

Министерство образования и науки РФ
Государственное образовательное учреждение
высшего профессионального образования

Тульский государственный университет

КАФЕДРА АВТОМАТИКИ И ТЕЛЕМЕХАНИКИ

**ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ ЛИНЕЙНОЙ, ВЕТВЯЩЕЙСЯ И
ЦИКЛИЧЕСКОЙ СТРУКТУРЫ**

Лабораторная работа № 8
по курсу «Программирование на ЯВУ»

Вариант № 4

Выполнил: студент группы 220601

(подпись) Белым А.А.

Проверил: к. ф.-м. н., доцент

(подпись) Сулимова В.В.

Тула 2011

Цель работы

Целью работы заключается в том, чтобы научиться использовать стандартные функции и арифметические операторы, изучить операторы отношения и присваивания, логические операции, условный оператор и оператор-переключатель, научиться использовать в программах операторы цикла. Написать программы на изученные темы.

Задание

1. Задан вектор с координатами (x_1, y_1) и (x_2, y_2) . Определить угол наклона вектора к оси ОХ.
2. По координатам трех вершин треугольника найти его площадь и периметр, если такой треугольник может существовать.
3. Найти 2 числа Фибоначчи, удовлетворяющих условию при заданном m : $\phi(i) < m < \phi(i+1)$, вывести эти числа и их порядковые номера.

1. ЗАДАЧА ВЫЧИСЛЕНИЯ УГЛА НАКЛОНА ВЕКТОРА К ОСИ

ОХ

1.1. Схема алгоритма

На рисунке 1.1 представлена схема алгоритма поиска угла наклона вектора, заданного координатами начала и конца, к оси ОХ.

