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 №4а

по дисциплине: Основы программирования
тема: «Введение в функции»

Выполнил: ст. группы ПВ-211
Чувилко Илья Романович

Проверили:
Притчин Иван Сергеевич
Черников Сергей Викторович

Белгород 2021 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.

№1. Напишите функцию `abs` для вычисления модуля вещественного числа `x`

Спецификация для функции `abs`**:**

1. **Заголовок:** `double abs(double a)`
2. **Назначение:** возвращает модуль вещественного числа `a`

Тестовые задания для задачи

Входные данные	Ожидаемый результат	Пояснение
`a = 1.23`	`1.23`	Положительное число
`a = -1.23`	`1.23`	Модуль отрицательного числа — Положительное число
`a = -0.0000001`	`0.0000001`	Модуль отрицательного числа — Положительное число
`a = 0`	`0`	Модуль нуля - 0

Код программы:

```
// Возвращает модуль вещественного числа a
double abs(const double a) {
  return a >= 0 ? a : -a;
}
```

№2. Напишите функцию `max2`, которая возвращает максимальное значение из двух целочисленных переменных типа `int`.

Спецификация для функции `max2`**:**

1. Заголовок: `int max2(int a, int b)`

2. Назначение: возвращает наибольшее значение из двух введенных целочисленных переменных `a` и `b`.

Тестовые задания для задачи

Входные данные	Ожидаемый результат	Пояснение
`a = 1` `b = 3`	3	Максимальная переменная: b = 3
`a = -1` `b = -8`	-1	Максимальная переменная: a = -1
`a = 0` `b = 0`	0	Значения равны, поэтому будет возвращено значение 0
`a = -5` `b = 5`	5	Максимальная переменная: b = 5

Код программы:

```
// Возвращает наибольшее число из двух введенных целочисленных
// переменных a и b
int max2(const int a, const int b) {
  if (a >= b)
    return a;
  else
    return b;
}
```

№3. Напишите функцию `max3`, которая возвращает максимальное значение из двух целочисленных переменных типа `int`. Используйте при решении функцию `max2`.

Спецификация для функции *`max3`***:**

1. Заголовок: `int max3(int a, int b, int c)`

2. Назначение: возвращает наибольшее значение из трех введенных целочисленных переменных `a, b, c`

Тестовые задания для задачи

Входные данные	Ожидаемый результат	Пояснение

`a = 1`
`b = 3`
`c = 0`
	`3`	Максимальная переменная: b = 3

`a = -1`
`b = - 8`
`c = 4`
	`4`	Максимальная переменная: c = 4

`a = 0`
`b = 0`
`c = 0`
	`0`	Значения равны, поэтому будет возвращено значение 0

`a = -1`
`b = -8`
`c = -4`
	`-1`	Максимальная переменная: a = -1

Код программы:

```
// Возвращает наибольшее значение из трех введенных целочисленных переменных
// a, b и c
int max3(const int a, const int b, const int c) {
  return max2(max2(a, b), c);
}
```

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

Спецификация для функции `getDistance`**:**

Заголовок: `double getDistance(int x1, int y1, int x2, int y2)`

Назначение: возвращает расстояние между двумя точками с координатами `(x1, y1)` и `(x2, y2)`

Тестовые задания для задачи

Входные данные	Ожидаемый результат	Пояснение
`x1 = 0, y1 = 0,` `x2 = 3, y2 = 4`	`5`	Египетский треугольник со сторонами 3 4 5
`x1 = 0, y1 = 0,` `x2 = 1, y2 = 1`	`1.414214`	По т. Пифагора, гипотенуза треугольника, с катетами 1 и 1, будет равна квадратному корню из двух

`x1 = 0, y1 = 0,` `x2 = -3, y2 = -4`	`5`	С отрицательными координатами результат будет тот же
`x1 = 12, y1 = -5,` `x2 = 8, y2 = 7`	`12.649111`	По т. Пифагора, гипотенуза треугольника, с катетами 4 и 12, будет равна квадратному корню из 160

Код программы:

```
// Возвращает расстояние между двумя точками с координатами (x1, y1) и (x2, y2)
double getDistance(const int x1, const int y1,
                   const int x2, const int y2) {
  int dx = abs(x1 - x2); // дельта x
  int dy = abs(y1 - y2); // дельта y
  return sqrt(dx * dx + dy * dy);
}
```

№5. Напишите функцию `solveX2`, которая выводит корни квадратного уравнения:

$$ax^2 + bx + c = 0$$

Найденные корни должны быть выведены в теле функции. Если действительных корней нет - вывести соответствующее сообщение.

