Министерство образования и науки РФ
Государственное образовательное учреждение
высшего профессионального образования

Тульский государственный университет

КАФЕДРА АВТОМАТИКИ И ТЕЛЕМЕХАНИКИ

СТРУКТУРЫ В ЯЗЫКЕ ПРОГРАММИРОВАНИЯ СИ

Лабораторная работа № 11

по курсу «Программирование на ЯВУ»

Вариант № 4

Выполнил: студент группы 220601 _____ Белым А.А.
(подпись)

Проверил: к. ф.-м. н., доцент _____ Сулимова В.В.
(подпись)

Тула 2011

Цель работы

Целью работы является изучение структурированного типа в языке Си, а также написать программу, использующую операции со структурами.

Задание

Описать переменную "студент", содержащую: имя, фамилию, отчество студента, название учебного заведения, номер группы. Создать список студентов (N>10). Определить фамилии студентов, учащихся в одной и той же группе, в одном и том же заведении.

Теоретическая справка

Схема алгоритма

На рисунке 1 представлена схема алгоритма ввода данных, поиска одногруппников среди данных студентов и вывода информации о них.

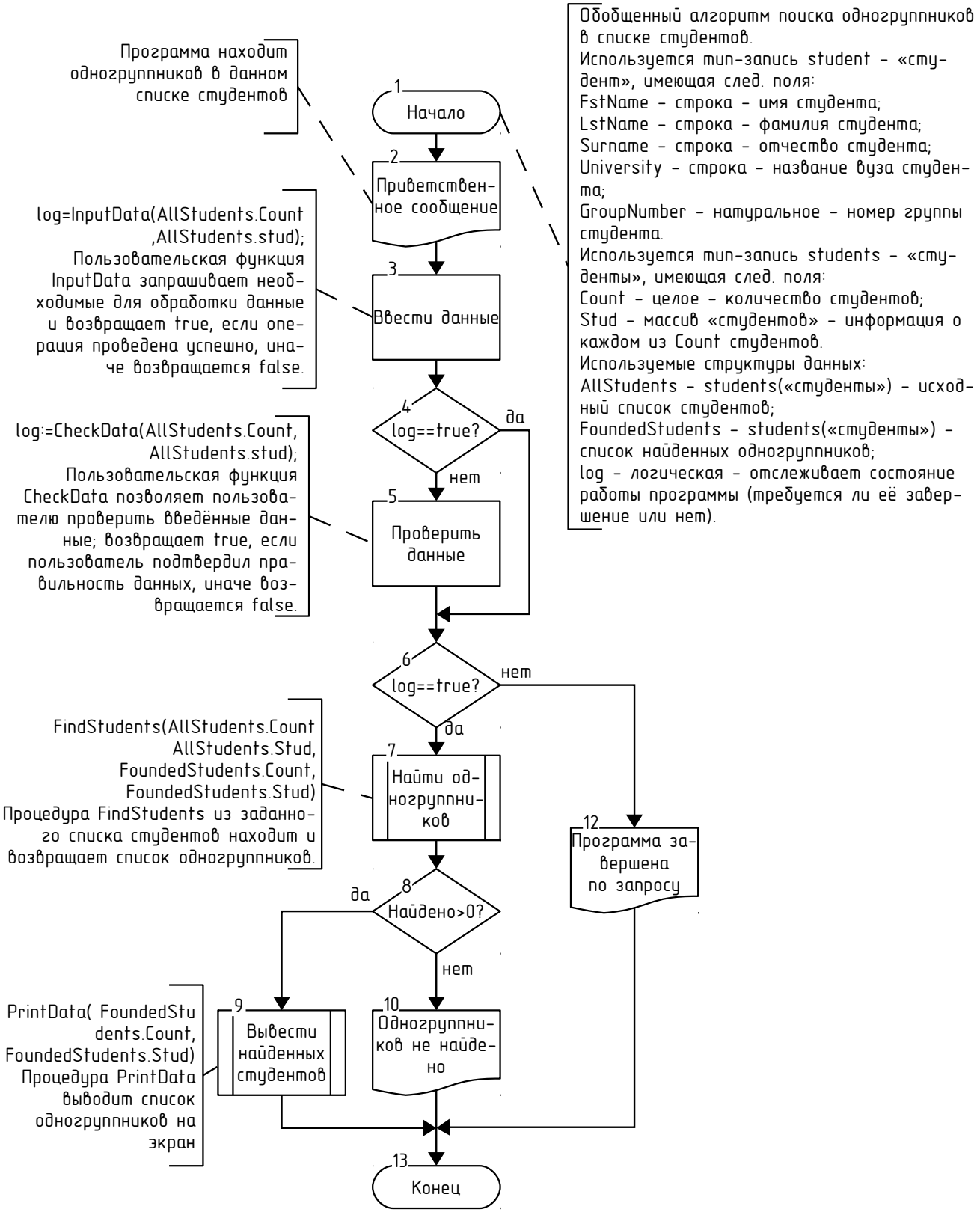

Рисунок 1 — Блок-схема обобщенного алгоритма поиска одногруппников

На рисунке 2 представлена схема алгоритма ввода списка студентов для поиска в них одногруппников.

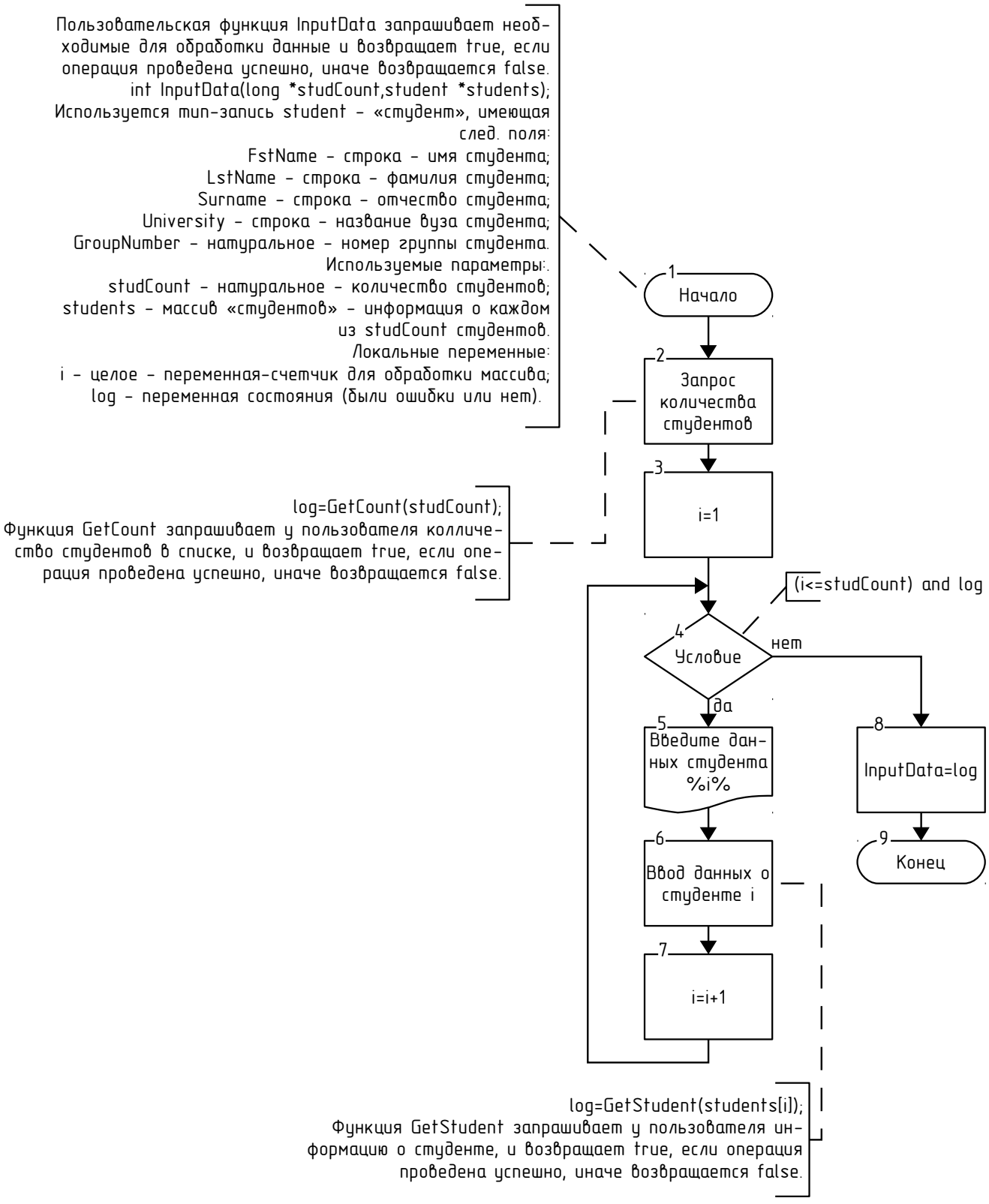

Рисунок 2 — Блок-схема алгоритма ввода списка студентов

На рисунке 3 представлена схема алгоритма ввода количества студентов в списке.