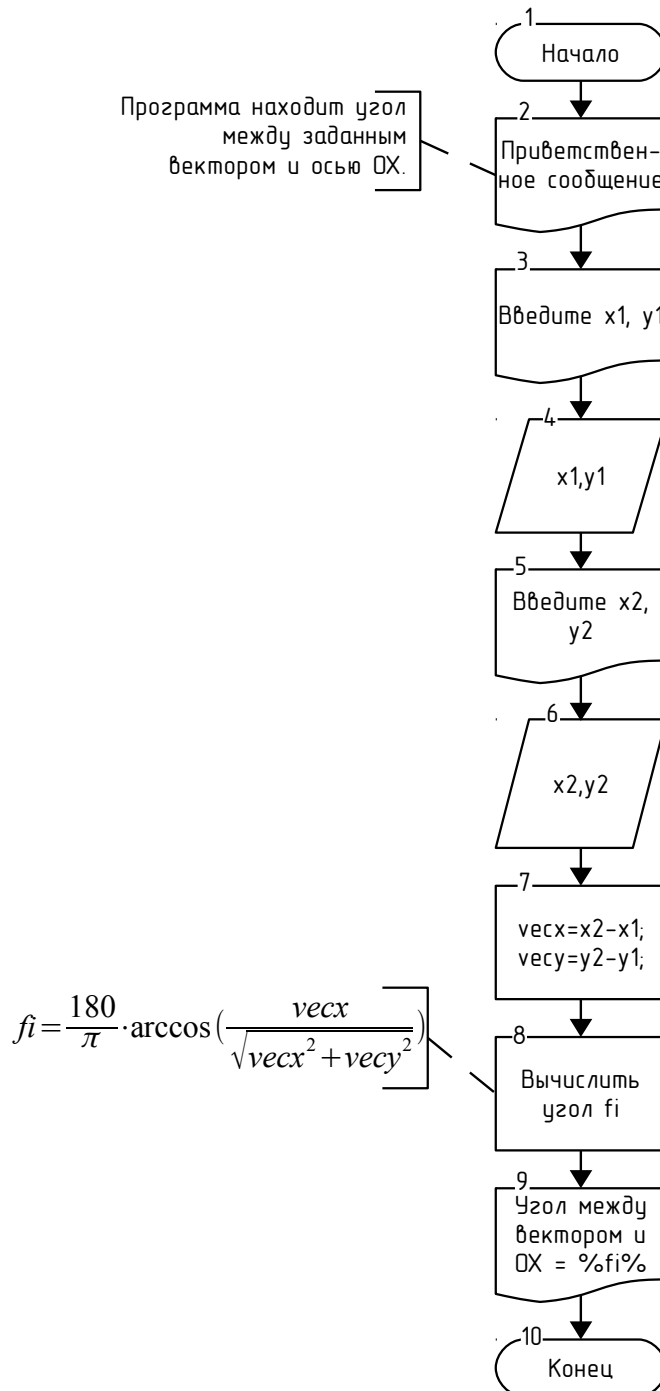


Рисунок 1.1 — Блок-схема поиска угла наклона вектора к оси ОХ

1.2. Инструкция пользователю

Данная программа позволяет найти угол наклона вектора, заданного координатами начала и конца, к оси ОХ.

Для работы программе необходимо передать координаты начала, а затем и конца вектора — 2 пары вещественных чисел. Если при вводе произойдет ошибка, и будет введено не вещественное число, программа завершится с ошибкой.

После окончания работы программа выведет искомый угол в градусах.

1.3. Инструкция программисту

При создании программы поиска угла наклона вектора к оси ОХ были предприняты следующие действия.

Были импортированы заголовочные файлы `stdio.h` - для функций ввода-вывода, `math.h` для функций `acos` и `sqrt` и константы `M_PI`.

Были введены структуры данных, описание которых представлено в таблице 1.1.

Таблица 1.1 - Структуры данных, используемые в в основной части программы поиска угла наклона вектора к оси ОХ

имя	тип	предназначение
x1,y1	float	Координаты начала вектора.
x2,y2	float	Координаты конца вектора.
vecx, vecy	float	Координаты вектора.
fi	float	Угол между вектором и осью ОХ.

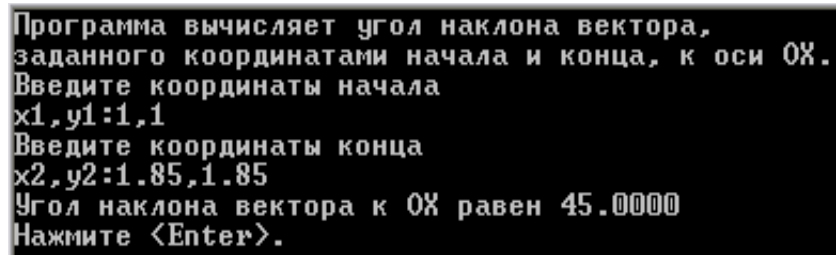
1.4. Текст программы

Ниже представлен текст программы, написанной на языке ANSI C для компиляторов Borland C++ 3.1 и GCC 4.5.2 (Linux 2.6.37 x86_64 и MinGW Windows 2000 SP4), которая находит угол наклона вектора, заданного координатами начала и конца, к оси OX.

```
#include <stdio.h>
#include <math.h>
int main(void) {
    float x1,x2,y1,y2,vecx,vecy,fi;
    printf("Программа вычисляет угол наклона вектора,
           заданного координатами начала и конца, к оси OX.\n");
    printf("Введите координаты начала\nx1,y1:");
    scanf("%f,%f",&x1,&y1);
    printf("Введите координаты конца\nx2,y2:");
    scanf("%f,%f",&x2,&y2);
    vecx=x2-x1;
    vecy=y2-y1;
    fi=acos(vecx/sqrt(vecx*vecx+vecy*vecy))*180/M_PI;
    printf("Угол наклона вектора к OX равен %.4f\n",fi);
    printf("Нажмите <Enter>.\n");
    (void) getchar();
    return 0;
}
```

1.5. Тестовый пример

На рисунке 1.2 представлен пример работы программы для вектора $(1;1) - (1,85;1,85)$ (угол наклона этого вектора — 45 градусов).



```
Программа вычисляет угол наклона вектора,  
заданного координатами начала и конца, к оси OX.  
Введите координаты начала  
x1,y1:1,1  
Введите координаты конца  
x2,y2:1.85,1.85  
Угол наклона вектора к OX равен 45.0000  
Нажмите <Enter>.
```

Рисунок 1.2 - Пример работы программы для вектора $\{0,85;0,85\}$

2. ЗАДАЧА ВЫЧИСЛЕНИЯ ПЛОЩАДИ И ПЕРИМЕТРА ТРЕУГОЛЬНИКА, ЗАДАННОГО КООРДИНАТАМИ ВЕРШИН

2.1. Схема алгоритма

На рисунке 2.1 представлена схема алгоритма поиска площади и периметра треугольника, заданного координатами трёх его вершин.

