МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра биомедицинской инженерии

Утверждаю
Проректор по учебной работе
О.Г. Локтионова
«25» 09 2023 г.

ЯЗЫК СИ

Методические рекомендации по выполнению лабораторных работ для студентов направления подготовки 12.03.04 – «Биотехнические системы и технологии» (бакалавр)

Курск 2023

УДК 621.(076.1)

Составители: А.А.Кузьмин

Рецензент:
Кандидат технических наук, доцент *Т.Н. Конаныхина*

Язык Си: методические рекомендации по выполнению лабораторных работ для студентов направления подготовки 12.03.04 – «Биотехнические системы и технологии» (бакалавр) / Юго-Зап. гос. ун-т; сост.: А.А.Кузьмин. - Курск, 2023. - 24 с.

Содержат методические рекомендации к проведению лабораторных работ по дисциплине «Язык Си». Методические указания по структуре, содержанию и стилю изложения материала соответствуют методическим и научным требования, предъявляемым к учебным и методическим пособиям.

Предназначены для студентов направления подготовки 12.03.04 – «Биотехнические системы и технологии» (бакалавр)

Текст печатается в авторской редакции

Подписано в печать 25.03.03 Формат 60x84 1/16
Усо.печ.л. 1,4 . Уч.-изд.л. 1,3 .Тираж 30 экз. Заказ: 1073 .Бесплатно.
Юго-Западный государственный университет.
305040. г. Курск, ул. 50 лет Октября, 94.

Лабораторная работа№1
Программирование разветвляющихся вычислительных процессов

1. Цель работы:

приобрести навыки программирования, отладки и тестирования разветвляющихся вычислительных процессов.

2. Условия:

даны действительные числа x,y,z. Найти максимальное значение.

3. Листинг программы:

```
#include <conio.h>
#include <iostream.h>
int main()
  {
   clrscr();
       int x,y,z,max;
       int *p=&x;
       cout<<"Введите число X: ";
       cin>>x;
       cout<<"Введите число Y: ";
       cin>>y;
       cout<<"Введите число Z: ";
       cin>>z;
       max=*p--;
   for(int i=0;i<2;i++)
       {
        if(max<*p)
          {
            max=*p;
            p--;
          }
        else p--;
       }
```

```
    cout<<"Максимальное число: "<<max;
  getch();
  return 0;
 }
```

4. Комментарии к тексту программы

Ключевым элементом данной модели языка C++ выступает функция.

Функцию можно представить как подпрограмму или некую процедуру, несущую законченную смысловую нагрузку. Любая программа на C++ обязательно включает в себя функцию main(), с которой и начинает своё выполнение. Функция main() может возвращать значение, в этом случае указывается тип возвращаемого значения, например:

int main()

Если функция не возвращает значение, то указывается:

void main()

В начальной части программы указываются подключаемые для выполнения программы библиотеки функций.

Поскольку используются функции clrscr () и getch (), то необходима библиотека conio.h (ввод-вывод с консоли; ввод с клавиатуры; вывод на экран).

Организация потокового ввода-вывода требует подключения библиотеки iostream.h.

Директива препроцессора #include подключает файл conio.h и iostream.h. Имя подключаемого модуля указывается в косых скобках (<>- файл находится в каталоге \INCLUDE), либо в кавычках (” ”- файл находится в текущем каталоге).

Тело функции заключается в фигурные скобки

```
int main()
{
…..
}
```

В функции main() объявлены локальные переменные x,y,z целого типа и указатель p на целый тип данных. При объявлении указатель p инициируется адресом переменной x (т.е. в указателе p будет содержаться адрес ячейки памяти, в которой находится значение x). Для того, чтобы узнать адрес конкретной переменной, используется унарная операция взятия адреса. При этом перед именем переменной ставится знак амперсанта &.

Оператор cout инициирует вывод в поток. Это реализуется с помощью оператора вставки, которым является перегруженный оператор сдвига влево <<. Правым его оператором может являться любая переменная, в том числе текст, заключённый в кавычки.

Можно строить цепочки вызовов оператора <<, которые выполняются слева направо:

```
cout<<”i=”<<i<<” ,d=”<<d<<’\n’;
```

Эта конструкция приводит к выводу на экран следующей строки (например, при i=5 и d=2.08):

```
i=5,d=2.08
```

Оператор cin инициирует ввод из потока.

Для ввода информации из потока используется оператор извлечения, которым является перегруженный оператор сдвига вправо >>. Это позволяет строить цепочки инструкций извлечения из потока, выполняемых слева направо.

Переменная max приобретает значение, находящееся по адресу, содержащемуся в указателе p . Для того, чтобы получить (прочитать) значение, записанное в некоторой области, на которую ссылается указатель, используют операцию косвенного обращения или разыменовывания * . При этом используется имя указателя со звёздочкой перед ним:

```
max=*p;
```

В данном примере указатель p инициирован адресом переменной x, поэтому max будет иметь значение, равное x.

for(int i=0;i<2;i++) – заголовок цикла. Тело цикла заключается в фигурные скобки:

```
for(int i=0;i<2;i++)
{
...
}
```

В заголовке цикла через ; перечисляется значение параметра цикла и его тип, условие выхода из цикла, приращение счётчика цикла i (++ операция инкремента (увеличения на +1)).

Поиск максимального значения реализуется путём попарного сравнения. Оператор p-- означает, что адрес, содержащийся в указателе p, модифицируется в соответствии с заданной арифметической операцией (в данном случае декрементируется с учётом принятого для данного типа размера памяти, отводимого на один элемент). Причём операция декрементирования выполняется после использования адреса указателя p. Это постфиксная форма записи. Если бы запись была --p, то адрес вначале декрементировался, а затем использовался (префиксная запись).