Спецификация для функции `solveX2`**:**

Заголовок: `double solveX2(int a, int b, int c)`

Назначение: Выводит корни квадратного уравнения, где `a`, `b`, `c` – коэффициенты.

Тестовые задания для задачи

Входные данные	Ожидаемый результат	Пояснение

`a = 1`
`b = 2`
`c = 1`
	`-1 -1`	Квадрат суммы

`a = 1`
`b = -8`
`c = 12`
	`6 2`	Корни найденные через дискриминант

`a = 10`
`b = -30`
`c = 20`
	`2 1`	Корни найденные через дискриминант

`a = 0`
`b = 25`
`c = 5`
	`Not quadratic`	Не является квадратичной функцией, так как коэффициент перед квадратом = 0

a = 10 b = 30 c = 50	No solutions	Нет решений, так как дискриминант < 0

Код программы:

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

#define EPS 0.00001

// Выводит корни квадратного уравнения, где a, b, c – коэффициенты.
void solveX2(const int a, const int b, const int c) {
  if (a == 0)
    printf("not quadratic");
  double D = pow(b, 2) - 4 * a * c;
  if (D < -EPS)
    printf("No solutions");
  else if (dabs(D) <= EPS)
    printf("x = %d", -b / (2 * a));
  else {
    double x1 = (-b + sqrt(D)) / (2 * a);
    double x2 = (-b - sqrt(D)) / (2 * a);
    printf("x1 = %lf\nx2 = %lf", x1, x2);
  }
}
```

№6. Написать функцию `isDigit`, которая возвращает значение ’истина’, если символ `x` является цифрой, ’ложь’ - в противном случае.

Спецификация для функции `isDigit`**:**

Заголовок: `int isDigit(char a)`

Назначение: Возвращает значение 1 (’истина’), если символ `x` является цифрой, 0 (’ложь’) - в противном случае.

Тестовые задания для задачи

Входные данные	Ожидаемый результат	Пояснение
`a`	0	Это буква
`7`	1	Это цифра
`!`	0	Это знак
` `	0	Это пробел

Код программы:

```c
// Возвращает значение 1 (’истина’), если символ x - является цифрой,
// 0 (’ложь’) - в противном случае.
int isDigit(const char x) {
  return x >= '0' && x <= '9';
}
```

№7. Напишите функцию `swap`, которая принимает две переменные `a` и `b` типа `float` и обменивает их значения.

Спецификация для функции `swap`**:**

Заголовок: `void swap(float *a, float *b)`

Назначение: Обменивает значения двух переменных `a` и `b` типа float

Тестовые задания для задачи

Входные данные	Ожидаемый результат	Пояснение
a = 5.0 b = 7.8	a = 7.8 b = 5.0	Обменивает значения двух переменных
a = 2.33 b = 7.67	a = 7.67 b = 2.33	Обменивает значения двух переменных
a = -7.123 b = 7.123	a = 7.123 b = -7.123	Обменивает значения двух переменных
a = 0 b = 0	a = 0 b = 0	Обменивает значения двух переменных

Код программы:

```c
// Обменивает значения двух переменных a и b типа float
void swap(float *a, float *b) {
  float t = *a;
  *a = *b;
  *b = t;
}
```

№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 меньше значения переменной `b` (при решении используйте функцию `swap` из прошлой задачи).

Спецификация для функции `sort2`**:**

Заголовок: `void sort2(float *a, float *b)`

Назначение: Сортирует значения `a` и `b` по возрастанию.

Тестовые задания для задачи

Входные данные	Ожидаемый результат	Пояснение
a = 5.0 b = 7.8	a = 5.0 b = 7.8	Изначально значения в порядке возрастания
a = 7. 67 b = 2.33	a = 2.33 b = 7.67	a > b, поэтому меняем их местами

a = 7.123 b = -7.123	a = -7.123 b = 7.123	a > b, поэтому меняем их местами
a = 0 b = 0	a = 0 b = 0	Значения равны

Код программы:

```
// Сортирует значения a и b по возрастанию.
void sort2(float *a, float *b) {
  if (*a > *b)
    swap(a, b);
}
```

№9. Напишите функцию `sort3`, которая упорядочивает значения переменных `a, b, c` типа *float* таким образом, чтобы: `a` ≤ `b` ≤ `c` (при решении используйте функцию `sort2` из прошлой задачи).

Спецификация для функции `sort3`:

Заголовок: `void sort3(float *a, float *b, float *c)`

Назначение: Сортирует значения `a, b` и `c` по неубыванию.

Тестовые задания для задачи

Входные данные	Ожидаемый результат	Пояснение
a = 5.0 b = 7.8 c = 9.0	a = 5.0 b = 7.8 c = 9.0	Изначально значения в порядке возрастания
a = 7. 67 b = 2.33 c = 1.0	a = 1.0 b = 2.33 c = 7.67	Применяем сортировку
a = 7.123 b = -7.123 c = 0	a = -7.123 b = 0 c = 7.123	Применяем сортировку
a = 42 b = 42 c = 42	a = 42 b = 42 c = 42	Значения равны

Код программы:

```
// Сортирует значения a, b и c по возрастанию.
void sort3(float *a, float *b, float *c) {
  sort2(a, b);
  sort2(b, c);
  sort2(a, b);
}
```

№10. Написать функцию, которая возвращает значение ’истина’, если можно составить треугольник с целочисленными сторонами a, b, c, ’ложь’ - в противном случае (при решении используйте функцию из прошлой задачи).

Спецификация для функции `isTriangleExist`**:**

Заголовок: `int isTriangleExist(float *a, float *b, float *c)`

Назначение: Возвращает 1 (`Истина`), если треугольник со сторонами `a`, `b` и `c` существует. 0 (`Ложь`) - в обратном случае

Тестовые задания для задачи

Входные данные	Ожидаемый результат	Пояснение
a = 5.0 b = 7.8 c = 9.0	0	Треугольник с такими сторонами не существует
a = 7. 67 b = 2.33 c = 10	1	Треугольник с такими сторонами существует
a = 7.123 b = 7.123 c = 15	1	Треугольник с такими сторонами существует
a = 42 b = 42 c = 42	0	Треугольник с такими сторонами не существует

Код программы:

```
// Возвращает 1 (`Истина`), если треугольник со сторонами a, b и c существует.
// 0 (`Ложь`) - в обратном случае
int isTriangleExist(float a, float b, float c) {
  sort3(&a, &b, &c);
  return a + b > c;
}
```

№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.

Спецификация для функции `getTriangleTypeLength`:

Заголовок: `int getTriangleTypeLength(float *a, float *b, float *c)`

Назначение: Возвращает 0, если треугольник со сторонами `a, b, c` - является остроугольным, 1 — если прямоугольный, 2 — тупоугольным, -1 — если треугольник с такими сторонами не существует

Тестовые задания для задачи

Входные данные	Ожидаемый результат	Пояснение
a = 3 b = 3 c = 7	-1	Треугольник с такими сторонами не существует
a = 4 b = 4 c = 5	0	Остроугольный треугольник
a = 3 b = 4 c = 5	1	Прямоугольный треугольник
a = 3 b = 3 c = 5	2	Тупоугольный треугольник

Код программы:

```
#define EPS 0.00001

// Возвращает 0, если треугольник со сторонами a, b, c - является остроугольным,
// 1 – если прямоугольный, 2 – тупоугольным, -1 – если треугольник
// с такими сторонами не существует
int getTriangleTypeLength(float a, float b, float c) {
  sort3(&a, &b, &c);
  if (isTriangleExist(a, b, c) == 0)
    return -1;
  float pythagoreanTheorem = a * a + b * b - c * c;
  if (pythagoreanTheorem > EPS)
    return 0;
  else if (dabs(pythagoreanTheorem) <= EPS)
    return 1;
  else
    return 2;
}
```

№12. Напишите функцию `isPrime`, которая возвращает значение 1, если число является простым, иначе – 0. Приложите 3 вариации:
(a) Без оптимизаций
(b) С оптимизацией перебора до $\sqrt{N}$.
(c) С оптимизацией перебора до $\sqrt{N}$ и шагом 2.

Тестовые задания для задачи

Входные данные	Ожидаемый результат
1	0
2	1
4	0
5	1
13	1
128	0
501	0
502	0
503	1
1999	1
10000019	1
10000355	0

(a) **Спецификация** для функции `isPrimeA`**:**

Заголовок: `int isPrimeA(const int a)`

Назначение: Возвращает 1, если число `a` простое, 0 - в обратном случае. Без оптимизаций

Код программы:

```
// Возвращает 1, если число a простое, 0 в обратном случае. Без оптимизаций
int isPrimeA(const int a) {
  if (a == 1)
    return 0;
  for (int divider = 2; divider < a; divider++)
    if (a % divider == 0)
      return 0;
  return 1;
}
```

(b) **Спецификация** для функции `isPrimeB`**:**

Заголовок: `int isPrimeB(const int a)`

Назначение: Возвращает 1, если число `a` простое, 0 - в обратном случае. С оптимизацией перебора до √ `a`.

Код программы:

```
// Возвращает 1, если число a простое, 0 в обратном случае. С оптимизацией
// перебора до √ N.
int isPrimeB(const int a) {
  if (a == 1)
    return 0;
  int sqrtA = (int) sqrt(a);
  for (int divider = 2; divider < sqrtA; divider++)
    if (a % divider == 0)
      return 0;
  return 1;
}
```

(c) **Спецификация** для функции `isPrimeC`**:**

Заголовок: `int isPrimeC(const int a)`

Назначение: Возвращает 1, если число `a` простое, 0 - в обратном случае. С оптимизацией перебора до √ `a` с шагом 2.

Код программы:

```
// Возвращает 1, если число a простое, 0 - в обратном случае. С оптимизацией
// перебора до √ N с шагом 2.
int isPrimeC(const int a) {
  if ((a % 2 == 0 || a == 1) && a != 2)
    return 0;
  for (int divider = 3; divider < (int) sqrt(a); divider += 2)
    if (a % divider == 0)
      return 0;
  return 1;
}
```

№13. Напишите функцию `deleteOctNumber`, которая удаляет цифру `digit` в записи данного восьмеричного числа `a`:

Спецификация для функции `deleteOctNumber`**:**

Заголовок: `int deleteOctNumber(int a, const int digit)`

Назначение: Возвращает число, которое будет получено, после удаления цифр `digit` из восьмеричной записи числа `a`

Тестовые задания для задачи

Входные данные	Ожидаемый результат
$3179_{10} = 110'001'101'011_2 = 6153_8$	$653_8 = 110'101'011_2 = 427_{10}$
$9_{10} = 1'001_2 = 11_8$	0_{10}

$37_{10} = 100'101_2 = 45_8$	$45_8 = 100'101_2 = 37_{10}$

Код программы:

```
// Возвращает число, которое будет получено, после удаление цифр digit из
// восьмеричной записи числа
int deleteOctNumber(int a, const int digit) {
  int afterProcessA = 0, octIndex = 0;
  while (a > 0) {
    int iDigit = a & 7;
    if (iDigit != digit)
      afterProcessA += iDigit << (3 * octIndex++);
    a >>= 3;
  }
  return afterProcessA;
}
```

№14. Напишите функцию `swapPairBites`, которая меняет местами соседние цифры пар в двоичной записи данного натурального числа. Обмен начинается с младших разрядов. Непарная старшая цифра остается без изменения

Спецификация для функции `swapPairBites`**:**

Заголовок: `int swapPairBites(int a)`

Назначение: Возвращает число, которое будет получено, после того, как соседние цифры в двоичной записи числа `a` , начиная с младших разрядов, поменяются местами.

Тестовые задания для задачи

Входные данные	Ожидаемый результат
$77_{10} = 1`00`11`01_2$	$1`00`11`10_2 = 78_{10}$
$165_{10} = 10`10`01`01_2$	$1`01`10`10_2 = 90_{10}$

Код программы:

```
// Возвращает число, которое будет получено, после того, как соседние цифры
// в двоичной записи числа a, начиная с младших разрядов, поменяются местами
int swapPairBites(int a) {
  int afterProcessA = 0, quaIndex = 0;
  while (a > 0) {
    if (a == 1)
      afterProcessA += 1 << (2 * quaIndex++);
    else {
      int iDigit = a & 3;
      if (iDigit == 1 || iDigit == 2)
        iDigit = 3 - iDigit;

      afterProcessA += iDigit << (2 * quaIndex++);
    }
    a >>= 2;
  }
  return afterProcessA;
}
```

№15. Напишите функцию `invertHex`, которая преобразует число `x`, переставляя в обратном порядке цифры в шестнадцатеричном представлении данного натурального числа.

Спецификация для функции `lengthN`**:**

Заголовок: `int lengthN(int a, const int n)`

Назначение: Возвращает длину числа `a`, записанного в (2^`n`)-ой системе счисления.

Спецификация для функции `invertHex`**:**

Заголовок: `int invertHex(int a)`

Назначение: Возвращает число, которое будет получено, после того, как цифры в шестнадцатеричном представлении числа `a` будут записаны в обратном порядке.

Тестовые задания для задачи

Входные данные	Ожидаемый результат
$77_{10} = 100'1101_2 = 4D_{16}$	$D4_{16} = 1101'0100_2 = 212_{10}$
$2732_{10} = 1010'1010'1100_2 = AAC_{16}$	$CAA_{16} = 1100'1010'1010_2 = 3242_{10}$

Код программы:

```
// Возвращает длину числа a, записанного в (2^n)-ой системе счисления
int lengthN(int a, const int n) {
  int lengthHex = 0;
  while (a > 0) {
    lengthHex++;
    a >>= n;
  }
  return lengthHex;
}