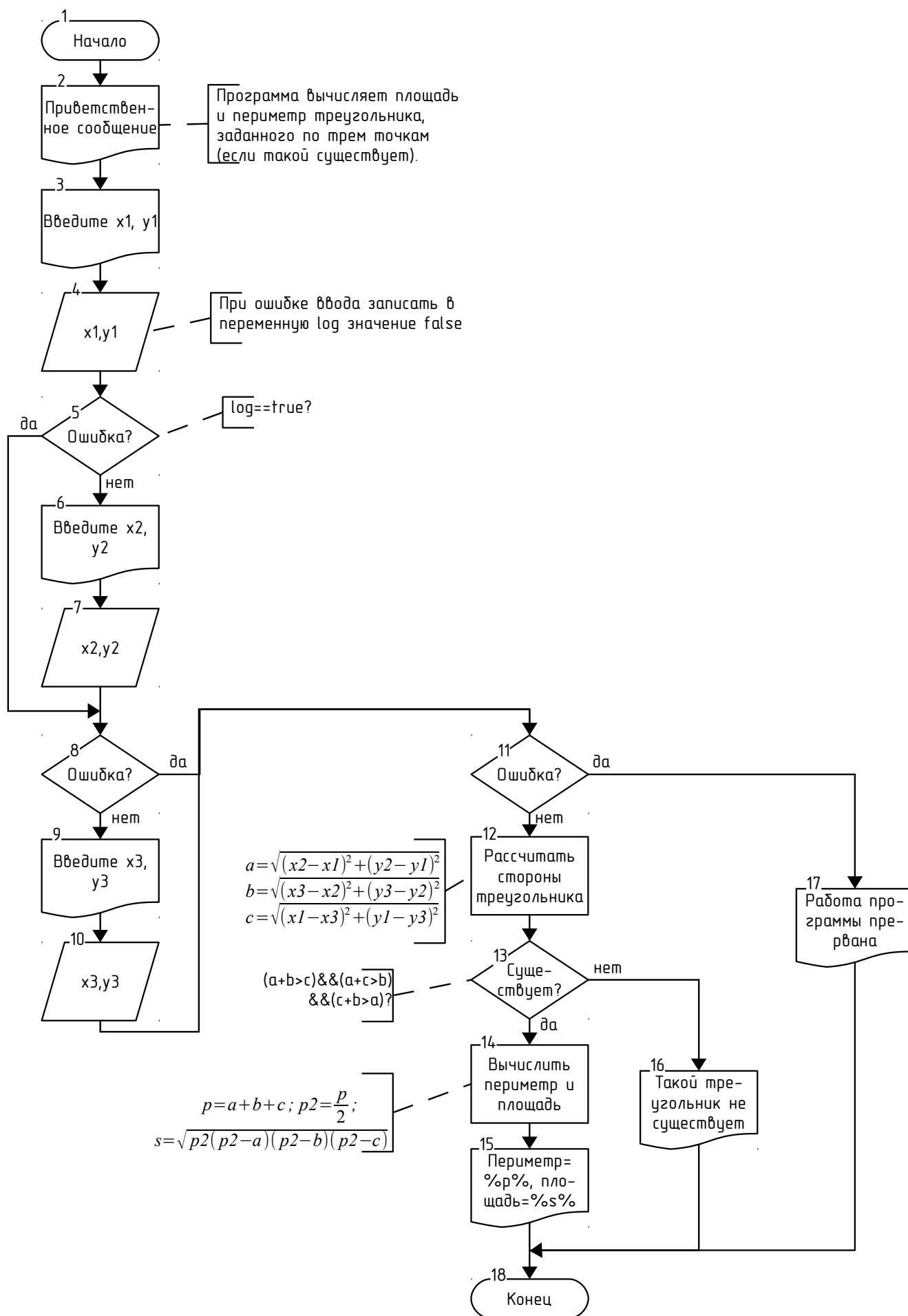


Рисунок 2.1 — Блок-схема поиска площади и периметра треугольника

2.2. Инструкция пользователю

Данная программа позволяет найти площадь и периметр треугольника, заданного координатами трёх его вершин, если, разумеется, такой треугольник существуют.