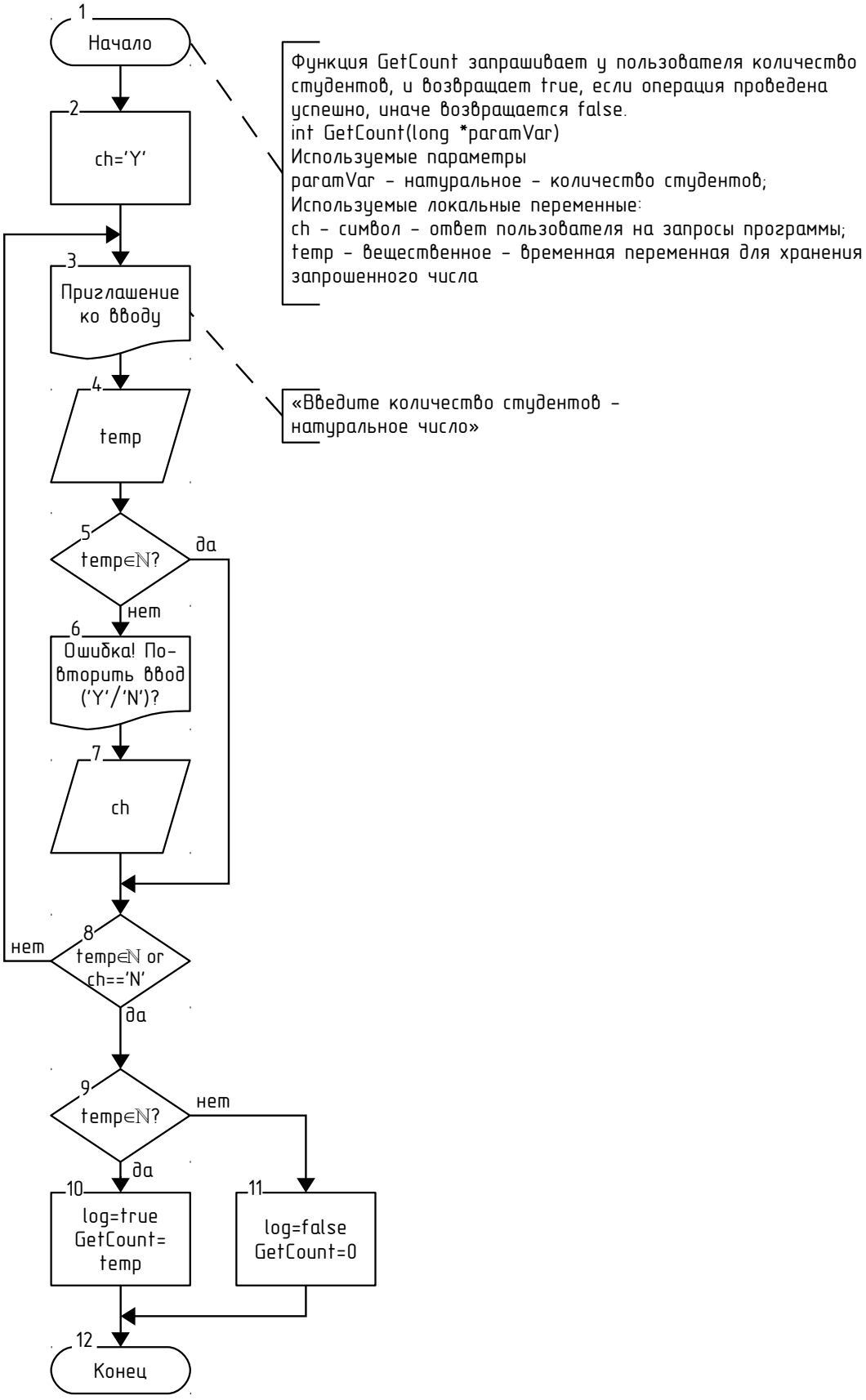

Рисунок 3 — Блок-схема алгоритма ввода количества студентов

На рисунке 4 представлена схема алгоритма ввода данных о конкретном студенте.

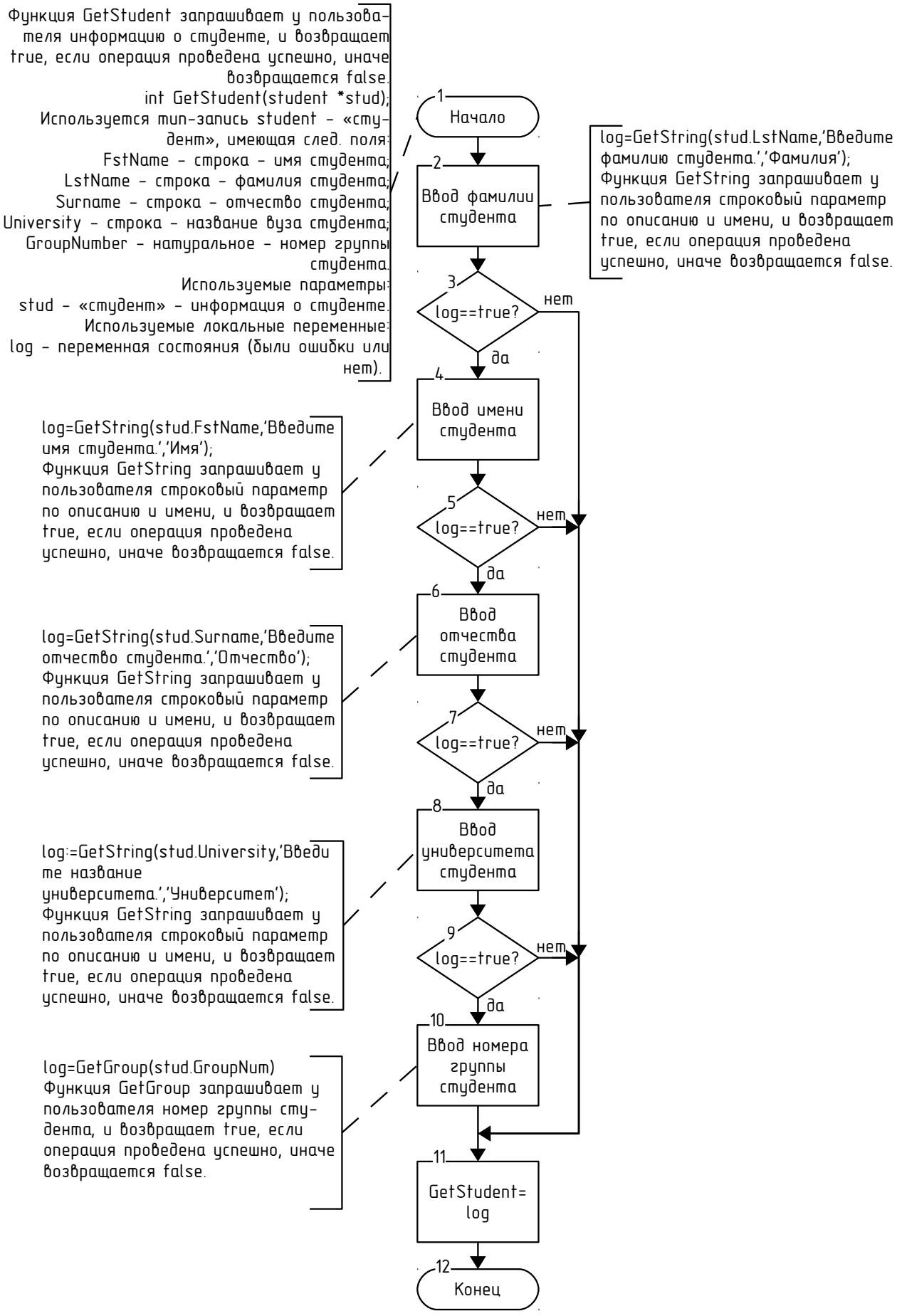

Рисунок 4 — Блок-схема алгоритма ввода данных о студенте

На рисунке 5 представлена блок-схема алгоритма запроса строкового параметра.

