max2(a, b), c);
}

int main() {
    int x, y, z;
    scanf("%d %d %d", &x, &y, &z);

    printf("%d", max3(x, y, z));

}
```

Тестовые данные:

Входные данные	Выходные данные	Примечание
-7 10 2	10	Работа с отрицательными значениями
7 777 900	900	Правильный поиск максимума из 3х значений

Задача №4: Напишите функцию $getDistance$, которая вычисляет расстояние между двумя точками, заданными целочисленными координатами $(x1, y1)$, $(x2, y2)$.

Код задачи:

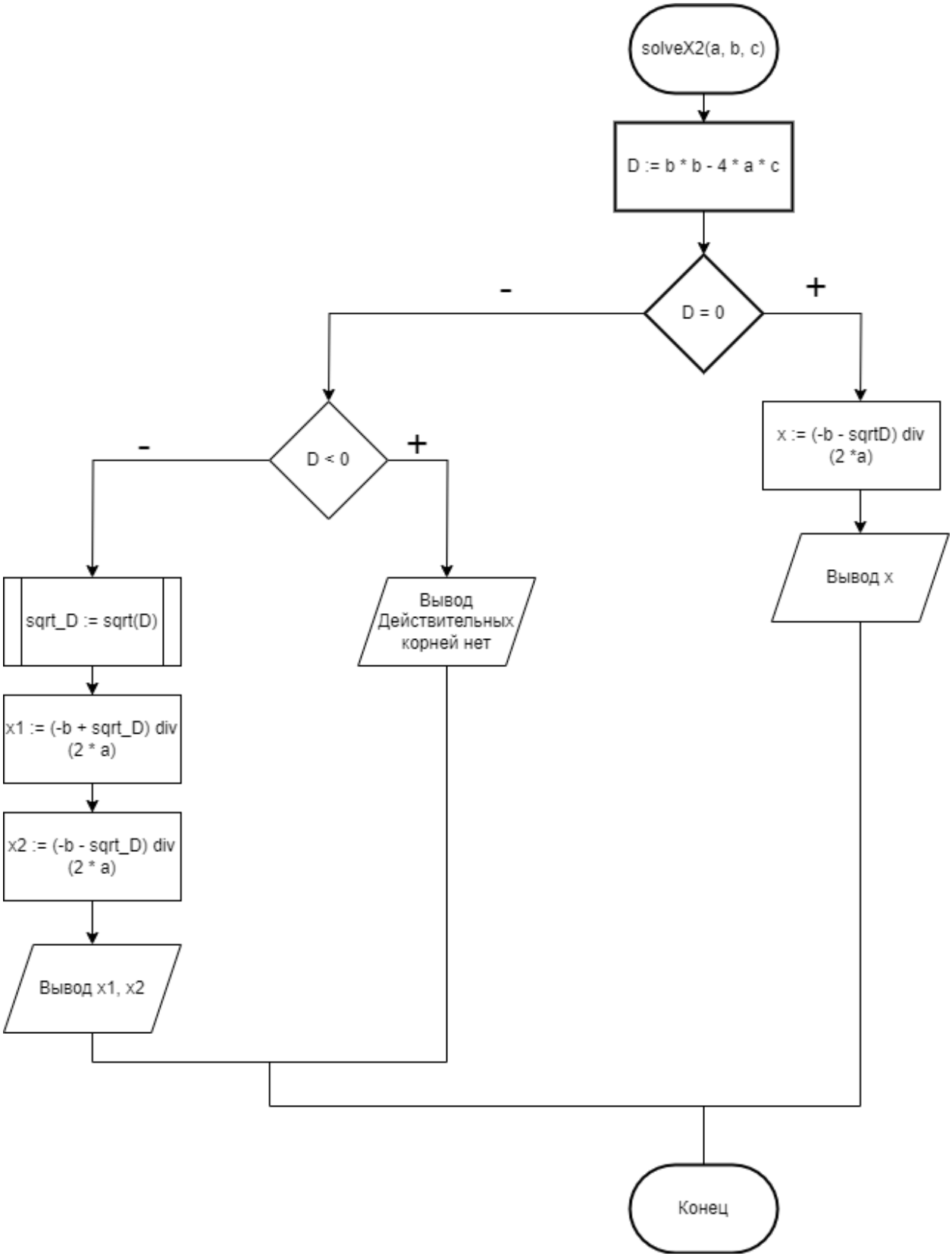
```c
#include <stdio.h>
#include <math.h>