Для работы программе необходимо передать координаты всех трех вершин треугольника — 3 пары вещественных чисел. Если при вводе произойдет ошибка, и будет введено не вещественное число, программа сообщит о ошибке ввода и завершит работу.

Если треугольник с указанными вершинами может существовать, то после окончания работы программа выведет искомые площадь и периметр треугольника, иначе программа сообщит, что указанный треугольник не существует.

2.3. Инструкция программисту

При создании программы поиска площади и периметра треугольника были предприняты следующие действия.

Были импортированы заголовочные файлы `stdio.h` - для функций ввода-вывода, `math.h` для функции `sqrt`.

Были введены структуры данных, описание которых представлено в таблице 2.1.

Таблица 2.1 - Структуры данных, используемые в в основной части программы поиска площади и периметра треугольника

имя	тип	предназначение
x1,y1	float	Координаты первой вершины треугольника.
x2,y2	float	Координаты второй вершины треугольника.
x3,y3	float	Координаты третьей вершины треугольника.
a,b,c	float	Длины сторон треугольника.
p,p2,s	float	Периметр, полупериметр и площадь треугольника.
log	int	Флаг ошибки — 0, если произошла ошибка ввода, 1 — нет ошибки.

2.4. Текст программы

Ниже представлен текст программы, написанной на языке ANSI C для компиляторов Borland C++ 3.1 и GCC 4.5.2 (Linux 2.6.37 x86_64 и MinGW Windows 2000 SP4), которая находит площадь и периметр треугольника, заданного координатами вершин.

```
#include <stdio.h>
#include <math.h>
int main(void) {
    float x1,x2,x3,
          y1,y2,y3,
          a,b,c,
          p,p2,s;
    int log;
    printf("Программа находит площадь и периметр треугольника,
           заданного координатами вершин,\n если такой треугольник существует.\n");
    printf("Введите координаты первой вершины треугольника\nx1,y1:");
    log=scanf("%f,%f",&x1,&y1)==2;
    while (!feof(stdin)&&(getc(stdin)!='\n'));
    if (log) {
        printf("Введите координаты второй вершины треугольника\nx2,y2:");
        log=scanf("%f,%f",&x2,&y2)==2;
        while (!feof(stdin)&&(getc(stdin)!='\n'));
    }
    if (log) {
        printf("Введите координаты третьей вершины треугольника\nx3,y3:");
        log=scanf("%f,%f",&x3,&y3)==2;
        while (!feof(stdin)&&(getc(stdin)!='\n'));
    }
    if (log) {
        a=sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1));
        b=sqrt((x3-x2)*(x3-x2)+(y3-y2)*(y3-y2));
        c=sqrt((x1-x3)*(x1-x3)+(y1-y3)*(y1-y3));
        if ((a+b>c)&&
            (a+c>b)&&
            (c+b>a)) {
            p=a+b+c;p2=p/2;
            s=sqrt(p2*(p2-a)*(p2-b)*(p2-c));
            printf("Периметр треугольника равен %.4f;\n
                   Площадь треугольника равна %.4f.\n",p,s);
        }
    }
    else {
        printf("Треугольник не существует.\n");
    }
}
```

```
    }  
} else  
    printf("Работа программы прервана.");  
printf("Нажмите <Enter>...");  
(void)getchar();  
return 0;  
}
```


2.5. Тестовый пример

На рисунке 2.2 представлен пример работы программы для «египетского» треугольника (прямоугольного со сторонами 3, 4, 5) с координатами (0;0),(3;0), (0;4)..

```
Программа вычисляет угол наклона вектора,  
заданного координатами начала и конца, к оси OX.  
Введите координаты начала  
x1,y1:1,1  
Введите координаты конца  
x2,y2:1.85,1.85  
Угол наклона вектора к OX равен 45.0000  
Нажмите <Enter>.
```