Тип int занимает в памяти 2 байта. Следовательно, адрес декрементируется на 2. Если длина размещения переменной будет 6 байтов, то адрес будет декрементироваться на 6.

Таблица 1. Типы данных и их размерность

тип	размер в байтах	значение
Unsigned short int	2	От 0 до 65 535
Short int	2	От –32 768 до 32 767
Unsigned long int	4	От 0 до 4 294 967 295
Long int	4	От –2 147 483 648 до 2 147 483 647
Int (16 разрядов)	2	От –32 768 до 32 767
Int (32 разряда)	4	От –2 147 483 648 до 2 147 483 647
Unsigned int (16 разрядов)	2	От 0 до 65 535
Unsigned int (32 разряда)	4	От 0 до 4 294 967 295
Char	1	От 0 до 256
Float	4	От $1{,}2*10^{-38}$ до $3{,}4*10^{38}$
Double	8	От $2{,}2*10^{-308}$ до $1{,}8*10^{308}$
Void	2 или 4	-----

Также необходимо учитывать, что локальные переменные x,y,z размещаются в стеке и следуют в обратном порядке:

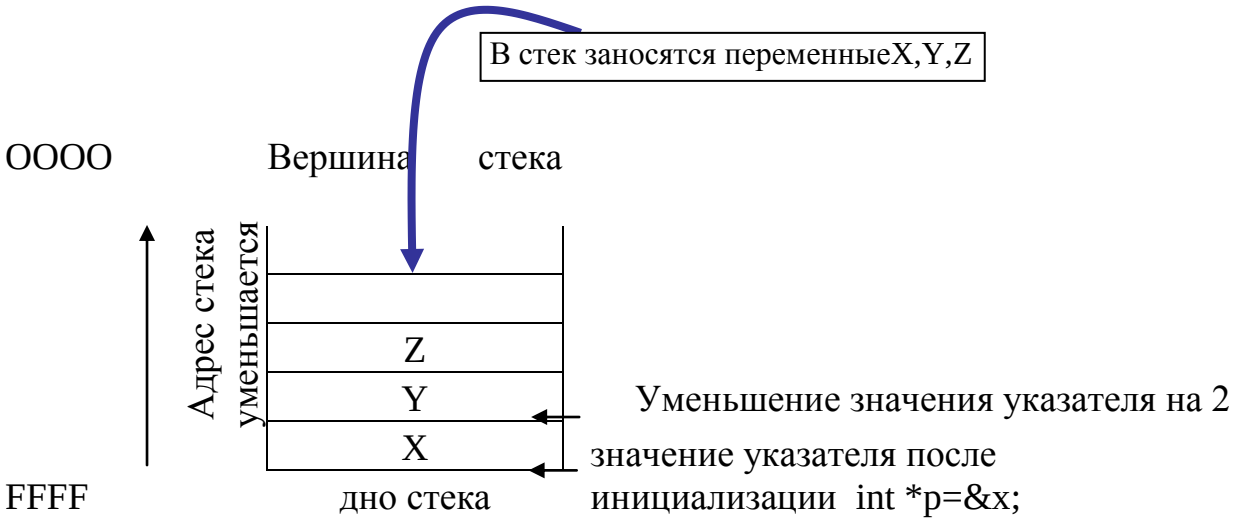


Рис. 1. Механизм работы стека

Принцип работы стека:

Последний пришёл – первый ушёл LIFO.

Размерность одной ячейки стека – 2 байта.

С учётом этого и необходимо было записать: p--.

После определения максимального значения max, оно выводится на экран.

Функция getch () – считывание символа с клавиатуры без эхопечати. Т.к. в скобках нет переменной, то этот приём можно использовать для приостановки работы программы, пока не будет нажата любая клавиша.

Завершается функция main оператором return 0, возвращающем управление вызывающей программе.

Можно реализовать указанные действия привычными по форме операторами:

вместо

```
max=*p;
```

можно записать

```
max=x;
```
вместо цикла

```
for(int i=0;i<2;i++)
        {
         if(max<*p)
           {
             max=*p;
             p--;
           }
         else p--;
        }
```
можно записать:
```
max=x;
if (max<y)  max=y;
if (max<z)  max=z;
```

Очевидно, что в объявлении указателя p нет необходимости.

В языке C++ есть единственный трёхнарный условный оператор ?. Его вполне можно использовать вместо операторов if-else, если входящие в него выражения достаточно просты. В нашем примере после ввода данных следует записать:

```
max=x;
max=(y>max) ? y : max;
max=(z>max) ? z : max;
```

Конструкция оператора ? имеет формат:

условие ? выражение1 : выражение2;

По аналогии с оператором if данный условный оператор работает так: если условие приняло истинное значение, выполняется выражение1, а если ложное – выражение2. Обычно возвращаемое значение присваивается какой-либо переменной. В данном случае,

если max>y, то переменной max будет присвоено значение y, иначе переменной max будет присвоено значение max.

5. Содержание отчёта:
   - задание;
   - лист программы с комментариями;
   - контрольный пример.

6. Контрольные вопросы

1. В чём функциональное назначение первых двух строк текста вышеприведённой программы ?
2. Какую функцию обязательно включает в себя любая программа на C++?
3. Каким образом включается комментарий в текст программы ?
4. Что понимается под идентификатором в C++ ?
5. Приведите примеры ключевых слов языка C++.
6. Что понимается под переменной в языке C++ ?
7. Каков формат объявления переменной?
8. Равнозначными ли являются переменные ABC и abc в языке C++?
9. Каков формат инициализации переменной при её объявлении ?
10. Какой тип имеют целочисленные переменные ?
11. Какой тип имеют логические переменные ?
12. Какой тип имеют символьные переменные?
13. Какой тип имеют числа с плавающей запятой?
14. Что понимается под константой в языке C++?
15. Приведите примеры символьных, строковых, целых, вещественных констант.
16. Что понимается под типизированной константой?