double getDistance(int x1, int y1, int x2, int y2) {
    int dx = x2 - x1;
    int dy = y2 - y1;
    float distance;
    distance = sqrt(dx*dx + dy*dy);

    return distance;
}

int main() {
    int x1, y1, x2, y2;
    float distance;
    scanf("%d %d", &x1, &y1);

    scanf("%d %d", &x2, &y2);

    distance = getDistance(x1, y1, x2, y2);

    printf("%f", distance);

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные	Примечание
2 5 -3 7	$5{,}38516 \approx \sqrt{29}$	Поиск нецелочисленного расстояния
-6 -7 -9 -3	5	Поиск целочисленного расстояния

Задача №5: Напишите функцию $solveX2$, которая выводит корни квадратного уравнения: $ax2 + bx + c = 0$ $(a \neq 0)$ Найденные корни должны быть выведены в теле функции. Если действительных корней нет, выведите соответствующее сообщение**.**

Блок-схема:



Код задачи:

```c
#include <stdio.h>
#include <math.h>
#include <windows.h>

void solveX2(float a, float b, float c) {
    SetConsoleOutputCP(CP_UTF8);

    float D = b * b - 4 * a * c;

    if (D == 0) {
        float sqrtD = sqrt(D);
        float x = (-b - sqrtD) / (2 * a);

        printf("%f", x);

    } else if (D < 0) {

        printf("Действительных корней нет");

    } else {
        float root_D = sqrt(D);
        float x1 = (-b + root_D) / (2 * a);
        float x2 = (-b - root_D) / (2 * a);

        printf("%f %f", x1, x2);
    }
}
int main() {
    float x, y, z;
    scanf("%f %f %f", &x, &y, &z);

     solveX2(x,y,z);

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные	Примечание
1 -5 4	4 1	Правильный поиск корней, при D > 0
16 -40 25	1,25	Поиск корня при D = 0
2 1 5	Действительных корней нет	Проверка на D < 0

Задача №6: Напишите функцию $isDigit$, которая возвращает значение ’истина’, если символ x является цифрой, ’ложь’ - в противном случае**.**

Код задачи:

```c
#include <stdio.h>
#include <stdbool.h>

bool isDigit(char x) {
    return x >= '0' && x <= '9';
}

int main()
{
    char x = getchar();

    printf("%d", isDigit(x));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные	Примечание
Z	0	Проверка заглавной буквы
a	0	Проверка строчной буквы
9	1	Проверка цифры

Задача №7: Напишите функцию *swap*, которая принимает две переменные типа $float$ и обменивает их значения**.**

Код задачи:

```c
#include <stdio.h>

float swap(float *a, float *b) {
    float t = *a;
    *a = *b;
    *b = t;
}

int main()
{
    float a, b;
    scanf("%f %f", &a, &b);

    swap(&a, &b);

    printf("%f %f", a, b);

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные	Примечание
2.5 7.8	7.8 2.5	Правильный обмен значениями вещественных чисел

Задача №8: Напишите функцию $sort2$, которая упорядочивает значения a и b типа $float$. Т.е. если $a > b$ то после выполнения функции значение переменной a должно быть не больше значения переменной b**.**

Код задачи:

```c
#include <stdio.h>

float swap(float *a, float *b) {
    float t = *a;
    *a = *b;
    *b = t;
}

float sort2(float *a, float *b) {
    if (*a > *b)
        swap(a, b);
}

int main()
{
    float a, b;
    scanf("%f %f", &a, &b);

    sort2(&a, &b);

    printf("%f %f", a, b);

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные	Примечание
9.5 -4.4	-4.4 9.5	Проверка с отрицательными числами
7.2 9.9	7.2 9.9	Правильное упорядочивание с положительными числами

Задача №9: Напишите функцию $sort3$, которая упорядочивает значения переменных a, b, c типа $float$ таким образом, чтобы: $a \leq b \leq$ c