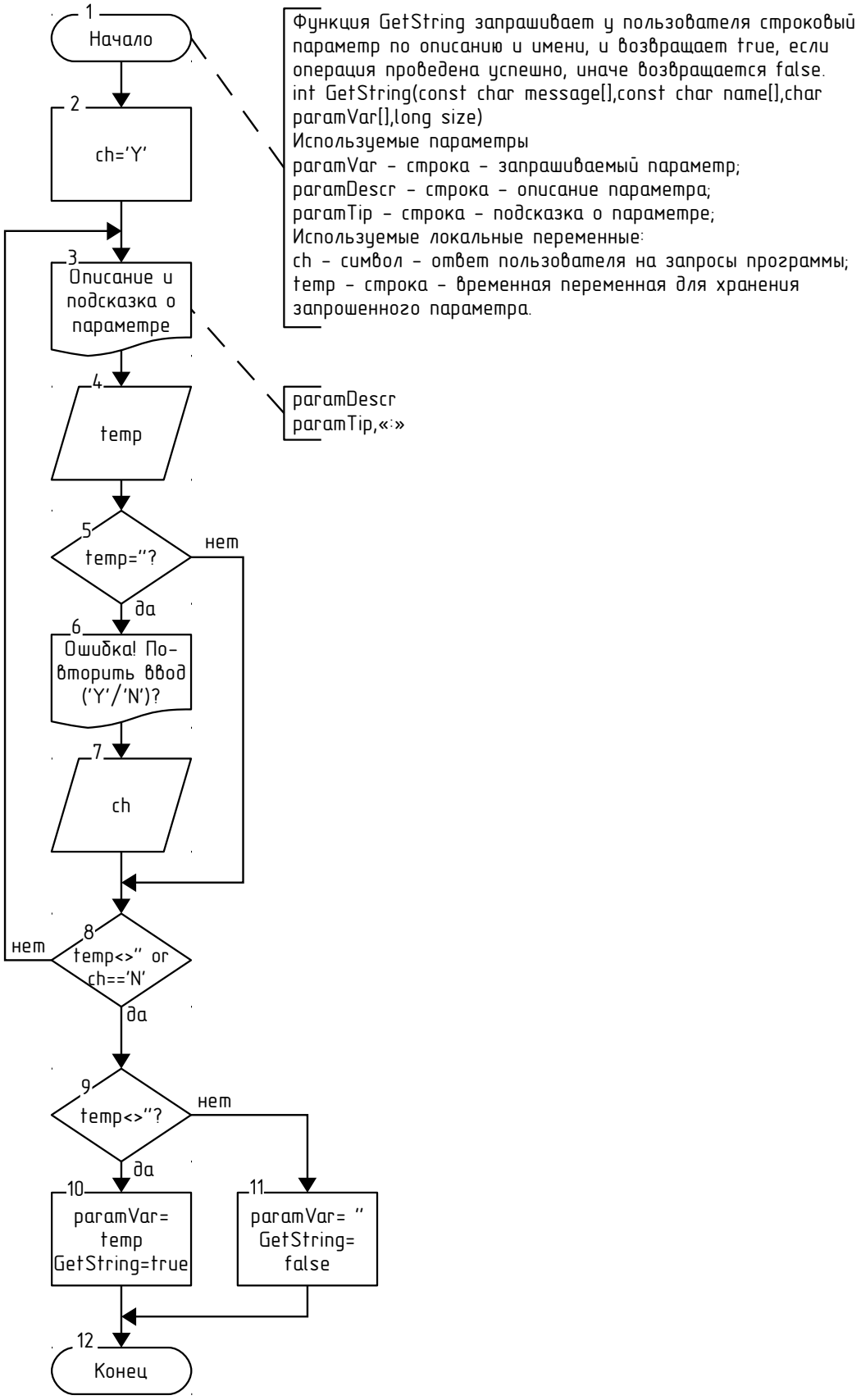

Рисунок 5 — Блок-схема алгоритма ввода строкового параметра

На рисунке 6 представлена схема алгоритма ввода номера группы студента.

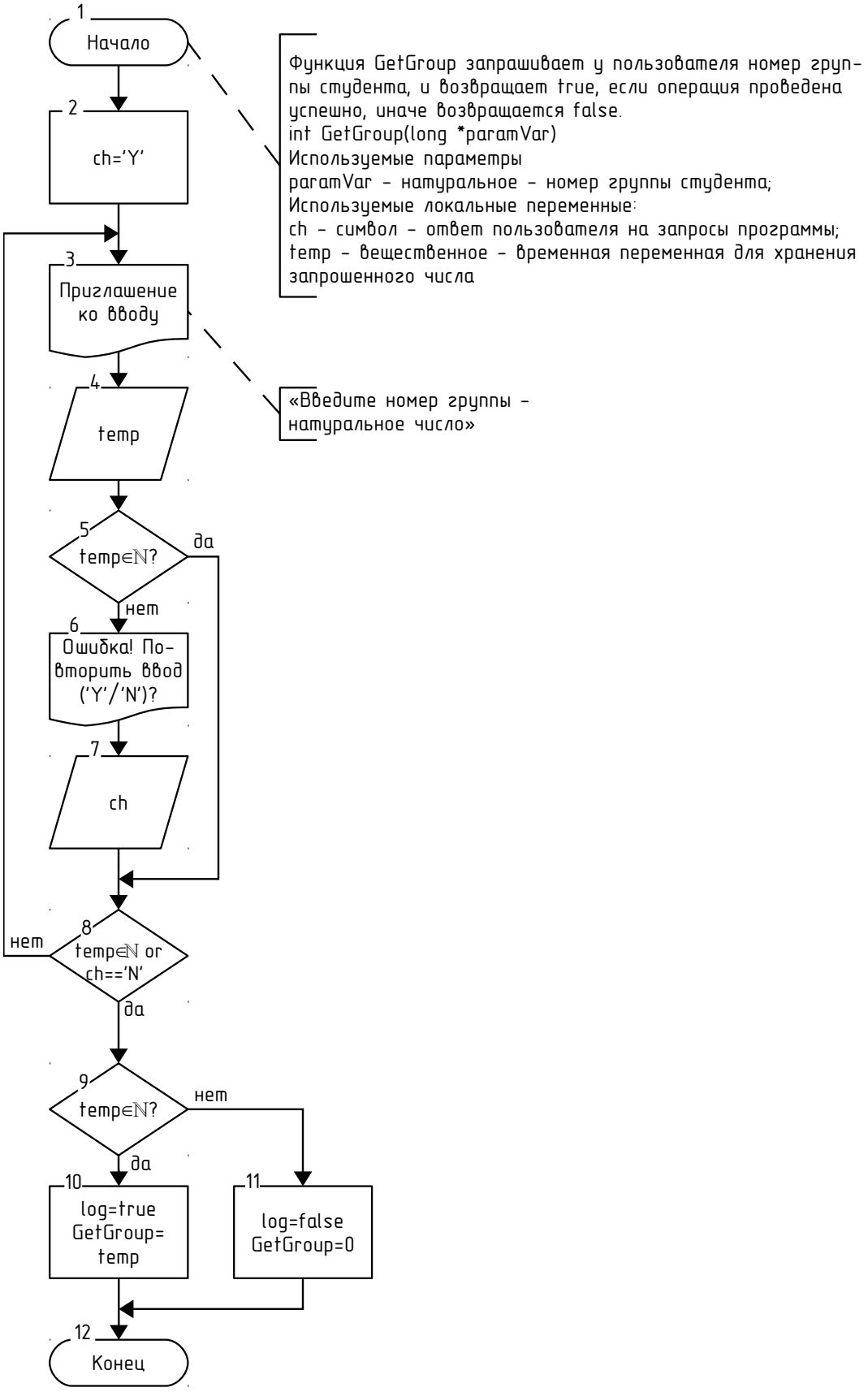

Рисунок 6 — Блок-схема алгоритма ввода номера группы

На рисунке 7 представлена схема алгоритма организации проверки пользователем введённых данных.