Лабораторная работа№2

Программирование итерационных вычислительных процессов

1. Цель работы:

приобрести навыки программирования, отладки и тестирования итерационных вычислительных процессов.

2. Условия:

вычислить сумму с точностью E: $\sum_{i=1}^{\infty} \frac{(-2)^i}{i!}$

3. Листинг программы:

```
#include <math.h>    //необходим для ф-ии fabs()- получение модуля
#include <conio.h>    //вещественного числа
#include <iostream.h>
int main()
  {
   clrscr();
      double chislitel=1,znamenatel=1,e,summa=0;
      int i=1;
//вывод условия задания на экран
      cout<<"\t\t\t\t\t\t (-2)^i "<<'\n';
      cout<<"Программа вычисляет  с точностью E сумму:  -------"<<'\n';
      cout<<"\t\t\t\t\t     i! "<<'\n';
      cout<<"Введите точность вычисления: ";
      cin>>e;
    while(1)   //бесконечный цикл
     {
      znamenatel*=i++;
      chislitel*=-2;
```

```
    if(fabs(chislitel/znamenatel)<e)   break;
    summa+=chislitel/znamenatel;
  }
    cout<<"Сумма равна: ";
    cout<<summa;

 getch();
 return 0;
}
```

4. Комментарии к тексту программы.

Данный листинг иллюстрирует программирование итерационных вычислительных процессов. В заголовочной части программы #include указываются подключаемые библиотеки функций. Ряд из них описаны в лабораторной работе №1. Поскольку используется функция fabs(), необходимо подключение библиотеки math.h (математические функции).

Объявлены переменные двойной точности.

В операторе

cin>>e;

вводится требуемое значение точности e вычисления суммы. При вычислении значения числителя и знаменателя заданной формулы используется формат оператора присваивания с соответствующей операцией.

Выражение

znamenatel*=i++;

можно записать и так:

znamenatel = znamenatel*i++;

от.е. значение переменной znamenatel умножается на значение переменной i и конечный результат присваивается переменной znamenatel. После выполнения операции умножения на i , значение i инкрементируется на 1 (постфиксная запись).

Аналогично следует рассматривать выражение для переменных chislitel и summa . Например, для суммы привычная форма записи:

Summa = summa+ chislitel/znamenatel;

При достижении необходимой точности (проверка в операторе if) реализуется выход из цикла while() по оператору break . Оператор break может встречаться в теле цикла сколько угодно раз и передаёт управление вне тела конструкции.

При работе с переменными можно использовать и указатели. Однако, следует отметить, что текст программы окажется несколько усложнённым. Ниже приведём текст функции int main() при использовании указателей:

```
int main()
  {
   clrscr();
      double e,summa=0,znamenatel=1,chislitel=1;
      int i=1;
            double
*p_e=&e,*p_summa=&summa,*p_znamenatel=&znamenatel,
*p_chislitel=&chislitel;
      cout<<"\t\t\t\t\t\t  (-2)^i "<<'\n';
      cout<<"Программа вычисляет  с точностью E сумму:  -------
"<<'\n';
      cout<<"\t\t\t\t\t     i! "<<'\n';
      cout<<"Введите точность вычисления: ";
      cin>>e;
    while(1)   //бесконечный цикл
     {
      *p_znamenatel*=i++;
      *p_chislitel*=-2;
      if(fabs(*p_chislitel/(*p_znamenatel))<e)     break;
      *p_summa+=*p_chislitel/(*p_znamenatel);
     }
      cout<<"Сумма равна: ";
      cout<<*p_summa;

    getch();
    return 0;
    }
```

Указатели инициируются адресом соответствующей переменной. В дальнейшем, по сути дела, работаем со значениями переменных, используя операцию разыменовывания (смотри лабораторную работу №1). Структура программы при использовании указателей остаётся прежней.

5. Содержание отчёта:
   - задание;
   - листинг программы с комментариями;
   - контрольный пример;

6. Контрольные вопросы

1. Что понимается под операцией в языке С++?
2. Что представляет собой выражение в С++?
3. В чём функциональное назначение унарных операторов инкремента (++) и декремента (--)?
4. Какова конструкция пустого оператора?
5. Поясните применение префиксного оператора к операнду.
6. Поясните применение постфиксного оператора к операнду.
7. Приведите пример оператора сложения с присваиванием.
8. Приведите пример оператора умножения с присваиванием.
9. Что вычисляет арифметическая операция %?
10. Каким образом определяется порядок вычисления математических выражений?

Лабораторная работа№3

Программирование операций с элементами массивов

1. Цель работы:

приобрести навыки программирования, отладки и тестирования операций с элементами массивов.

2. Условия:

Дана последовательность A1...An. Определить в этой последовательности число соседних двух чисел одного знака, причём модуль первого числа должен быть больше модуля второго числа.