// Возвращает число, которое будет получено, после того, как цифры в
// шестнадцатеричном представлении числа a будут записаны в обратном порядке
int invertHex(int a) {
  int afterProcessA = 0;
  for (int i = lengthN(a, 4) - 1; i >= 0; i--) {
    int iDigit = a & 15;
    afterProcessA += iDigit << (4 * i);
    a >>= 4;
  }
  return afterProcessA;
}
```

№16. Напишите функцию `isBinPoly`, которая возвращает значение 1, если число `x` является палиндромом в двоичном представлении, иначе - 0.

Спецификация для функции `isBinPoly`**:**

Заголовок: `int isBinPoly(int x)`

Назначение: Возвращает значение 1, если число `x` является палиндромом в двоичном представлении, иначе - 0.

Тестовые задания для задачи

Входные данные	Ожидаемый результат
$27_{10} = 11011_2$	1
$454_{10} = 111000110_2$	0

Код программы:

```
// Возвращает значение 1, если число x является палиндромом в
// двоичном представлении, иначе - 0.
int isBinPoly(const int x) {
  int isPoly = 0, tmp = x, afterProcessX = 0;
  while (tmp > 0) {
    if (afterProcessX != 0)
      afterProcessX <<= 1;
    afterProcessX += tmp & 1;
    tmp >>= 1;
  }
  if (x == afterProcessX)
    isPoly = 1;
  return isPoly;
}
```

№17. Даны два двухбайтовых целых `sh1` и `sh2`. Получить целое число, последовательность нечетных битов которого представляет собой значение `sh1`, а последовательность четных – значение `sh2`.

Спецификация для функции `twoBitHybrid`**:**

Заголовок: `long long twoBitHybrid(long long sh1, long long sh2)`

Назначение: Возвращает целое число, последовательность нечетных битов которого представляет собой значение `sh1`, а последовательность четных – значение `sh2` .

Тестовые задания для задачи

Входные данные	Ожидаемый результат
sh1 = $0000000000001100_2 = 12_{10}$ sh2 = $0000000000010010_2 = 18_{10}$	$00000000000000000000000110100100_2$ $= 420_{10}$
sh1 = $0111111100000000_2 = 32512_{10}$ sh2 = $0000000000000000_2 = 0$	$00010101010101010000000000000000$ **2** = 715784192_{10}

Код программы:

```
// Возвращает целое число, последовательность нечетных битов которого
// представляет собой значение sh1, а последовательность четных – значение
sh2.
unsigned long long twoBitHybrid(long long sh1, long long sh2) {
  unsigned long long result = 0;
  for (int i = 0; i < 32; i += 2) {
    result += sh2 % 10 << i;
    result += sh1 % 10 << (i + 1);
    sh2 /= 10;
    sh1 /= 10;
  }
  return result;
}
```

№18. Вывести восьмеричное представление записи числа `x`

Спецификация для функции `twoBitHybrid`**:**

Заголовок: `int octNum(int x)`

Назначение: Выводит восьмеричное представление числа `x`.

Тестовые задания для задачи:

Входные данные	Ожидаемый результат
42	52
66	102
100	144
128	200
668	1234
1337	2471
2850	5442
342391	1234567

Код программы:

```
// Выводит восьмеричное представление числа x
void octNum(int x) {
  while (x > 0) {
    printf("%d", x & 7);
    x >>= 3;
  }
}
```

№19. Определить максимальную длину последовательности подряд идущих битов, равных единице в двоичном представлении данного целого числа.

Спецификация для функции `twoBitHybrid`**:**

Заголовок: `int binSequence(int x)`

Назначение: Возвращает максимальную длину последовательности подряд идущих битов равных единице, в представлении числа `x`.

Тестовые задания для задачи

Входные данные	Ожидаемый результат
$x = 61454_{10} = 1111000000001110_2$	4
$x = 11_{10} = 1011_2$	2

Код программы:

```
// Возвращает максимальную длину последовательности подряд идущих битов
// равных единице в двоичном представлении числа x.
int binSequence(int x) {
  int maxSequence = 0, lastSequence = 0;
  while (x > 0) {
    if (x & 1) {
      lastSequence++;
      maxSequence = max2(lastSequence, maxSequence);
    } else
      lastSequence = 0;

    x >>= 1;
  }
  return maxSequence;
}
```

Вывод: Получил навыки написания функций при решении простых задач. Закрепил навыки разработки алгоритмов разветвляющейся и циклической структуры. Получил навыки формулирования спецификаций к разрабатываемым функциям.