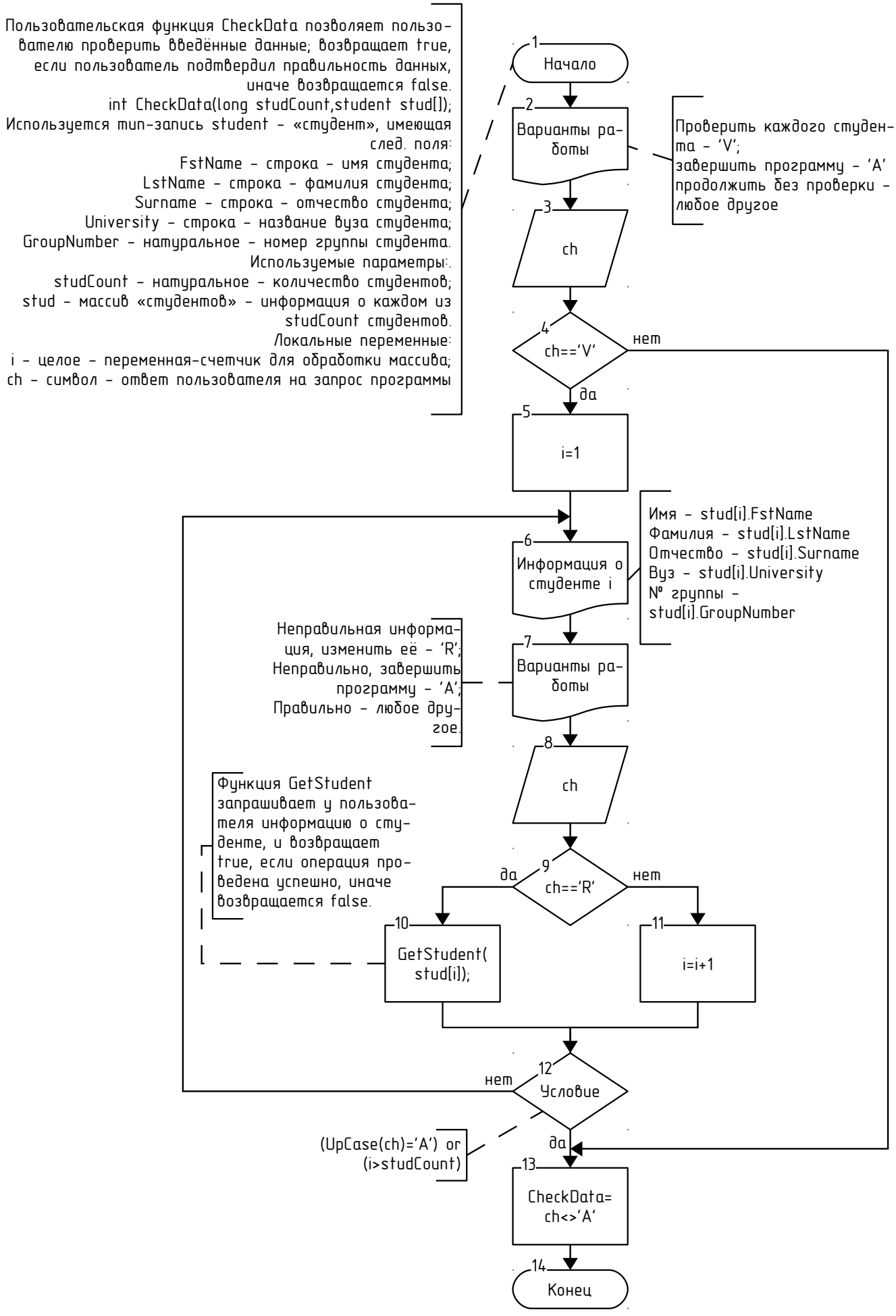

Рисунок 7 — Блок-схема алгоритма проверки пользователем введенных данных

На рисунке 8 представлена схема алгоритма поиска одногрупников среди заданного списка студентов.

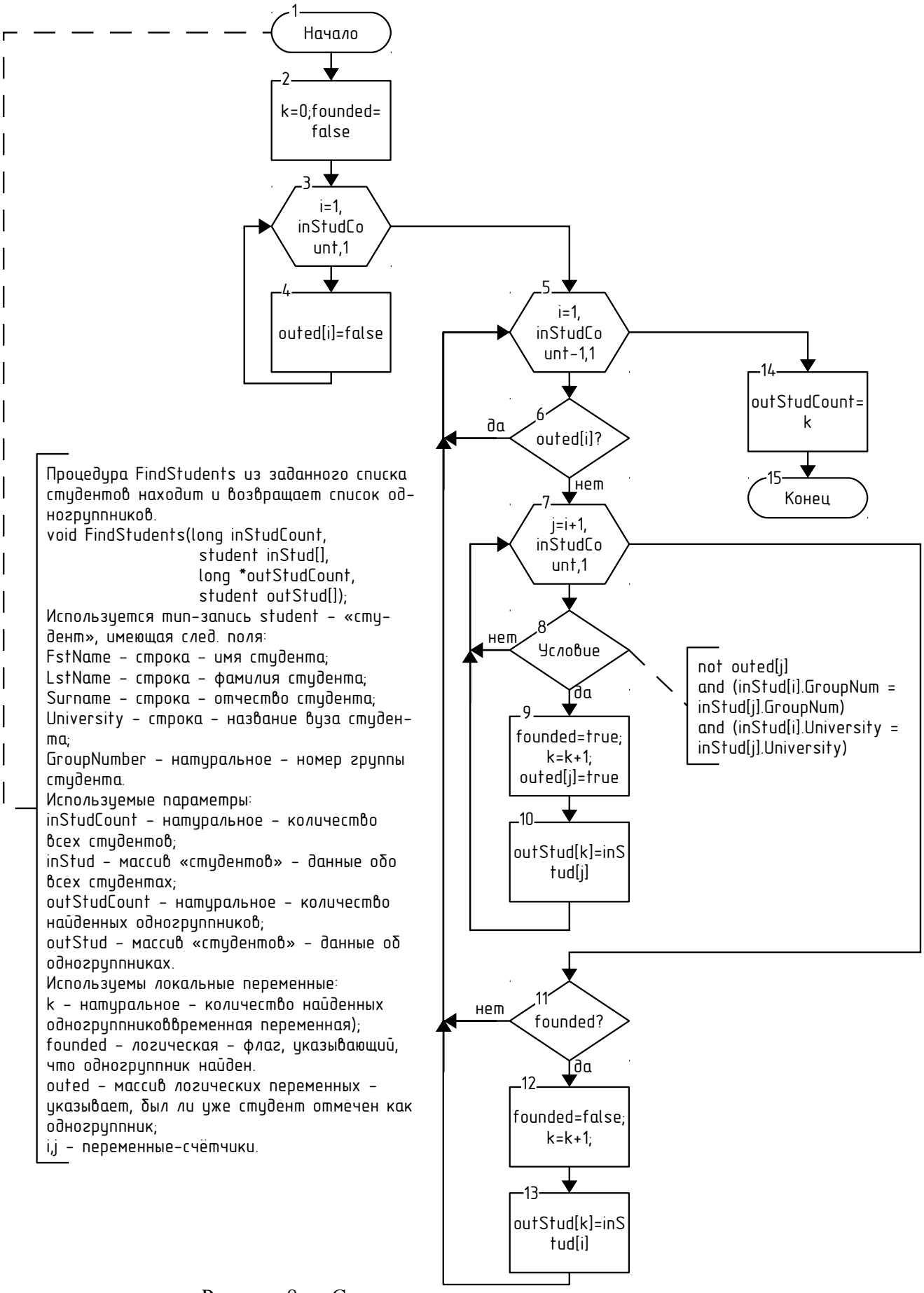

Рисунок 8 — Схема алгоритма поиска одногруппников

На рисунке 9 представлена схема алгоритма вывода данных о найденных одногруппниках на экран.

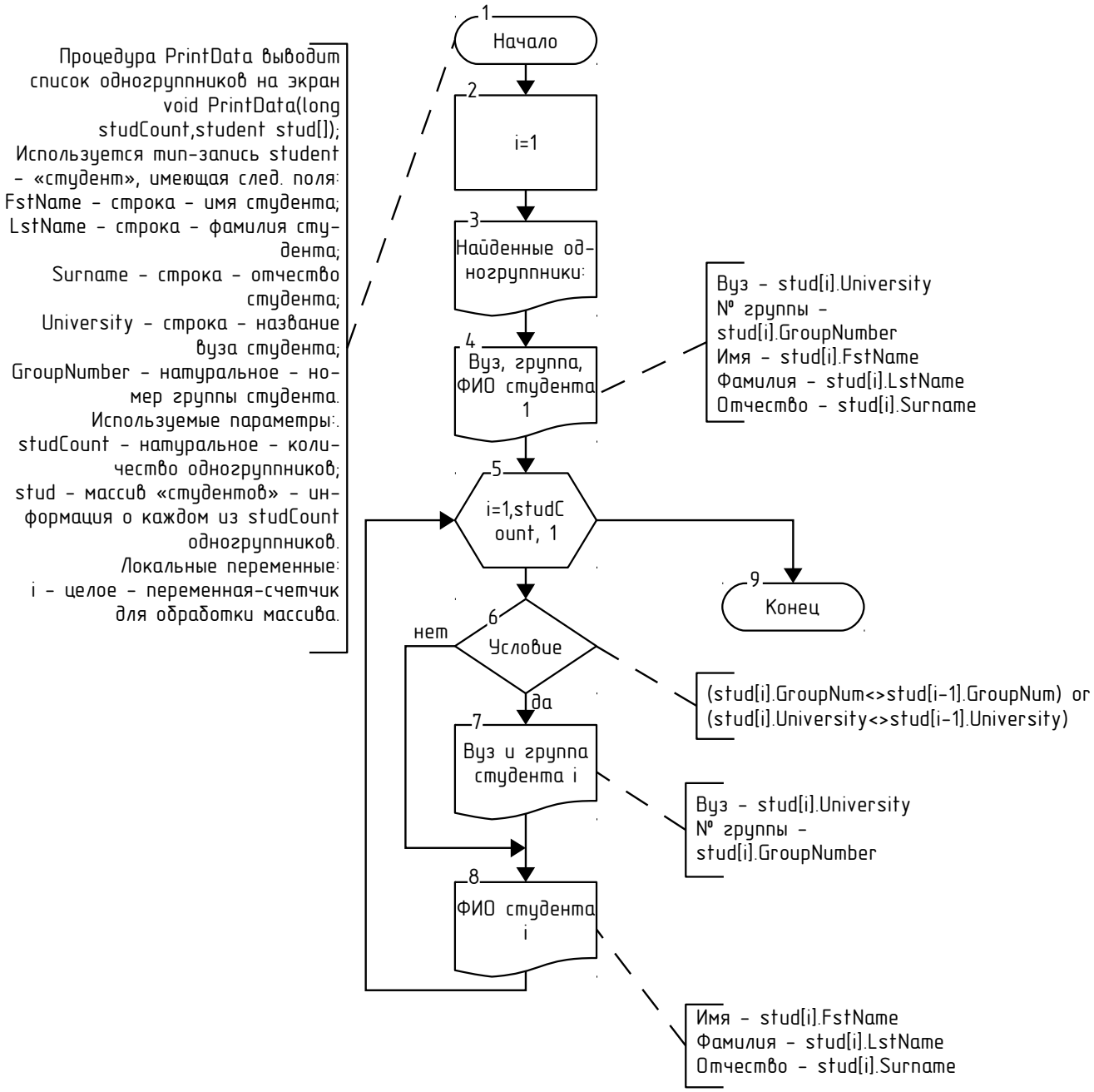

Рисунок 9 — Блок-схема алгоритма вывода данных об одногруппниках

Схему алгоритма получения ответа от пользователя на поставленный вопрос, можно увидеть на рисунке 10.