3. Листинг программы

```
#include <math.h>       //необходим для ф-ии ABS
#include <conio.h>
#include <iostream.h>
int main()
  {
   clrscr();
      int array[10];            //10 элементов массива
      for(int i=0;i<10;i++)
        {
          cout<<"Введите ";
          cout<<i+1;
          cout<<"й элемент последовательности: ";
          cin>>array[i];
        }
      int n=0;          //количество соседних чисел,которые надо
найти
```

```
            int *p_array=array;
            for(i=0;i<9;i++)
               if (((*p_array<0 && *(p_array+1)<0) || (*p_array>=0 &&
*(p_array+1)>=0)) && abs(*p_array)>abs(*(p_array+1)))
                 {
                   p_array++;
                   n++;
                 }
               else p_array++;

            cout<<"Количество таких чисел равно: ";
            cout<<n<<'\n';

          getch();                //ждём нажатия клавы
          return 0;
         }
```

4. Комментарии к тексту программы

Директива препроцессора #include <math.h> указывает на необходимость подключения библиотеки математических функций (используется функция abs).

Объявлен массив array целого типа int из 10 элементов.

В цикле for(int i=0;i<10;i++) организуется ввод элементов массива. Элементы массива индексируются от 0. Поэтому параметру цикла i присваивается начальное значение, равное 0, а условие выхода из цикла i<10.

Следует отметить, что три выражения не обязательно должны присутствовать в конструкции for , однако синтаксис не допускает пропуска символа ; . Если в цикле должны синхронно изменяться несколько переменных, которые зависят от переменной цикла, вычисление этих значений можно поместить в оператор for , воспользовавшись оператором «запятая».

Например:

```
for(int i=10, j=2;i<20;i++,j=i+17)
{
…
```

}

Как следует из текста программы, объявление переменных не обязательно находится в начале текста программы, а может встречаться в любой точке программы. Но необходимо учитывать, что область действия переменной: от точки объявления до конца файла. Поэтому массив array объявлен до цикла ввода его элементов, а переменная n –после цикла ввода, т.к. выше она не используется.

Переменная p_array – указатель на массив array. Он инициируется начальным адресом array. Имя объявляемого массива ассоциируется компилятором с адресом его самого первого элемента (с индексом 0). Таким образом, можно присвоить указателю адрес нулевого элемента, используя имя массива.

В условном операторе if реализуется проверка заданного условия задачи. && - логическая функция «И», || - логическая функция «ИЛИ» (не путать с поразрядными «И» и «ИЛИ»). Используется запись if с ключевым словом else. Если используется if с ключевым словом else, то перед else ставится символ «точка с запятой (;)». При заключении последовательности операторов в фигурные скобки ({…}) символ «точка с запятой (;)» после фигурной скобки перед else не ставиться. Оператор сравнения в if имеет вид “= =”, а не ‘’=’’.

Конструкция *p_array определяет значение по адресу, находящемуся в указателе.

Конструкция p_array++ означает, что над указателем выполняется арифметическая операция инкрементирования. То есть адрес, хранящийся в указателе будет увеличен на единицу длины размещения переменной типа int в памяти (на 2).

Если записать *p_array+1, то в соответствии с приоритетом операций вначале будет выполнена операция разыменовывания, т.е. будет прочитано значение, находящееся по адресу в указателе p_array, а затем это значение будет увеличено на 1.

Оператор n++ просто инкрементирует значение переменной на 1. Можно записать и так:

n+=1; привычная запись n=n+1;

\n – символ escape последовательности – перевод строки.

Другие символы, полезные при программировании:

\’ – вывод апострофа,
\” – вывод кавычки,
\a – подача звукового сигнала,
\f – перевод страницы,
\r – возврат курсора на начало текущей строки,
\t – перевод курсора к следующей позиции табуляции.

Оператор for обработки элементов массива можно записать привычным (как в Pascal) способом:

```
for(i=0;i<9;i++)
   if (((array[i]<0 && array[i+1]<0) ||
(array[i]>=0 && array[i+1]>=0)) &&
abs(array[i])>abs(array[i+1]))
      n++;
```

При этом объявлять указатель p_array нет необходимости.

5. Содержание отчёта:
   - задание;
   - лист программы с комментариями;
   - контрольный пример.

6. Контрольные вопросы

1. В чём функциональное назначение и формат оператора if ?
2. Каким образом определяется блок исполняемых операторов ?
3. Конструкция и исполнение условного оператора.
4. Функциональное назначение, формат и выполнение оператора switch.
5. Функциональное назначение, формат и выполнение оператора цикла for.
6. Функциональное назначение, формат и выполнение оператора цикла while.
7. Функциональное назначение, формат и выполнение оператора цикла do-while.
8. В чём отличие в действии операторов break и continue?
9. Использование оператора goto и меток в языке C++.

10. Определите понятие массива.
11. Что необходимо указать при объявлении массива?
12. Какими способами можно обращаться к элементам массива?
13. Какой индекс имеет первый элемент массива?
14. Чем инициируется в программе указатель p_array?
15. Каким образом реализуется инициализация массивов?
16. Особенности инициализации многомерных массивов.

Лабораторная работа№4

Программирование строковых операций

1. Цель работы:

приобрести навыки программирования, отладки и тестирования строковых операций.

2. Условия:

дана строка символов, состоящая из слов. Найти длину самого короткого слова.