Рисунок 2.2 - Пример работы программы для «египетского» треугольника

На рисунке 2.3 представлен пример работы программы для случая ошибочного ввода.

```
Программа находит площадь и периметр треугольника,  
заданного координатами вершин,если такой треугольник существует.  
Введите координаты первой вершины треугольника  
x1,y1:fds  
Работа программы прервана.Нажмите <Enter>...
```

Рисунок 2.3 — Пример работы программы вычисления параметров треугольника для ошибочного ввода

На рисунке 2.4 представлен пример работы программы для случая несуществования треугольника.

```
Программа находит площадь и периметр треугольника,  
заданного координатами вершин,если такой треугольник существует.  
Введите координаты первой вершины треугольника  
x1,y1:1,1  
Введите координаты второй вершины треугольника  
x2,y2:3,3  
Введите координаты третьей вершины треугольника  
x3,y3:2,2  
Треугольник не существует.  
Нажмите <Enter>...
```

Рисунок 2.4 — Пример работы программы для несуществующего треугольника

3.1 ЗАДАЧА ПОИСКА «СОСЕДНИХ» ЧИСЕЛ ФИБОНАЧЧИ

3.1. Схема алгоритма

На рисунке 2.1 представлена схема алгоритма поиска чисел Фибоначчи и их номеров, удовлетворяющих условию $\phi(i) < m < \phi(i+1)$, для заданного m - «соседних» для m .