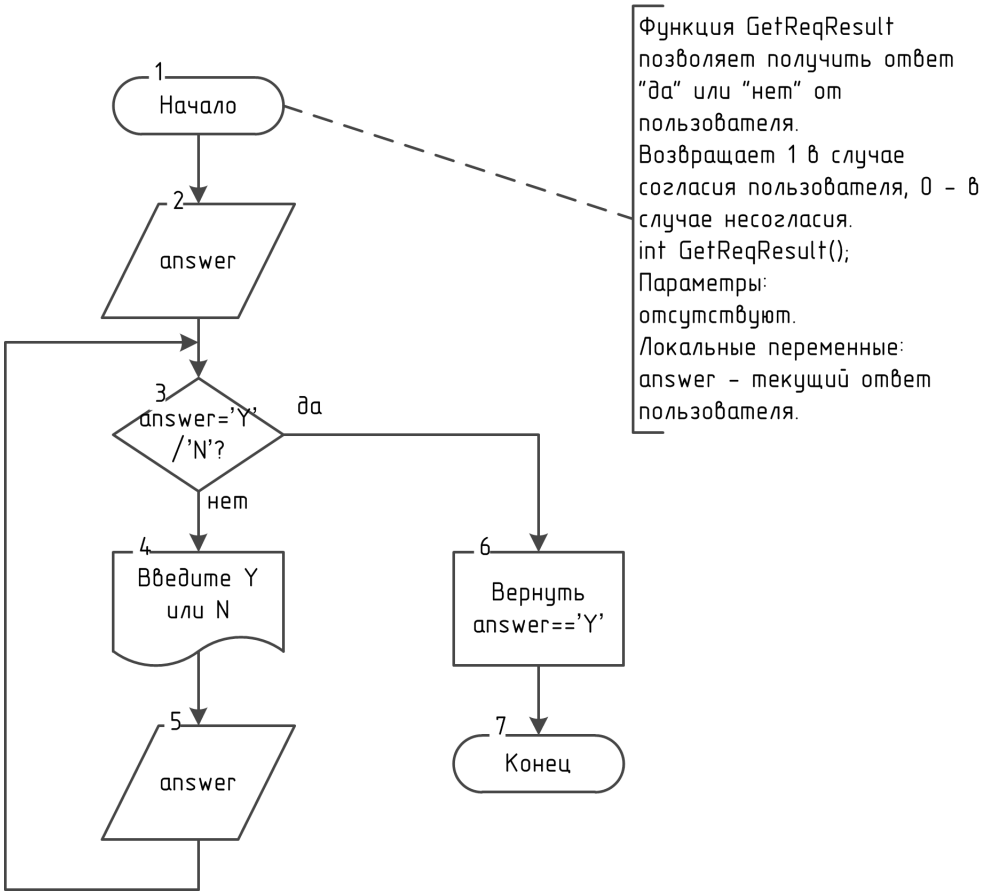

Рисунок 10 - Схема алгоритма получения ответа от пользователя

На рисунке 11 представлена схема алгоритма предоставления пользователю различных вариантов ответа.

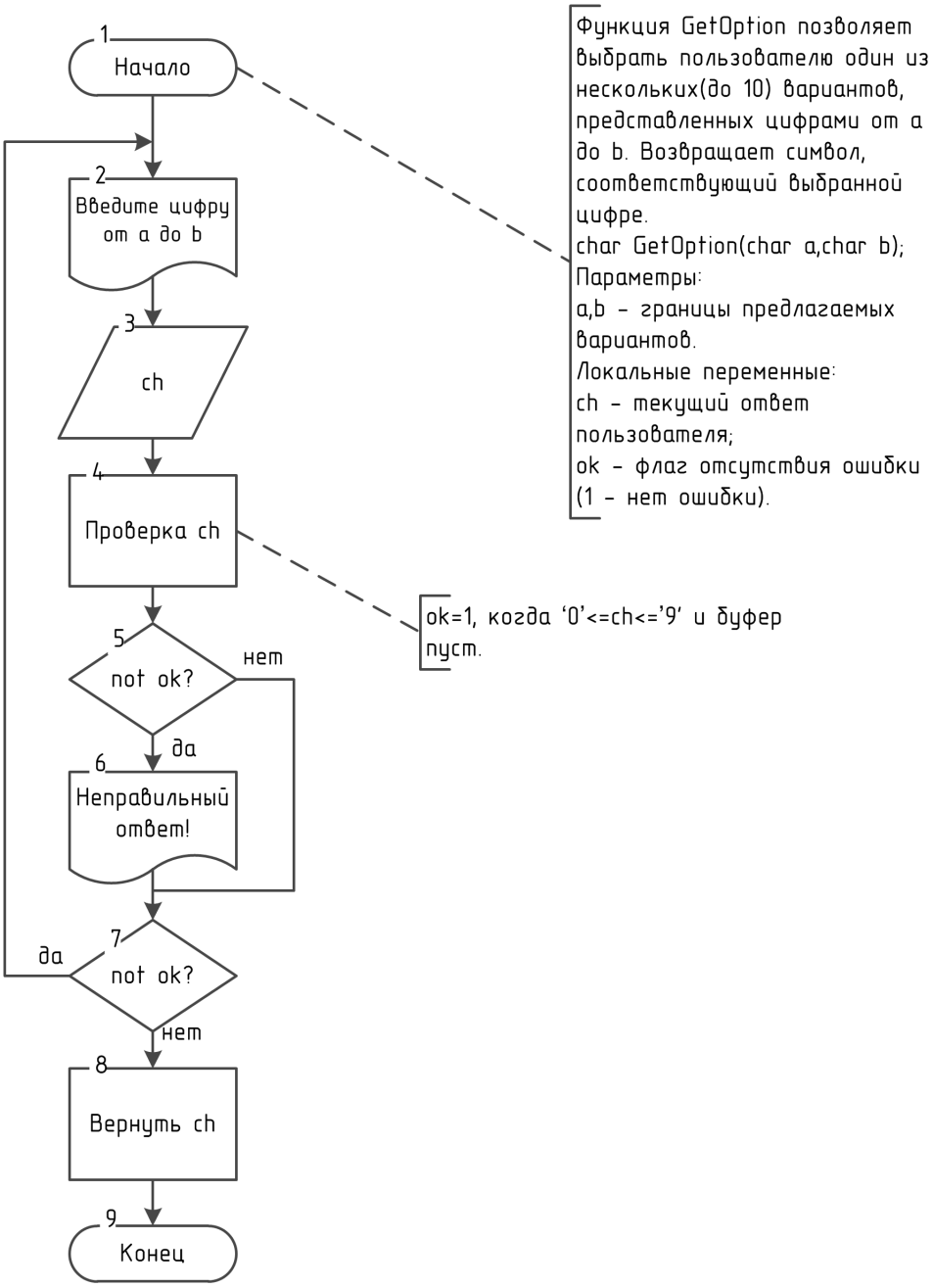

Рисунок 11 - Схема алгоритма предоставления пользователю различных вариантов ответа

Инструкция пользователю

Данная программа позволяет найти среди переданного ей списка студентов таких студентов, которые учатся в одном вузе в одной группе, т.е. являются одногруппниками.

Для работы программы необходимо ввести количество студентов, среди которых будет производится поиск одногруппников. Это натуральное число, большее нуля и меньшее определенного значения (определяется программой). Затем вводятся данные о каждом из студентов - его имя, фамилия, отчество, название учебного заведения и номер группы. Все параметры, кроме последнего - строковые значения, последнее - натуральное число. В случае неправильного ввода имеется возможность ввести заново данные, или отказаться от повторного воода и завершить программу. После этого предоставляется возможность проверить введённые данные для каждого студента поочередно, ответив на запрос программы '1', продолжить работу без проверки данных('2') или завершить работу программы(ответ '0').

В качестве результатов работы программа выведет список одногруппников: сначала выводятся общие вуз и номер группы, затем данные обучающихся в этой группе: и так делается для каждой различной группы в списке. Если одногруппников не будет найдено, программа выведет соответствующее сообщение.

Инструкция программисту

При создании программы поиска одногруппников были предприняты следующие действия.

В основной части программы определена константа MAX_N - максимальное количество студентов в задаваемом списке.

Определён тип-структура student, описывающий некоторого студента; поля этого типа представленны в таблице 1.

Таблица 1 - Поля типа-структуры, описывающего студента

имя	тип	предназначение
FstName, LstName, Surname	char[20]	Имя, фамилия, отчество студента соответственно.
University	char[50]	Название вуза студента.
GroupNum	long	Номер группы студента.

Локально определен тип-структура students, описывающий список студентов; поля этого типа представленны в таблице 2.