3. Листинг программы

```
#include <conio.h>
#include <iostream.h>
int main()
  {
   clrscr();
       char string[80];          //строка символов
       int i=0,dlina=80;              //длина слова
       char *min_word,*begin_word,*end_word;
       cout<<"Введите строку символов:"<<'\n';
       cin.getline(string,sizeof(string));
//поиск самого маленького слова
       begin_word=string;
       end_word=string;
       begin_word--;
       while(*end_word!='\0')
         {
          if(*end_word==' ' && *(end_word+1)!=' ')
            {
```

```
                    begin_word=end_word;
                    end_word++;
                  }
                 else
                  {
                   if(*end_word!='  '  &&  (*(end_word+1)=='  '  ||
*(end_word+1)=='\0'))
                    {
                     i=end_word-begin_word;
                     if (dlina>i)
                       {
                        dlina=i;
                        min_word=begin_word;
                       }
                    }
                   end_word++;
                  }
                }
        //Вывод результатов
              if (dlina<80)
               {
                cout<<"Самое короткое слово: ";
                for(i=0;i<dlina;i++)
                    cout<<*(++min_word);
                cout<<'\n'<<"Оно состоит из "<<dlina<<" букв(ы)";
               }
              else          //были введены пробелы или ничего не введено
               cout<<"НАДО ВВОДИТЬ СЛОВА!!!";
            getch();                //ждём нажатия клавы
            return 0;
           }
```

4. Комментарии к тексту программы

Объявлена строка символов: char string[80];

Введены указатели char *min_word, *begin_word, *end_word соответственно для указания позиции в строке начала минимального и текущего слова, позиции конца слова, введённой строки.

Для ввода строки символов используется функция

cin.getline(string,sizeof(string));

Параметры функции определяют имя вводимой строки и количество символов строки string. Эта функция извлекает из входного потока символы в буфер string, пока не встретится символ-разграничитель (по умолчания перевод строки) или не будет прочитано (strlen()-1) символов, или не будет прочитан конец файла. При размещении строки в памяти к ней добавляется символ ‘\0’ – окончания строки.

Указатели begin_word и end_word инициализируются начальным адресом сроки.

Цикл while(*end_word!='\0') выполняется до тех пор, пока очередной символ не равен (!=) символу конца строки ‘\0’ . При выделении слова принимается во внимание, что слова разделены пробелом, т.е. проверяется условие:

if(*end_word==' ' && *(end_word+1)!=' ') , т.е. если очередной символ «пробел», а следующий символ «не пробел», то указатель begin_word получает адрес позиции пробела, предшествующего началу слова. Иначе, если же очередной символ не является «пробелом», а следующий либо «пробел» либо «конец строки»:

if(*end_word!=' ' && (*(end_word+1)==' ' || *(end_word+1)=='\0')), то определяется длина слова:

i=end_word-begin_word;

и сравнивается найденное значение с предыдущим:

```
if (dlina>i)
        {
          dlina=i;
          min_word=begin_word;
        }
```

Если значение i меньше, то запоминается новое значение (dlina=i) и фиксируется позиция пробела, предшествующего минимальному слову (min_word=begin_word). При выходе из цикла указатель min_word имеет значение позиции, предшествующей первому символу минимального слова.

Цикл for обеспечивает вывод на экран найденного слова.

Конструкция end_word++ (или (end_word+1), или (++min_word)) читается так: адрес, содержащийся в указателе, инкрементируется на 1, а затем на экран выводится символ, определённый указателем. Указатель инкрементируется именно на 1, т.к. указатель end_word (как впрочем, и другие указатели этой программы) ссылаются на тип данных char (символьный тип). Символьный тип занимает в памяти 1 байт.

Операции со строками можно выполнить и без использования указателей. Текст функции main() имеет вид:

```
int main()
  {
   clrscr();
       char string[80];          //строка символов
       int i=0,dlina=80,k;           //длина слова
       int min_word=0,begin_word=-1,end_word=0;
       cout<<"Введите строку символов:"<<'\n';
       cin.getline(string,sizeof(string));
//поиск самого маленького слова
       while(string[i]!='\0')
        {
         if(string[i]==' ' && string[i+1]!=' ')
          {
            begin_word=i;
            i++;
          }
         else
          {
           if(string[i]!=' ' && (string[i+1]==' ' || string[i+1]=='\0'))
             {
               k=i-begin_word;
               if (dlina>k)
                 {
                   dlina=k;
                   min_word=begin_word;
                 }
             }
```

```
            i++;
           }
         }
//Вывод результатов
       if (dlina<80)
        {
         cout<<"Самое короткое слово: ";
         for(i=0;i<=dlina;i++)
             cout<<string[++min_word];
         cout<<'\n'<<"Оно состоит из "<<dlina<<" букв(ы)";
        }
       else          //были введены пробелы или ничего не введено
        cout<<"НАДО ВВОДИТЬ СЛОВА!!!";
    getch();                //ждём нажатия клавы
    return 0;
   }
```

Как следует из текста, обращение к элементу строки реализуется по индексу i.

5. Содержание отчёта:
   - задание;
   - листинг программы с комментариями;
   - контрольный пример.

6. Контрольные вопросы
   1. Как представляются в C++ строки?
   2. Чем заканчивается ASCIIZ строка?
   3. Приведите пример объявления строки размером 10 символов.
   4. Объявлена строка: char buf [20]. Каков реальный размер строки?
   5. Какие два параметра имеет функция getline ()?
   6. Оператор ввода строки имеет вид: cin.getline(string,4)); Сколько значащих символов может иметь данная строка?
   7. Назначение и пример записи функции strcpy?
   8. Назначение и пример записи функции strcat?

9. Назначение и пример записи функции strchr?
10. Назначение и пример записи функции strcmp?
11. Назначение и пример записи функции strlen?
12. Поясните использование функции sizeof() в операторе ввода (см. текст программы): cin.getline(string,sizeof(string)); Какое значение при этом возвращает функция sizeof().

Лабораторная работа №5

Программирование операций со структурированными типами данных

1. Цель работы:

приобрести навыки программирования, отладки и тестирования операций со структурированным типом данных – структура.

2. Условия:

Организовать работу со структурой (номер дома, номер квартиры, количество жильцов). Обеспечить формирование файла с указанной структурой, добавление новой записи, просмотр, поиск по номеру дома и количеству жильцов.

