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 №19.6
по дисциплине: «Введение в функции»

Выполнил/а: ст. группы ВТ-231
Кисиль Николай Владимирович

Проверили:
Черников Сергей Викторович
Новожен Никита Викторович

Белгород, 2023 г.

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

Пример тестовых данных:

№	Входные данных	Выходные данные
1	-1	1
2	23	23
3	-3.5	3.5

Спецификация функции fAbs:

1. Заголовок `float fAbs (float x).`
2. Назначение: возвращает модуль вещественного числа x

Код:

```
float fAbs (const float x) {
    return x < 0 ? -x : x;
}
```

Задача 2: Напишите функцию *max2*, которая возвращает максимальное значение из двух целочисленных переменных типа *int*.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 2	2
2	1 1	1
3	-12 -2	-2

Спецификация функции max2:

1. Заголовок `max2(long long a, long long b).`
2. Назначение: возвращает максимальное значение из двух целочисленных a и b.

Код:

```
long long max2(const long long a, const long long b) {
    return a > b ? a : b;
}
```

Задача 3: Напишите функцию *max3*, которая возвращает максимальное значение из трёх целочисленных переменных типа *int*.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 2 3	3
2	1 1 1	1
3	-12 -2 0	0

Спецификация функции max3:

1. Заголовок `long long max3(long long a, long long b, long long c).`
2. Назначение: возвращает максимальное значение из трех целочисленных a, b и c.

Код:

```
long long max3(const long long a, const long long b, const long long c) {
    return max2(max2(a, b), c);
}
```

Задача 4: Напишите функцию *getDistance*, которая вычисляет расстояние между двумя точками, заданными целочисленными координатами ($x1$, $y1$), ($x2$, $y2$).

Пример тестовых данных:

№	Входные данных	Выходные данные
1	0 -3 3 1	5
2	0 5 1 3	2.236068
3	-1 -2 -3 0	2.828427

Спецификация функции getDistance:

1. Заголовок `double getDistance(int x1, int y1, int x2, int y2).`
2. Назначение: возвращает расстояние между двумя точками, заданными целочисленными координатами ($x1$, $y1$), ($x2$, $y2$).

Код:

```
double getDistance(const int x1, const int y1, const int x2, const int y2) {
    const int delta_x = x1 - x2;
    const int delta_y = y1 - y2;

    return sqrt(delta_x * delta_x + delta_y * delta_y);
}
```

Задача 5: Напишите функцию *solveX2*, которая выводит корни квадратного уравнения.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	2 1 1	«Корней нет»
2	2 6 4	-2 -1
3	1 2 1	-1

Спецификация функции solveX2:

1. Заголовок `double solveX2(int a, int b, int c).`
2. Назначение: возвращает корни квадратного уравнения

Код:

```
void solveX2(const int a, const int b, const int c) {
    SetConsoleOutputCP(CP_UTF8);

    const double D = pow(b, 2) - 4*a*c;
    const double sqrtD = sqrt(D);

    if(D < 0) {
        printf("Корней нет");
    } else if(D == 0) {
        const double x = (-b - sqrtD) / (2 * a);

        printf("%f", x);
    } else {
        const double x1 = (-b - sqrtD) / (2 * a);
        const double x2 = (-b + sqrtD) / (2 * a);

        printf("%f %f", x1, x2);
    }
}
```

Задача 6: Написать функцию *isDigit*, которая возвращает значение ’истина’, если символ *x* является цифрой, ’ложь’ - в противном случае.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	a	0
2	1	1
3	12	1

Спецификация функции isDigit

1. Заголовок `bool isDigit(char x).`
2. Назначение: возвращает значение ’истина’, если символ *x* является цифрой, ’ложь’ - в противном случае.

Код:

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

Задача 7: Напишите функцию *swap*, которая принимает две переменные типа *float* и обменивает их значения.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 2	2 1
2	0 0	0 0
3	1.34 -2.134	-2.134 1.34

Спецификация функции swap:

1. Заголовок `void swap(float *a, float *b).`
2. Назначение: обменивает значение переменных

Код:

```
void swap(float * const a, float * const b) {
    const float temp = *a;
    *a = *b;
    *b = temp;
}
```

Задача 8: Напишите функцию $sort2$, которая упорядочивает значения a и b типа $float$.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 2	1 2
2	3 0	0 3
3	1.34 -2.134	-2.134 1.34

Спецификация функции sort2:

1. Заголовок `void sort2(float *a, float *b).`
2. Назначение: упорядочивает значения a и b

Код:

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

Задача 9: Напишите функцию $sort3$, которая упорядочивает значения переменных a, b, c типа $float$ таким образом, чтобы: $a \leq b \leq c$

Пример тестовых данных:

№	Входные данных	Выходные данные
1	1 3 2	1 2 3
2	0 3 0	0 0 3
3	1.34 -2.134 -1.5	-2.134 -1.5 1.34

Спецификация функции sort3:

1. Заголовок `void sort3(float *a, float *b, float *c)`
2. Назначение: упорядочивает значения переменных a, b, c

Код:

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

Задача 10: Написать функцию, которая возвращает значение ’истина’, если можно составить треугольник с целочисленными сторонами a, b, c ($a, b, c \in N$), ’ложь’ - в противном случае.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	3 4 5	1
2	1 2 1	0
3	1 1 1	1

Спецификация функции isTrianglePossible:

1. Заголовок `bool isTrianglePossible(int a, int b, int c).`
2. Назначение: возвращает значение ’истина’, если можно составить треугольник с целочисленными сторонами a, b, c, ’ложь’ - в противном случае.