Таблица 2 - Поля типа-структуры, описывающего список студентов

имя	тип	предназначение
Count	long	Количество студентов в списке.
Stud	student[MAX_N]	Данные о каждом из Count студентов (естественно, Count<=nn)

Были введены структуры данных, описание которых представлено в таблице 3.

Таблица 3 - Структуры данных, используемые в в основной части программы поиска одногруппников

имя	тип	предназначение
AllStudents	students	Задаваемый пользователем список студентов
FoundedStudents	students	Список найденных одногруппников.
log	int	Указывает на отсутствие ошибок (значение true) ввода данных или желания пользователя прервать программу.

Кроме того, в процессе создания вышеуказанной программы были определены следующие подпрограммы:

1. Функция GetCount - запрашивает у пользователя количество студентов в передаваемом списке. Возвращает true, если пользователь передал корректное значение, иначе возвращается false. Для работы подпрограммы необходима константа nn.

int GetCount(long *paramVar);

В теле функции в цикле с постусловием происходит запрос у пользователя целого числа. Запрошенное значение сначала считывается в буфер, потом производится попытка его извлечения (функцией val). Если попытка была успешной, а также это число лежит в промежутке [0,nn], функция завершает работу и передает указанное значение и значение log, равное true . Если нет, то производится запрос пользователя о продолжении работы. Если пользователь ответил 'N', функция завершает работу и возвращает значение log, равное false и значение параметра - 0. Иначе операция повторяется до достижения двух вышеуказанных условий.

Используемые функцией параметры-переменные представленны - в таблице 4, локальные переменные - в таблице 5.

Таблица 4 - Параметры-переменные функции ввода количества студентов

имя	**тип**	**предназначение**
paramVar	long	Количество студентов в списке

Таблица 5 - Локальные переменные функции ввода ввода количества студентов

имя	**тип**	**предназначение**
req_rslt	int	Содержит ответ пользователя на запросы программы о повторении ввода данных(1 – "да", 0 - "нет").

2. Функция GetString запрашивает у пользователя строковый параметр - фамилию студента, имя и т.д. Описание параметра и его название(или подсказка) передается через параметры функции. Возвращает true, если пользователь передал корректное значение, иначе возвращается false.

int GetString(const char message[],const char name[],char paramVar[],long size)

В теле функции в цикле с постусловием происходит запрос у пользователя вещественного числа, причем сначала выводится переданная информация о параметре. Если введенная строка не пустая, функция завершает работу и

передает указанное значение и значение log, равное true . Если нет, то производится запрос пользователя о продолжении работы. Если пользователь ответил 'N', функция завершает работу и возвращает значение log, равное false и пустую строку. Иначе операция повторяется до достижения двух вышеуказанных условий.

Используемые функцией параметры-константы приведены в таблице 6; параметры-переменные - в таблице 7, локальные переменные - в таблице 8.

Таблица 6 - Параметры-константы функции ввода строкового параметра

имя	**тип**	**предназначение**
message	char*	Описание запрашиваемого параметра.
name	char*	Краткая подсказка о параметре.
size	long	Длина запрашиваемой строки.

Таблица 7 - Параметры-переменные функции ввода строкового параметра

имя	**тип**	**предназначение**
paramVar	char*	Запрашиваемый параметр.

Таблица 8 - Локальные переменные функции ввода строкового параметра

имя	**тип**	**предназначение**
req_rslt	int	Содержит ответ пользователя на запросы программы о повторении ввода данных(1 – "да", 0 - "нет").
log	int	Указывает на отсутствие ошибок (значение true) ввода данных.
format	char[15]	Форматная строка для считывания запрашиваемой строки.

3. Функция GetGroup запрашивает у пользователя номер группы студента. Возвращает true, если пользователь передал корректное значение, иначе возвращается false.

int GetGroup(long *paramVar);

В теле функции в цикле с постусловием происходит запрос у пользователя целого числа. Запрошенное значение сначала считывается в буфер, потом производится попытка его извлечения (функцией val). Если попытка была успешной, а также это число больше 0, функция завершает работу и передает указанное значение и значение log, равное true . Если нет, то производится запрос

пользователя о продолжении работы. Если пользователь ответил 'N', функция завершает работу и возвращает значение log, равное false и значение параметра - 0. Иначе операция повторяется до достижения двух вышеуказанных условий.

Используемые функцией параметры-переменные приведены в таблице 9, локальные переменные - в таблице 10.

Таблица 9 - Параметры-переменные функции ввода номера группы

имя	**тип**	**предназначение**
paramVar	long	Номер группы студента.

Таблица 10 - Локальные переменные функции ввода номера группы

имя	**тип**	**предназначение**
req_rslt	int	Содержит ответ пользователя на запросы программы о повторении ввода данных(1 – "да", 0 - "нет").
log	int	Указывает на отсутствие ошибок (значение true) ввода данных.

4. Функция GetStudent запрашивает у пользователя информацию о конкретном студенте. Возвращает true, если пользователь передал корректное значение, иначе возвращается false. Должен быть доступен тип “student”.

int GetStudent(student *stud);

В теле функции через GetString запрашиваются имя, фамилия, отчество, вуз студента и через GetGroup - номер группы студента. Если на каком-то этапе возникла ошибка, то последующие этапы не выполняются и возвращается false, иначе возвращается true.

Используемые функцией параметры-переменные приведены в таблице 11, локальные переменные - в таблице 12.

Таблица 11 - Параметры-переменные функции ввода произвольного параметра расчета корня

имя	**тип**	**предназначение**
stud	student*	Данные о студенте.

Таблица 12 - Локальные переменные функции ввода произвольного параметра расчета корня

имя	**тип**	**предназначение**
log	int	Указывает на отсутствие ошибок (значение true) ввода данных.

5. Функция InputData вводит необходимые данные - количество студентов и данные об этих студентах; возвращает true, если не было ошибки ввода, иначе возвращается false. Функция InputData использует функции GetStudent и GetCount, поэтому они должны быть глобальными по отношению к InputData. Должен быть доступен тип “student”.

int InputData(long *studCount,student *students);

Сначала запрашивается количество студентов с помощью GetCount. Если не возникло ошибки, то в цикде с предусловием запрашивается указанное количество студентов через GetStudent, пока не будет введено нужное количество студентов или не возникнет ошибки. Если ошибок не было, возвращается true, иначе - false.

Используемые функцией параметры-переменные приведены в таблице 13; локальные переменные - в таблице 14.

Таблица 13 - Параметры-переменные функции ввода данных для поиска одногруппников

имя	**тип**	**предназначение**
studCount	long*	Количество студентов в списке.
students	student*	Данные о студентах.

Таблица 14 - Локальные переменные функции ввода данных для поиска одногруппников

имя	**тип**	**предназначение**
i	long	Переменная-счетчик для обработки массива.
log	int	Указывает на отсутствие ошибок (значение true) ввода данных.

6. Функция CheckData позволяет организовать проверку пользователем введенных им ранее значений. Возвращает true, если пользователь подтвердил правильность данных. Должен быть доступен тип “student”. Функция использует GetStudent , поэтому она должны быть глобальными по отношению к CheckData.

int CheckData(long studCount,student stud[])

Функция сначала спрашивает пользователя, желает ли он продолжить работу без проверки, проверить данные, или завершить программу. Если он ответил '0', подпрограмма завершается, если он ответил '1', то перебираются в цикле с

постусловием студенты из массива stud, где выводятся данные о текущем студенте и предлагаются варианты: считать данные правильными, завершить программу или изменить данные. Если пользователь ответил '1', то данные вводятся заново с помощью функции GetStudent. Это продолжается, пока студенты не закончатся, либо пользователь не ответит '0', означающее завершение работы. В конце возвращается true, если пользователь не отвечал '0', иначе возвращается false.

Используемые функцией параметры-константы приведены в таблице 15; локальные переменные - в таблице 16.

Таблица 15 - Параметры-константы функции проверки пользователем введенных значений

имя	**тип**	**предназначение**
studCount	long	Количество студентов в списке.
stud	student*	Данные о студентах.

Таблица 16 - Локальные переменные функции проверки пользователем введенных значений

имя	**тип**	**предназначение**
answer	char	Содержит ответ пользователя на запрос программы о правильности данных.
i	long	Переменная-счетчик для обработки массива.

7. Процедура FindStudents находит одногруппников в переданном списке студентов. Должен быть доступен тип “student”. Для работы подпрограммы необходима константа nn.

```
void FindStudents(long inStudCount,
                  student inStud[],
                  long *outStudCount,
                  student outStud[]);
```

Сначала все элементы массива outed устанавливаются в false, такое же значение устанавливается и для founded. k устанавливается в 0. Затем в цикле для i от 1 до InStudCount-1 проверяется, не “выбыл” ли этот студент - не был ли он уже найден как одногрупник, т. е. не равно ли outed[i] true. Если нет то, то проходим j от i+1 до inStudCount, опять-таки проверяем не равно ли outed[j] true, и если нет, то если имя вуза и номер группы студентов i и j, то устанавливается founded в true, увеличивается на 1 k, inStud[j] сохраняется в outStud[k]. После завершения цикла

j, если founded равно true, то founded сбрасывается в false, k инкрементируется и inStud[i] сохраняется в outStud[k]. После окончания цикла i k созраняется в outStudCount.