3. Листинг программы

```
#include <io.h> //необходим для определения размера файла
#include <fcntl.h>              //необходим для ф-ии open()
#include <iostream.h>
#include <stdio.h>
#include <string.h>
#include <conio.h>

 struct Baza
   {
     char House[6];
     char Flat[3];
     unsigned int Kolichestvo;
   };
 char FileName[]="ips.dat";
 FILE* stream;
 Baza Zapis;
 int count;                  //элемент записи
 int MenuItem;                   //номер меню
```

```
        //создание файла заданной структуры
        Filecreate()
         {
          stream=fopen(FileName,"wb");        //создание файла для записи
          cout<<"\t\t\t\tВВЕДИТЕ КОЛИЧЕСТВО ЗАПИСЕЙ: ";
          cin>>count;
          for (int i=1; i<=count; i++)
           {
            clrscr();
             cout<<"\n\n\n\n\n\n\n\t\t\tЗАПИСЬ НОМЕР: "<<i<<"\n";
             cout<<"\t\t\tВВЕДИТЕ НОМЕР ДОМА:  ";
             cin>>Zapis.House;
             cout<<"\t\t\tВВЕДИТЕ НОМЕР КВАРТИРЫ: ";
             cin>>Zapis.Flat;
             cout<<"\t\t\tВВЕДИТЕ КОЛИЧЕСТВО ЖИЛЬЦОВ: ";
             cin>>Zapis.Kolichestvo;
             fwrite(&Zapis,sizeof(Zapis),1,stream);
           }
          clrscr();
          fclose(stream);
          return 0;
         }

        //просмотр файла
        FileView ()
        {
          if((stream = fopen(FileName,"rb"))==NULL) //открытие файла
для чтения
            {                                 //проверка
             clrscr();                          //на наличие файла
             cout<<"\n\n\n\n\n\n\t\t\t\tФАЙЛ НЕ НАЙДЕН";
             getch();
             return 0;
            }
          rewind(stream);             //репозиционируем файл
          clrscr();
                      //количество записей в файле
```

```
          for (int i=1;i<=count;i++)
          {
           fread(&Zapis,sizeof(Zapis),1,stream);
           cout<<"\n\t\t\t\tЗАПИСЬ № "<<i<<"\n";
           cout<<"\t\t\tНОМЕР ДОМА:  "<<Zapis.House<<"\n";
           cout<<"\t\t\tНОМЕР КВАРТИРЫ:  "<<Zapis.Flat<<"\n";
           cout<<"\t\t\tКОЛИЧЕСТВО ЖИЛЬЦОВ:
"<<Zapis.Kolichestvo<<"\n";
           getch();
          }
          fclose(stream);
          return 0;
         }

        //добавление новой записи
         InsertNewZapis ()
         {
          if((stream = fopen(FileName,"ab"))==NULL) //проверка
            {                                 //на наличие файла
             clrscr();
             cout<<"\n\n\n\n\n\n\t\t\t\tФАЙЛ НЕ НАЙДЕН";
             getch();
             return 0;
            }
          rewind(stream);             //репозиционируем файл
          clrscr();
          count++;
          cout<<"\n\n\n\n\n\n\n\t\t\tЗАПИСЬ НОМЕР: "<<count<<"\n";
          cout<<"\t\t\tВВЕДИТЕ НОМЕР ДОМА: ";
          cin>>Zapis.House;
          cout<<"\t\t\tВВЕДИТЕ НОМЕР КВАРТИРЫ: ";
          cin>>Zapis.Flat;
          cout<<"\t\t\tВВЕДИТЕ КОЛИЧЕСТВО ЖИЛЬЦОВ: ";
          cin>>Zapis.Kolichestvo;
          fwrite(&Zapis,sizeof(Zapis),1,stream);
          fclose(stream);
          return 0;
```

```
        }

       //поиск по ключу
        Poisk ()
        {
                    //открытие файла для чтения
         if((stream = fopen(FileName,"rb"))==NULL) //проверка
          {                              //на наличие файла
           clrscr();
           cout<<"\n\n\n\n\n\n\t\t\t\tФАЙЛ НЕ НАЙДЕН";
           getch();
           return 0;
          }
         rewind(stream);             //репозиционируем файл
         clrscr();
         char Key[6];          //ключ поиска по номеру дома
         unsigned int Key_kolichestvo;//ключ поиска по количеству
жильцов
         unsigned int vibor,proverka=0;
         clrscr();
         cout<<"\n\n\t\tПОИСК ПО ЗАДАННОМУ ПОЛЮ
ЗАПИСИ\n\n\n\n\n\n";
         cout<<"\t\t 1) ПО КОЛИЧЕСТВУ ЖИЛЬЦОВ \n\n";
         cout<<"\t\t 2) ПО НОМЕРУ ДОМА \n\n";
         cout<<"\t\t 3) ВЫХОД ИЗ ПОИСКА \n\n\n\n";
         cout<<"\t\tВВЕДИТЕ НОМЕР ОПЕРАЦИИ:  ";
         cin>>vibor;
         if(vibor==1)
          {
           cout<<"\t\tВВЕДИТЕ КОЛИЧЕСТВО ЖИЛЬЦОВ :  ";
           cin>>Key_kolichestvo;
           clrscr();
           cout<<"\n\n\t\t\t\tРЕЗУЛЬТАТЫ ПОИСКА:\n\n\n\n";
           for (int i=1;i<=count;i++)
            {
              fread(&Zapis,sizeof(Zapis),1,stream);
```

```
            if(Zapis.Kolichestvo==Key_kolichestvo) //сравниваем с
ключом
             {
              cout<<"\n\n\t\t\tНОМЕР ДОМА: "<<Zapis.House<<"\n";
              cout<<"\t\t\tНОМЕР КВАРТИРЫ: "<<Zapis.Flat<<"\n";
              cout<<"\t\t\tКОЛИЧЕСТВО ЖИЛЬЦОВ:
"<<Zapis.Kolichestvo<<"\n\n";
              proverka=1;
              getch();
             }
           }
            if(proverka==0)
             {
              cout<<"\t\t\tДАННЫЕ НЕ ОБНАРУЖЕНЫ";
              getch();
             }
          }
           if(vibor==2)
            {
             cout<<"\t\tВВЕДИТЕ НОМЕР ДОМА: ";
             cin>>Key;
             clrscr();
             cout<<"\n\n\t\t\tРЕЗУЛЬТАТЫ ПОИСКА:\n\n\n";
             for (int i=1;i<=count;i++)
              {
               fread(&Zapis,sizeof(Zapis),1,stream);
               if(stricmp(Key,Zapis.House)==0)    //сравниваем с ключом
                 {
                  cout<<"\n\n\t\t\tНОМЕР ДОМА: "<<Zapis.House<<"\n";
                  cout<<"\t\t\tНОМЕР КВАРТИРЫ: "<<Zapis.Flat<<"\n";
                  cout<<"\t\t\tКОЛИЧЕСТВО ЖИЛЬЦОВ
:"<<Zapis.Kolichestvo<<"\n";
                  getch();
                  proverka=1;
                 }
              }
             if(proverka==0)
```

```
            {
              cout<<"\t\t\tДАННЫЕ НЕ ОБНАРУЖЕНЫ";
              getch();
            }
          }
         fclose(stream);
         return 0;
        }
       //создание меню
        Menu()
        {
         clrscr();
         cout<<"\n\n\n\n\n";
         cout<<"\t\t\t   ГЛАВНОЕ МЕНЮ:\n\n\n";
         cout<<"\t\t\t1) СОЗДАНИЕ ФАЙЛА\n";
         cout<<"\t\t\t2) ПРОСМОТР ФАЙЛА\n";
         cout<<"\t\t\t3) ДОБАВЛЕНИЕ ЗАПИСИ\n";
         cout<<"\t\t\t4) ПОИСК\n";
         cout<<"\t\t\t5) ВЫХОД\n\n\n\n";
         cout<<"\t\tВЫБЕРИТЕ НОМЕР ОПЕРАЦИИ И НАЖМИТЕ
*Enter*: ";
         cin>>MenuItem;
         return 0;
        }

       //основная программа
        main()
         {
                  //количество записей в файле
       count=filelength(open(FileName,O_RDONLY))/sizeof(Zapis);
          close(open(FileName,O_RDONLY));
          do
            {
              Menu();
              if(MenuItem==1) Filecreate();
              if(MenuItem==2) FileView();
              if(MenuItem==3) InsertNewZapis();
```

```
      if(MenuItem==4) Poisk();
    }
  while(MenuItem!=5);

  return 0;
 }
```