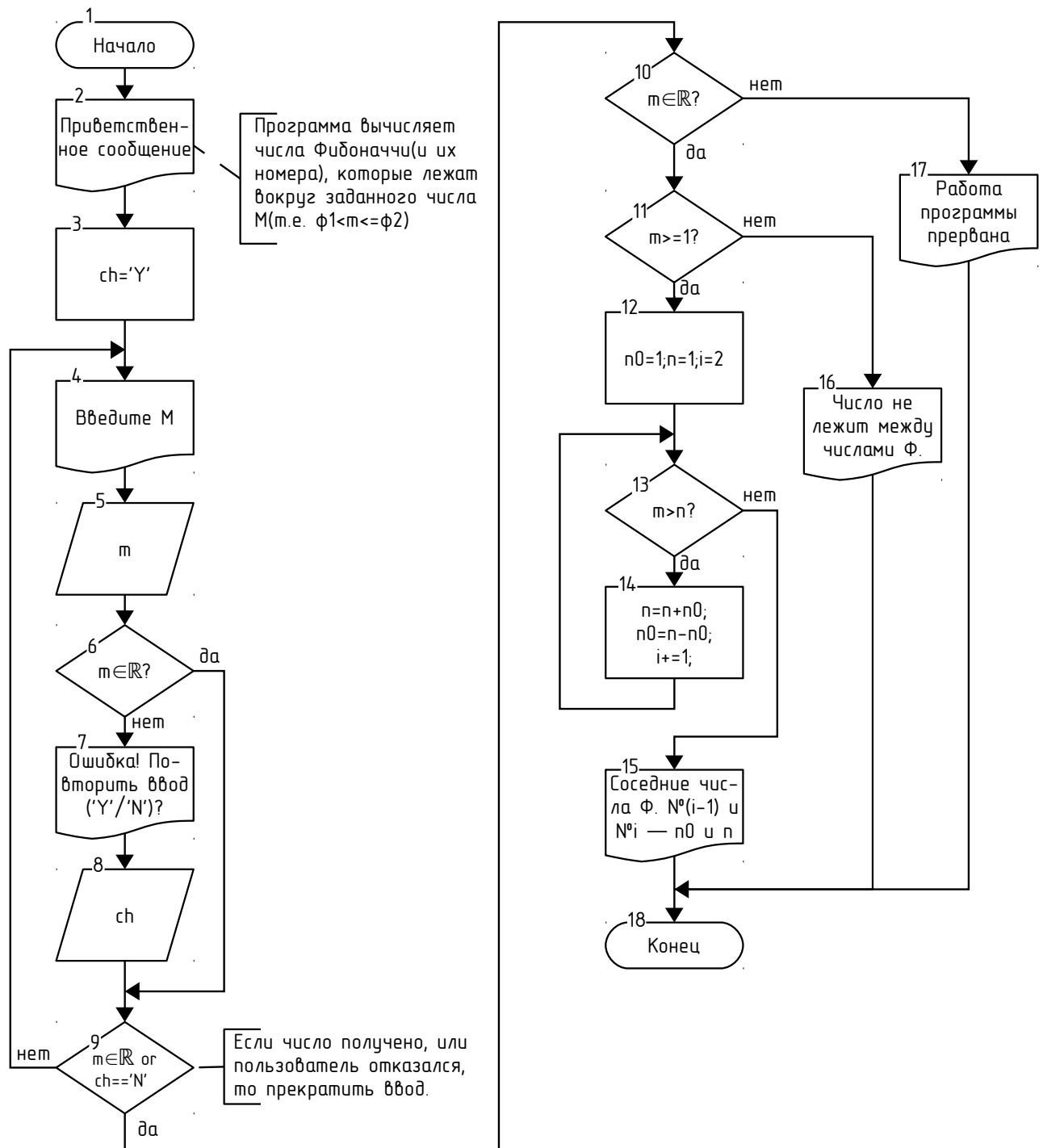


Рисунок 3.1 — Блок-схема алгоритма поиска «соседних» чисел Фибоначчи

3.2. Инструкция пользователю

Данная программа позволяет найти такие числа Фибоначчи и их номера, которые удовлетворяют условию $f(i) < m < f(i+1)$, где m – заданное число.

Для работы программе необходимо передать вещественное число m – для которого будут искаться «соседние» числа Фибоначчи. Если передано не вещественное число, то предоставляется возможность повторного ввода — на соответствующий запрос надо ответить 'Y'. Можно отказаться от повторного ввода, ответив 'N', тогда ввод считается неудачным.

Если число m запрошено успешно и оно не меньше 1, то будут сообщены числа Фибоначчи, удовлетворяющие вышеуказанному условию, и их номера. Если число меньше 1, то будет выведено сообщение, что для данного числа не существует «соседних» чисел Фибоначчи. Если ввод был неудачным, то сообщается о прерывании программы.

3.3. Инструкция программисту

При создании программы поиска «соседних» чисел Фибоначчи были предприняты следующие действия.

Были импортированы заголовочные файлы `stdio.h` - для функций ввода-вывода, `ctype.h` для функции `tolower`.

Были введены структуры данных, описание которых представлено в таблице 3.1.

Таблица 3.1 - Структуры данных, используемые в в основной части программы поиска «соседних» чисел Фибоначчи

имя	тип	предназначение
m	float	Число, для которого ищутся соседние числа Фибоначчи.
i	long	Номер последнего вычисленного числа Фибоначчи
n0,n	long	Предыдущее и текущее вычисленные числа Фибоначчи .
log	int	Флаг ошибки — 0, если произошла ошибка ввода, 1 — нет ошибки.
ch	char	Ответ пользователя на запрос о повторении ввода.