Используемые процедурой параметры-константы приведены в таблице 17, параметры-переменные - в таблице 18, локальные переменные - в таблице 19 .

Таблица 17 - Параметры-константы процедуры поиска одногруппников

имя	**тип**	**предназначение**
inStudCount	long	Количество студентов в общем списке.
inStud	student*	Данные о студентах общего списка.

Таблица 18 - Параметры-переменные процедуры поиска одногруппников

имя	**тип**	**предназначение**
outStudCount	long*	Количество найденных одногруппников.
outStud	student*	Данные о найденных одногруппниках.

Таблица 19 - Локальные переменные процедуры поиска одногруппников

имя	**тип**	**предназначение**
k	long	Временная переменная, для хранения количества найденных одногруппников.
i,j	long	Переменные-счетчики для обработки массива.
founded	int	Указывает на то, что одногруппник был найден.
outed	int[MAX_N]	Указывает, был ли уже студент отмечен как одногруппник.

8. Процедура PrintData выводит список одногруппников на экран. Должен быть доступен тип “student”.

void PrintData(long studCount,student stud[]);

Сначала процедура выводит все данные о первом студенте, потом в цикле перебираются все студенты, и если его вуз и группы отличны от предыдушего, то они выводятся на экран, иначе выводятся только фамилия, имя и отчество студента.

Используемые процедурой параметры-константы приведены в таблице 20, локальные переменные - в таблице 21 .

Таблица 20 - Параметры-константы процедуры вывода списка одногруппников

имя	тип	предназначение
StudCount	long	Количество найденных одногруппников.
Stud	student*	Данные о найденных одногруппниках.

Таблица 21 - Локальные переменные процедуры вывода списка одногруппников

имя	тип	предназначение
i	long	Переменная-счетчик для обработки массива.

9. Функция clearline используется для очистки строки файла.

Заголовок функции:

int clearline(FILE *f);

Функция считывает до конца файла или строки файла f символы в цикле while и возвращает их количество.

Параметры функции представлены в таблице 22, локальные переменные — в таблице 23.

Таблица 22 - Параметры функции очистки строки

имя	тип	предназначение
f	FILE*	Файл, в котором очищается строка.

Таблица 23 - Локальные переменные функции очистки строки

имя	тип	предназначение
count	long	Счетчик считанных символов.

10. Функция GetReqResult используется для получения ответов пользователя на запросы программы.

Заголовок функции:

int GetReqResult();

Функция запрашивает у пользователя символы 'y','Y','n' или 'N' до тех пор, пока он не введет какой-либо из них. Соответственно возвращается либо символ 'Y', либо 'N'.

Локальные переменные функции представлены в таблице 24.

Таблица 24 - Локальные переменные функции получения ответа от пользователя

имя	тип	предназначение
answer	char	Ответ пользователя.

11. Функция GetOption позволяет выбирать пользователю из нескольких (до 10) вариантов ответа.

Заголовок функции:

int GetOption(char a,char b);

Функция запрашивает у пользователя символы из промежутка от a до b до тех пор, пока он не введет какой-либо из них. Соответственно возвращается значение введенного символа.

Параметры функции представлены в таблице 25, локальные переменные — в таблице 26.

Таблица 25 - Параметры функции предоставления различных вариантов ответа

имя	**тип**	**предназначение**
a,b	char	Различные варианты представлены цифрами от a до b.

Таблица 26 - Локальные переменные функции предоставления различных вариантов ответа

имя	**тип**	**предназначение**
ch	char	Ответ пользователя.
ok	int	Флаг — равен 1, если ответ пользователя допустим, иначе равен 0.

Текст программы

Ниже представлен текст программы, написанной на языке Turbo Pascal 7, которая находит среди данных студентов одногруппников.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define MAX_N 10L
typedef struct{
    char FstName[20],LstName[20],SurName[20],University[50];
    long GroupNum;
} student;
int clearline(FILE *f){
  int count=0;char ch;
      while (!feof(f)&&((ch=getc(f))!='\n'))
        if(ch!=' '&&ch!='\t')
            count++;
      return count;
}
char GetOption(char a,char b){
  char ch; int ok;
    do{
        printf("Введите цифру от %c до %c.\n",a,b);
            ch=getchar();
            ok=(a<=ch)&&(ch<=b)&&!clearline(stdin);
            if (!ok)
            printf("Неправильный ответ!\n");
    }while (!ok);
      return ch;
}

int GetReqResult(){
  char answer;
    answer=getchar();
    clearline(stdin);
    while (toupper(answer)!='Y'
           &&toupper(answer)!='N'){
        printf("Неправильный ответ! Допустимо:\n"\
                                "Y - да; N - нет.\n");
        answer=getchar();
        clearline(stdin);
    }
```

```
    return toupper(answer)=='Y';
}
int GetString(const char message[],const char name[],char paramVar[],long size){
  int log,req_rslt; char format[15];
    sprintf(format,"%%%ld[^\n]",size);
    do{
        req_rslt=0;
        printf("%s",message);
        printf("%s: ",name);
        log=scanf(format,paramVar);
        log=!clearline(stdin)&&log;
        if (!log||!strcmp(paramVar,"")){
            log=0;
            printf("Ошибка! Повторить ввод?(Y/N)\n");
            req_rslt=GetReqResult();
        }
    } while(!log&&req_rslt);
    return log;
}
int GetGroup(long *paramVar){
  int log,req_rslt;
    do{
        req_rslt=0;
        printf("Введите номер группы.\n");
        printf("Номер группы: ");
        log=scanf("%ld",paramVar);
        log=!clearline(stdin)&&log;
        if (!log){  //введено не вещественное число?
            printf("Введено не число!\n"
                   "Повторить ввод? Y/N\n");
            req_rslt=GetReqResult();
        }
        if (log&&*paramVar<1){
            log=0;
            printf("Номер группы<=0! Повторить ввод?(Y/N)\n");
            req_rslt=GetReqResult();
        }
    } while(!log&&req_rslt);
    return log;
}
int GetCount(long *paramVar){
  int log,req_rslt;
    do{
        req_rslt=0;
```

```
        printf("Введите n - общее количество студентов.(1<=n<=%ld)\n",MAX_N);
        printf("Студентов: ");
        log=scanf("%ld",paramVar);
        log=!clearline(stdin)&&log;
        if (!log){  //введено не вещественное число?
            printf("Введено не число!\n"
                   "Повторить ввод? Y/N\n");
            req_rslt=GetReqResult();
        }
        if (log&&(*paramVar<1||*paramVar>MAX_N)){
            log=0;
            printf("Количество студентов неправильно! Повторить ввод?(Y/N)\n");
            req_rslt=GetReqResult();
        }
    } while(!log&&req_rslt);
    return log;
}
int GetStudent(student *stud){
  int log;
    log=GetString("Введите фамилию студента.\n","Фамилия",stud->LstName,20);
    if (log)
        log=GetString("Введите имя студента.\n","Имя",stud->FstName,20);
    if (log)
        log=GetString("Введите отчество студента.\n","Отчество",stud->SurName,20);
    if (log)
        log=GetString("Введите название университета.\n","Университет",stud-
>University,50);
    if (log)
        log=GetGroup(&(stud->GroupNum));
    return log;
}

int InputData(long *studCount,student *students){
  long i; int log;
    log=GetCount(studCount);
    for (i=0;log&&i<*studCount;i++){
        printf("****************************\n");
        printf("Ввод данных о студенте №%ld\n",i+1);
        printf("****************************\n");
        log=GetStudent(students+i);
    }
    return log;
}
```

```
int CheckData(long studCount,student stud[]){
  long i; char answer;
    printf("1 - проверить данные о каждом студенте по порядку;\n");
    printf("2 - завершить работу;\n");
    printf("0 - продолжить работу.\n");
    answer=GetOption('0','2');
    if (answer=='1')
        for (i=0;i<studCount&&answer!='2';i++){
            printf("****************************");
            printf("Карточка студента №%ld",i*1);
            printf("****************************");
            printf("***Университет: %s Группа:
%ld\n",stud[i].University,stud[i].GroupNum);
            printf("%s %s %s\n",stud[i].LstName, stud[i].FstName,
stud[i].SurName);
            printf("Правильно?\n"
                   "0 - Да;\n"
                   "1 - Нет, изменить данные;\n"
                   "2 - Нет, завершить программу.\n");
            answer=GetOption('0','2');
            if (answer=='1'){
                printf("****************************");
                printf("Ввод данных о студенте №%ld",i+1);
                printf("****************************");
                GetStudent(&stud[i]);
            }
        }

    return answer!='2';
}
void FindStudents(long inStudCount,
                  student inStud[],
                  long *outStudCount,
                  student outStud[]){
  long k=0,i,j; int founded;int outed[MAX_N];
    for (i=0;i<MAX_N;i++)
        outed[i]=0;
    founded=0;

    for (i=0;i<inStudCount-1;i++){
        if (!outed[i]){
            for (j=i+1;j<inStudCount;j++)
                if (!outed[j]
                   &&(inStud[i].GroupNum==inStud[j].GroupNum)
```