4. Комментарии к тексту программы

В заголовочной части программы указываются подключаемые библиотеки:

<io.h> содержит конструкции и объявления для операций низкоуровневого ввода-вывода,
<fcntl.h> содержит список констант доступа к файлу,
<iostream.h> содержит базовые функции потокового ввода-вывода,
<stdio.h> содержит типы и макросы, необходимые для ввода-вывода,
<string.h> содержит функции для работы со строками,
<conio.h> содержит функции для работы с консолью ввода-вывода через DOS.

В соответствии с условиями задачи объявлен тип данных: структура Baza, включающая поля символьного и целого типов:

char House[6] номер дома
char Flat[3] номер квартиры
unsigned int Kolichestvo количество жильцов

Также объявлены переменные:

FileName - имя создаваемого файла с записями,
Stream - файловый указатель,
Zapis - переменная типа Baza,
Count - количество записей в файле целого типа,
MenuItem - номер пункта меню целого типа.

Основная функция main() содержит вызов функции Menu(). В зависимости от выбранного пункта меню реализуется соответствующая функция. Вызов функции осуществляется указанием её имени и передачей фактических параметров. Фактические параметры могут отсутствовать, тогда записывается ( ).

Функция filelength() возвращает размер файла в байтах. Разделив это число на размер записи (в нашем случае 6+3+3=9 байт), получим количество записей в файле. Если файл не создан, то функция filelength() возвращает 0.

Функция Filecreate() осуществляет формирование файла с записями.

Файловому указателю присваивается соответствующий номер, определяемый при открытии файла. Для открытия используется функция fopen(), параметрами которой являются:

- имя создаваемого файла(FileName),
- параметры передачи данных (w – запись в двоичном формате - (b)).

После ввода числа записей (count) организуется цикл ввода полей структуры (House, Flat, Kolichestvo). Для вывода в файл используется функция fwrite() – запись в поток:

&Zapis	адрес выводимой записи,
sizeof(Zapis)	размер выводимого одного экземпляра структуры (в байтах), для определения размера структуры используют функцию sizeof(),
1	количество выводимых записей,
stream	файловый указатель.

По завершении ввода записей файл закрывается функцией fclose(stream).

Функция FileView() – просмотр файла. Вначале выполняется проверка того, что файл существует. Файл отсутствует, если при выполнении функции открытия файла fopen() она возвращает в качестве значения указателя NULL. ‘rb’- открытие файла для чтения (r) в двоичном формате (b). При отсутствии файла выводится сообщение «Файл не найден». Функция rewind(stream) позиционирует файловый указатель на начало файла.

Запись считывается из файла функцией fread(). Она имеет следующие параметры:

&Zapis	адрес, по которому размещается вводимая запись,
sizeof(Zapis)	размер вводимого одного экземпляра структуры,
1	количество вводимых записей,

stream файловый указатель.