Код:

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

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.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	3 3 3	0
2	3 4 5	1
3	3 3 5	2
4	1 2 1	-1

Спецификация функции getTriangleTypeLength:

1. Заголовок `int getTriangleTypeLength(int a, int b, int c).`
2. Назначение: 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:

```
int getTriangleTypeLength(int a, int b, int c) {
    sort3(&a, &b, &c);
    if(isTrianglePossible(a, b, c)) {
        if(c * c < a * a + b*b) {
            return 0;
        } else if(c*c == a*a + b*b) {
            return 1;
        } else if(c*c > a*a + b*b) {
            return 2;
        }
    }
    return -1;
}
```

Задача 12 (а): Напишите функцию *isP rime*, которая возвращает значение ’истина’, если число является простым, иначе – ’ложь’. Без оптимизации

Пример тестовых данных:

№	Входные данных	Выходные данные
1	3	1
2	4	0
3	2147483647	1

Спецификация функции isPrime:

1. Заголовок `int isPrime(int n).`
2. Назначение: возвращает значение ’истина’, если число является простым, иначе – ’ложь’.

Код:

```
int isPrime(const int n) {
    int d = 2;
    while (d < n && n % d != 0) {
        d++;
    }
    return d == n;
}
```

Задача 12 (б): Напишите функцию *isP rime*, которая возвращает значение ’истина’, если число является простым, иначе – ’ложь’. С оптимизацией перебора до √ N

Пример тестовых данных:

№	Входные данных	Выходные данные
1	3	1
2	4	0
3	2147483647	1

Спецификация функции isPrime:

1. Заголовок `int isPrime(int n).`
2. Назначение: возвращает значение ’истина’, если число является простым, иначе – ’ложь’.

Код:

```
int isPrime(const int n) {
    int max_d = sqrt(n);
    int d = 2;

    while (d <= max_d && n % d != 0) {
        d++;
    }
    return d == max_d + 1 && n != 1;
}
```

Задача 12 (в): Напишите функцию *isP rime*, которая возвращает значение ’истина’, если число является простым, иначе – ’ложь’. С оптимизацией перебора до $\sqrt{N}$ и шагом 2.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	3	1
2	4	0
3	2147483647	1

Спецификация функции isPrime:

1. Заголовок `int isPrime(int n).`
2. Назначение: возвращает значение ’истина’, если число является простым, иначе – ’ложь’.

Код:

```
int isPrime(const int 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;
}
```

Задача 13: Дано натуральное число n. Получить все совершенные числа, меньшие n.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	10	6
2	100	6 28
3	10000	6 28 496 8128

Спецификация функции isNumberPerfect:

1. Заголовок `long long isNumberPerfect(int x).`
2. Назначение: возвращает значение ’истина’, если число является совершенным, иначе – ’ложь’.

Спецификация функции printPerfectNumber:

1. Заголовок `void printPerfectNumber(int n).`
2. Назначение: выводит все совершенные числа, меньшие n.

Код:

```
long long isNumberPerfect(const int x) {
    int sum_dividers = 0;

    for(register size_t i = 1; i < x; i++) {
        if(x % i == 0) {
            sum_dividers += i;
        }
    }
    return sum_dividers == x;
}

void printPerfectNumber(const int n) {
    for(size_t i = 1; i < n; i++) {
        if(isNumberPerfect(i)) {
            printf("%d ", i);
        }
    }
}
```

Задача 14: Найти количество чисел-палиндромов от 1 до *n*.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	10	9
2	100	18
3	2	1

Спецификация функции isPalindrome:

3. Заголовок `int isPalindrome(int x).`
4. Назначение: возвращает значение ’истина’, если число является палиндромом, иначе – ’ложь’.

Спецификация функции getNumberPalindrome:

3. Заголовок `int getNumberPalindrome(int n).`
4. Назначение: счет количества палиндромов от 1 до *n*

Код:

```
int isPalindrome(const int x) {
    int temp_x = x;
    int reverse = 0;

    while (temp_x != 0) {
        reverse = reverse * 10 + temp_x % 10;
        temp_x /= 10;
    }

    return reverse == x;
}

int getNumberPalindrome(const int n) {
    int count_palindrome = 0;

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

    return count_palindrome;
}
```

Задача 15: В шестизначных автобусных билетах найти счастливые.

Пример тестовых данных:

№	Входные данных	Выходные данные
1	123321	1
2	104311	1
3	112321	0

Спецификация функции sumFirstThreeDigits:

5. Заголовок `int sumFirstThreeDigits(int x).`
6. Назначение: возвращает значение суммы первых 3 цифр числа

Спецификация функции sumLastThreeDigits:

5. Заголовок `int sumLastThreeDigits(int x).`
6. Назначение: возвращает значение суммы последних 3 цифр числа

Спецификация функции isLuckyTicket:

1. Заголовок `int isLuckyTicket(int x).`
2. Назначение: возвращает значение ’истина’, если число является «счастливым», иначе – ’ложь’.
3.

Код:

```
int sumFirstThreeDigits(const int x) {
    int temp_x = x / 1000;
    int sum = 0;
    for(size_t i = 0; i < 3; i++) {
        sum += temp_x % 10;
        temp_x /= 10;
    }

    return sum;
}

int sumLastThreeDigits(const int x) {
    int temp_x = x % 1000;
    int sum = 0;
    for(size_t i = 0; i < 3; i++) {
        sum += temp_x % 10;
        temp_x /= 10;
    }

    return sum;
}

int isLuckyTicket(const int x) {
    return sumFirstThreeDigits(x) == sumLastThreeDigits(x);
}
```

Вывод: почили навыки написания функций для решения задач. Получили навыки формулирования спецификаций к разрабатываемым функциям.