```
                    &&!strcmp(inStud[i].University,inStud[j].University)){
                     founded=1;
                     outStud[k++]=inStud[j];
                     outed[j]=1;
                }
            if (founded) {
                outStud[k++]=inStud[i];
                founded=0;
            }

        }
    }


    *outStudCount=k;
}

void PrintData(long studCount,student stud[]){
  long i=0;
    printf("Найденные одногрупники:");
    printf("\n***Университет: %s Группа:
%ld\n",stud[i].University,stud[i].GroupNum);
    printf("%s %s %s\n",stud[i].LstName, stud[i].FstName, stud[i].SurName);
    for (i=1;i<studCount;i++){
        if ((stud[i].GroupNum!=stud[i-1].GroupNum)
           ||strcmp(stud[i].University,stud[i-1].University))
              printf("\n***Университет: %s Группа:
%ld\n",stud[i].University,stud[i].GroupNum);
        printf("%s %s %s\n",stud[i].LstName, stud[i].FstName, stud[i].SurName);
    }


}

typedef struct{
    long Count;
    student Stud[MAX_N];
} students;

int main(){
  students AllStudents, FoundedStudents;
  int log;
    printf("Программа находит одногрупников среди данных студентов.\n");
    log=InputData(&(AllStudents.Count),AllStudents.Stud);
```

```
    if (log)
        log=CheckData(AllStudents.Count,AllStudents.Stud);
    if (log){
        FindStudents(AllStudents.Count,AllStudents.Stud,&(FoundedStudents.Count),F
oundedStudents.Stud);
        if (FoundedStudents.Count)
            PrintData(FoundedStudents.Count,FoundedStudents.Stud);
        else
            printf("Одногрупников не найдено...");
    } else
        printf("Программа завершена пользователем...");

    printf("Нажмите <Enter>...");
    clearline(stdin);
    return 0;
}
```

Тестовый пример

Ниже на рисунках 12 и 13 представлен пример работы программы для списка из 5 студентов, которые в сумме обучаются в 2-х вузах, и двое из списка — одногруппники.

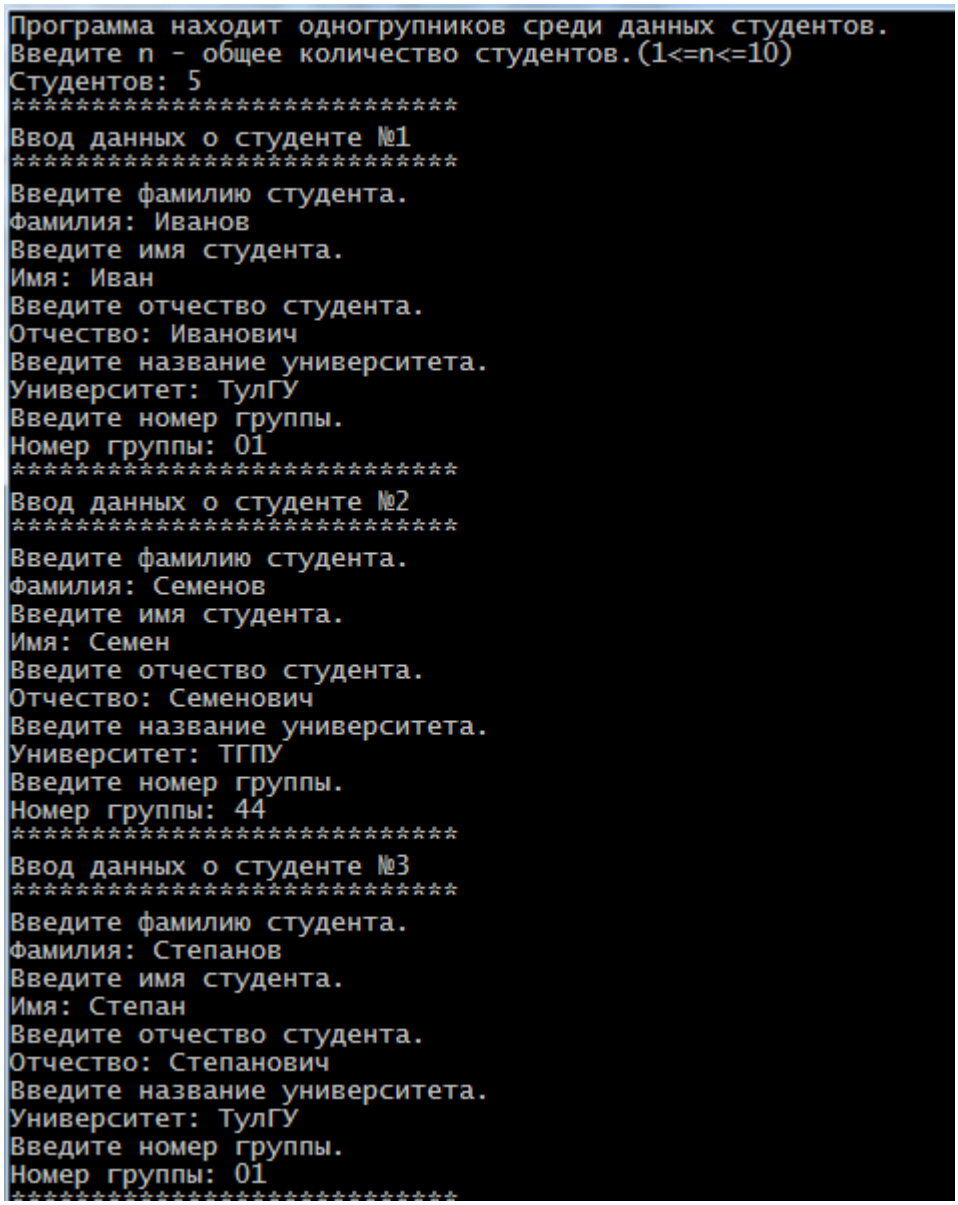

Рисунок 12 — Пример работы программы поиска одногруппников(начало)

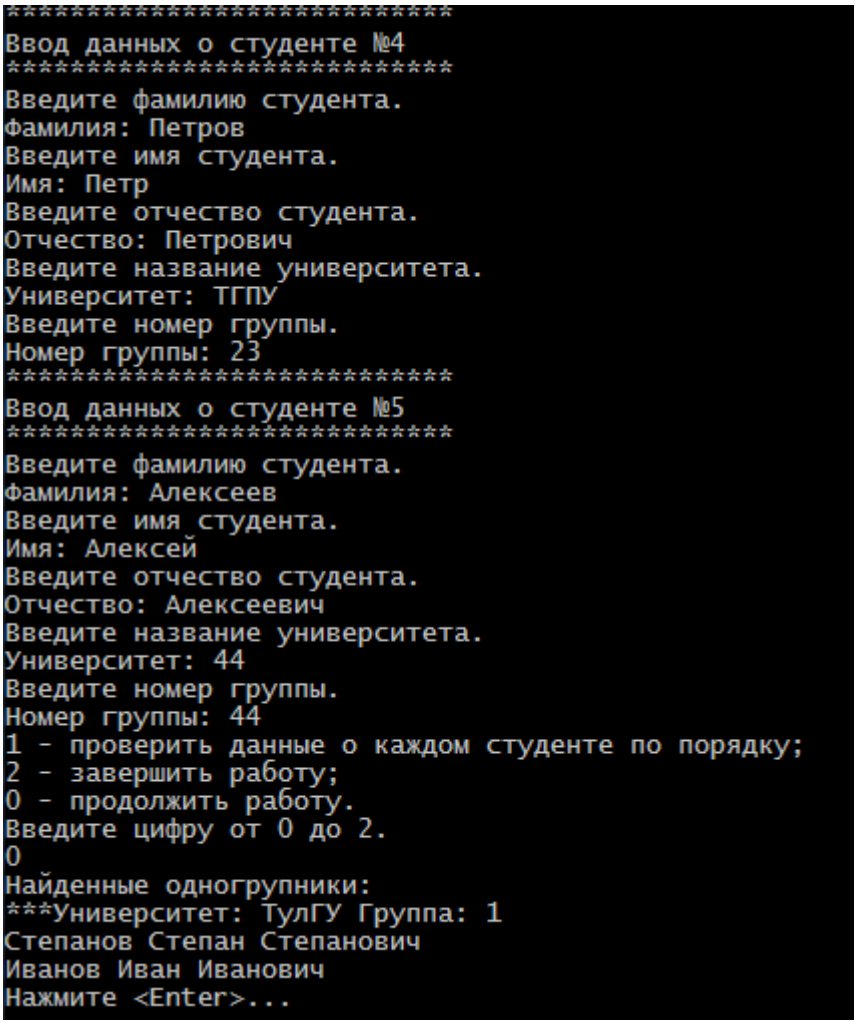

Рисунок 13 — Пример работы программы поиска одногруппников(продолжение)

Вывод

В ходе выполнения данной лабораторной работы я научился использовать структуры при написании программ на Си. Структуры помогают скомпоновать разнородные данные, что помогает при построении моделей объектов реального мира; кроме того т. н. пространства имен позволяют обеспечить более высокую структуризацию программы. Структуры очень часто используются при моделировании объектов реального мира.