3.4. Текст программы

Ниже представлен текст программы, написанной на языке ANSI C для компиляторов Borland C++ 3.1 и GCC 4.5.2 (Linux 2.6.37 x86_64 и MinGW Windows 2000 SP4), которая находит числа Фибоначчи и их номера, удовлетворяющие условию $\phi(i) < m < \phi(i+1)$ для заданного m .

```
#include <stdio.h>
#include <ctype.h>
int main(void) {
    long n0,n,i;
    float m;
    char ch='Y'; int log;
    printf("Данная программа позволяет найти такие числа Фибоначчи(и их номера),\n
        которые удовлетворяют условию  $\phi(i) < m < \phi(i+1)$ , где  $m$  - заданное число.\n");
    do{
        printf("Введите m\nm:"); log=scanf("%f",&m)==1;
        while (!feof(stdin)&&(getc(stdin)!='\n'));
        if (!log){
            printf("Ошибка! Повторить ввод?Y\\N\n");
            ch=getchar();
            while (!feof(stdin)&&(getc(stdin)!='\n'));
        }
    } while ((toupper(ch)!='N')&&!log);
    if (log){
        if (m>=1){
            n0=1;n=1;i=2;
            while (m>n){
                n0=(n+=n0)-n0;
                i++;
            }
            printf("Номера чисел: %ld и %ld\n"
                " $\phi$ (%ld) = %ld,  $\phi$ (%ld) = %ld.\n",
                i-1,i,i-1,n0,i,n);
        } else
            printf("Число не лежит между числами Фибоначчи.\n");
    } else
        printf("Работа программы прервана.\n");
    printf("Нажмите <Enter>.\n");
    (void) getchar();
    return 0;
}
```

3.5. Тестовый пример

На рисунке 3.2 представлен пример работы программы для числа m , равного 3.14.

```
Данная программа позволяет найти такие числа Фибоначчи(и их номера),  
которые удовлетворяют условию  $\phi(i) < m < \phi(i+1)$ , где  $m$  – заданное число.  
Введите  $m$   
 $m$ :3.14  
Номера чисел: 4 и 5  
 $\phi(4) = 3$ ,  $\phi(5) = 5$ .  
Нажмите <Enter>.
```

Рисунок 3.2 - Пример работы программы для числа 3.14

На рисунке 3.3 представлен пример работы программы для $m=1$.

```
Данная программа позволяет найти такие числа Фибоначчи(и их номера),  
которые удовлетворяют условию  $\phi(i) < m < \phi(i+1)$ , где  $m$  – заданное число.  
Введите  $m$   
 $m$ :1  
Номера чисел: 1 и 2  
 $\phi(1) = 1$ ,  $\phi(2) = 1$ .  
Нажмите <Enter>.
```

Рисунок 3.3 — Пример работы программы для $m=1$

На рисунке 3.4 представлен пример работы программы для $m=-42$.

```
Данная программа позволяет найти такие числа Фибоначчи(и их номера),  
которые удовлетворяют условию  $\phi(i) < m < \phi(i+1)$ , где  $m$  – заданное число.  
Введите  $m$   
 $m$ :-42  
Число не лежит между числами Фибоначчи.  
Нажмите <Enter>.
```

Рисунок 3.4 — Пример работы программы для $m=-42$

На рисунке 3.5 представлен пример работы программы при повторении ввода данных и вводе правильного числа, а на рисунке 3.6 — отказе от повторного ввода.

```
Данная программа позволяет найти такие числа Фибоначчи(и их номера),  
которые удовлетворяют условию  $\phi(i) < m < \phi(i+1)$ , где  $m$  – заданное число.  
Введите  $m$   
 $m$ :ddgg  
Ошибка! Повторить ввод?Y\N  
y  
Введите  $m$   
 $m$ :100  
Номера чисел: 11 и 12  
 $\phi(11) = 89$ ,  $\phi(12) = 144$ .  
Нажмите <Enter>.
```

Рисунок 3.5 — Результат работы программы при повторном корректном вводе данных

```
Данная программа позволяет найти такие числа Фибоначчи(и их номера),  
которые удовлетворяют условию  $\phi(i) < m < \phi(i+1)$ , где  $m$  – заданное число.  
Введите m  
m:ffgdfhj  
Ошибка! Повторить ввод?Y\N  
y  
Введите m  
m:d3343hj  
Ошибка! Повторить ввод?Y\N  
n  
Работа программы прервана.  
Нажмите <Enter>.
```

Рисунок 3.6 — Результат работы программы при повторном некорректном вводе данных

Вывод

В ходе выполнения данной лабораторной работы я познакомился с функциями стандартных библиотек языка Си, изучил операторы ветвления и цикла этого языка. По сравнению с языком Паскаль данные синтаксические конструкции Си более компактны и мощны; обратной стороной такой ситуации является то, что нестандартное применение таких конструкций может снижать читабельность кода. Также следует отметить и богатство функций библиотек Си по сравнению с Паскалем.