Код задачи:

```c
#include <stdio.h>

float swap(float *a, float *b) {
    float t = *a;
    *a = *b;
    *b = t;
}

float sort2(float *a, float *b) {
    if (*a > *b)
        swap(a, b);
}

float sort3(float *a, float *b, float *c) {
    sort2(a, b);
    sort2(b, c);
    sort2(a, b);
}

int main()
{
    float a, b, c;
    scanf("%f %f %f", &a, &b, &c);

    sort3(&a, &b, &c);

    printf("%f %f %f", a, b, c);

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные	Примечание
1.2 2.4 5.6	1.2 2.4 5.6	Проверка условия $a \leq b \leq$ c
12.5 5.9 -7.7	-7.7 5.9 12.5	Проверка с отрицательными числами
6 7 4	4 6 7	Работа алгоритма «Без багов»

Задача №10: Напишите функцию, которая возвращает значение ’истина’, если можно составить треугольник с целочисленными сторонами a, b, c (a, b, $c \in N$), ’ложь’ - в противном случае**.**

Код задачи:

```c
#include <stdio.h>
#include <stdbool.h>

int swap(int *a, int *b) {
    int t = *a;
    *a = *b;
    *b = t;
}

int sort2(int *a, int *b) {
    if (*a > *b)
        swap(a, b);
}

int sort3(float *a, float *b, float *c) {
    sort2(a, b);
    sort2(b, c);
    sort2(a, b);
}

bool isTrianglePossible (int a, int b, int c) {
    sort3(&a, &b, &c);
    return a + b - c > 0;
}

int main() {
    int a, b, c;
    scanf("%d %d %d", &a, &b, &c);

    printf("%d", isTrianglePossible(a, b, c));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные	Примечание
2 5 70	0	Треугольник с такими сторонами нельзя составить
3 4 5	1	Треугольник с такими сторонами можно составить

Задача №11: Напишите функцию $getTriangleTypeLength$, которая возвращает значение 0, если треугольник со сторонами a, b, c является остроугольным, 1 – если прямоугольным, 2 – тупоугольным, -1 – если треугольник с такими сторонами не существует

Код задачи:

```c
#include <stdio.h>
#include <stdbool.h>

void swap(int *a, int *b) {
    int t = *a;
    *a = *b;
    *b = t;
}

void sort2(int *a, int *b) {
    if (*a > *b)
        swap(a, b);
}

void sort3(int *a, int *b, int *c) {
    sort2(a, b);
    sort2(b, c);
    sort2(a, b);
}

bool isTrianglePossible(int a, int b, int c) {
    sort3(&a, &b, &c);
    return a + b > c;
}

int getTriangleTypeLenght(int a, int b, int c) {
    int type_triangle;
    if (isTrianglePossible(a, b, c)) {
        int square_sum_sides = a * a + b * b;
        int hypotenuse = c*c;

        if (hypotenuse < square_sum_sides) {
            type_triangle = 0;
        } else if (hypotenuse == square_sum_sides) {
            type_triangle = 1;
        } else {
            type_triangle = 2;
        }
    } else {
        type_triangle = -1;
    }

    return type_triangle;
}

int main() {
    int a, b, c;
    scanf("%d %d %d", &a, &b, &c);

    printf("%d", getTriangleTypeLenght(a, b, c));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные	Примечание
5 6 9	2	Тупоугольный треугольник
4 5 6	0	Остроугольный треугольник
8 6 10	1	Прямоугольный треугольник
82 6 10	-1	Треугольник с такими сторонами нельзя составить

Задача №12: Напишите функцию $isPrime$, которая возвращает значение ’истина’, если число является простым, иначе – ’ложь’. Приложите 3 вариации:

(a) Без оптимизаций

Код задачи:

```c
#include <stdio.h>
#include <stdbool.h>

int isPrime(int n) {
    if (n <= 1) {
        return false;
    }
    for (int i = 2; i < n; i++) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}

int main()
{
    int x;
    scanf("%d", &x);

    printf("%d", isPrime(x));

    return 0;

}
```

(b) С оптимизацией перебора до $\sqrt{N}$

Код задачи:

```c
#include <stdio.h>
#include <stdbool.h>
#include <math.h>

bool isPrime(long long n) {
    if (n <= 1) {
        return false;
    }
    for (long long i = 2; i <= sqrt(n); i++) {
        if (n % i == 0) {
            return false;
        }
    }
    return true;
}

int main()
{
    long long x;
    scanf("%lld", &x);

    printf("%lld", isPrime(x));

    return 0;

}
```

(c) С оптимизацией перебора до $\sqrt{N}$ и шагом 2

Код задачи:

```c
#include <stdio.h>
#include <stdbool.h>
#include <math.h>

bool isPrime(long long n) {
    int max_d = sqrt(n);
    int d = 3;
    int is_prime = !(n == 1 || n % 2 == 0 && n != 2);
    while (d <= max_d && is_prime) {
        is_prime = n % d;
        d += 2;
    }
    return is_prime;
}

int main()
{
    long long x;
    scanf("%lld", &x);

    printf("%lld", isPrime(x));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные	Примечание
2147483647	1	Проверка огромного простого числа
1324165494654688587	0	Проверка огромного непростого числа
2	1	Проверка единственного, четного простого числа

Задача №13: Натуральное число называется совершенным, если оно равно сумме всех своих делителей, за исключением самого себя. Число 6 – совершенное, т.к. 6=1+2+3. Число 8 – не совершенное, т.к. 8 ≠ 1+2+4. Дано натуральное число n. Получить все совершенные числа, меньшие n.

Код задачи:

```c
#include <stdio.h>
#include <windows.h>

// Функция для проверки, является ли число x совершенным
int isPerfect(long long x) {
    long long sum = 0;
    for (long long i = 1; i <= x/2; i++) {
        if (x % i == 0) {
            sum += i;
        }
    }
    return sum == x;
}

// Функция для вывода всех совершенных чисел меньше n
void printPerfectNumbers(long long n) {
    for (long long i = 1; i < n; i++) {
        if (isPerfect(i)) {
            printf("%lld ", i);
        }
    }
    printf("\n");
}

int main() {
    SetConsoleOutputCP(CP_UTF8);

    long long n;
    scanf("%lld", &n);

    printPerfectNumbers(n);

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные	Примечание
1000	6 28 496	Правильный поиск совершенных чисел

Задача №14: Найти количество чисел-палиндромов от 1 до n.

Код задачи:

```c
#include <stdio.h>
#include <stdbool.h>

// Функция для определения числа-палиндрома
int isPalindrome(int num) {
    int reversed = 0;
    int original = num;

    while (num > 0) {
        reversed = reversed * 10 + num % 10;
        num /= 10;
    }

    if (original == reversed) {
        return true;
    } else {
        return false;
    }
}

// Функция для нахождения количества чисел-палиндромов от 1 до n
int countPalindromes(int n) {
    int count = 0;

    for (int i = 1; i <= n; i++) {
        if (isPalindrome(i)) {
            count++;
        }
    }

    return count;
}

int main() {
    int n;
    scanf("%d", &n);

    int palindromeCount = countPalindromes(n);

    printf("%d", palindromeCount);

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные	Примечание
12	10 (1..9, 11)	Правильный поиск чисел-палиндромов
4000	138	Поиск большого количества чисел-палиндромов

Задача №15: В шестизначных автобусных билетах найти счастливые.

Код задачи:

```c
#include <stdio.h>

int sumFirstThreeDigits(int n) {
    int num1 = n / 1000;
    int sum1 = 0;
    for(size_t i = 0; i < 3; i++) {
        sum1 += num1 % 10;
        num1 /= 10;
    }
    return sum1;
}
int sumLastThreeDigits(int n) {
    int num2 = n % 1000;
    int sum2 = 0;
    for(size_t i = 0; i < 3; i++) {
        sum2 += num2 % 10;
        num2 /= 10;
    }
    return sum2;
}
int findLuckyTickets(int x) {
    return sumFirstThreeDigits(x) == sumLastThreeDigits(x);
}

int main()
{
    int n;
    scanf("%d", &n);

    printf("%d", findLuckyTickets(n));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные	Примечание
122320	1	Вывод 1 при счастливом билете (1+2+2 = 3+2)
369810	0	Вывод 0 при отсутствии счастливого билета (3+6+9 ≠ 8+1)

Вывод: в ходе проведения лабораторной работы я получил навыки написания функций для решения задач и закрепил навыки написания тестовых данных к программам.