Функция InsertNewZapis() – добавление новой записи.

Новая запись добавляется в конец файла. Файл открывается на добавление (a) в двоичном формате (b). После установки файлового указателя на начало и увеличения числа записей на 1 осуществляется формирование полей структуры и её вывод в файл функцией fwrite(). Функция имеет параметры аналогичные функции fread().

Функция Poisk() – поиск в файле по заданному ключу. Функцией fopen() файл открывается на чтение. На экран выводится меню ключей поиска: по количеству жильцов, по номеру дома. После выбора ключа и задания его значения, поиск реализуется последовательным чтением записей из файла функцией fread(). Функция stricmp() сравнивает введённое значение ключа с соответствующим полем записи. При совпадении поля записи выводятся на экран, в противном случае выводится сообщение «Данные не обнаружены».

Функция Menu() – меню программы. На экран выводятся пункты меню работы программы.

5. Содержание отчёта:
   - задание;
   - листинг программы с комментариями;
   - контрольный пример.

6. Контрольные вопросы

1. Поясните структуру и параметры функции неформатированного ввода fread() и неформатированного вывода fwrite().
2. Что понимается под структурой в языке C++?
3. Что понимается под объединением в языке C++?
4. Как может осуществляться инициализация полей структуры?
5. Каким образом осуществляется доступ к элементам структуры?
6. Как осуществляется инициализация полей объединения?

Лабораторная работа №6

Динамические структуры данных

1. Цель работы:

приобрести навыки программирования, отладки и тестирования операций с динамическими структурами данных.

2. Условия:

организовать работу со списком (номер студента, фамилия студента). Обеспечить создание списка, добавление новых данных в список, просмотр, поиск по номеру и фамилии студента.

3. Листинг программы

```
#include <math.h>     //необходим для fmod()
#include <alloc.h>
#include <conio.h>
#include <iostream.h>
#include <string.h>
#define SPISOK struct spisok //идентификатору SPISOK
присвоена
                             //последовательность struct spisok
 struct spisok
  {
    unsigned short int nomer;       //номер студента
    char FIO[10];                   //ФИО студента
    SPISOK *next_adres; //указатель на следующий элемент
списка
  };
 int MenuItem,nomer_spiska=0;
 SPISOK *active_spisok;             //указатель на текущий элемент
 SPISOK *first_spisok=NULL;   //указатель на первый элемент
 SPISOK *end_spisok=NULL;   //указатель на последний элемент
```

```
        //добавление нового элемента списка
         INSERT ()
         {
           char FIO[10];
           unsigned int nomer;
           SPISOK *pred_spisok;            //указатель на предыдущий
элемент
           SPISOK *insert_spisok;  //указатель на вставляемый элемент
           cout<<"\n\n\n\n\n\n\n\t\t\t"<<++nomer_spiska<<"-й ЭЛЕМЕНТ
СПИСКА:"<<"\n";
           cout<<"\t\t\tВВЕДИТЕ НОМЕР СТУДЕНТА: ";
           cin>>nomer;
           cout<<"\t\t\tВВЕДИТЕ ФАМИЛИЮ СТУДЕНТА: ";
           cin>>FIO;
        // проверка и выделение памяти под новый элемент списка
           if((insert_spisok=(SPISOK*)malloc(sizeof(SPISOK)))==NULL)
            {
             cout<<"НЕДОСТАТОЧНО ПАМЯТИ";
             getch();
             return 1;
            }
           insert_spisok->nomer=nomer;
           strcpy(insert_spisok->FIO,FIO);
        //если список ещё пустой
           if(first_spisok==NULL && end_spisok==NULL)
             {
              first_spisok=insert_spisok;
              end_spisok=insert_spisok;
              end_spisok->next_adres=NULL;
             }
             else //если в списке уже есть элементы
               {   //защита от повторного ввода номера студента
                active_spisok=first_spisok;
                while(active_spisok!=NULL) //до конца списка
                  {
                    if(insert_spisok->nomer==active_spisok->nomer)
                      {
```

```
                    cout<<"\t\tЭЛЕМЕНТ СПИСКА С ТАКИМ
НОМЕРОМ УЖЕ СУЩЕСТВУЕТ!!!";
                    free(insert_spisok);
                    getch();
                    return 0;  //выход
                   }
                  active_spisok=active_spisok->next_adres;
                 }
                //вставка перед 1ым
                if(insert_spisok->nomer<first_spisok->nomer)
                 {
                  insert_spisok->next_adres=first_spisok;
                  first_spisok=insert_spisok;
                 }
                 else //вставка в середину
                   {
                    active_spisok=first_spisok->next_adres; //переход на 2
элемент
                    pred_spisok=first_spisok;
                    while(active_spisok!=NULL) //до конца списка
                     {
                      if(insert_spisok->nomer<active_spisok->nomer)
                        {
                         pred_spisok->next_adres=insert_spisok;
                         insert_spisok->next_adres=active_spisok;
                         return 0;  //выход
                        }
                      pred_spisok=pred_spisok->next_adres;
                      active_spisok=active_spisok->next_adres;
                     }
                    //вставка в конец
                    end_spisok->next_adres=insert_spisok;
                    end_spisok=insert_spisok;
                    end_spisok->next_adres=NULL;
                   }
              }
        return 0;
```

```
        }

        TABLICA()
         {
          gotoxy(25,wherey());
          cout<<"------=<[ ПРОСМОТР ]>=------\n";
          cout<<"| АДРЕС ЭЛЕМЕНТА | НОМЕР СТУДЕНТА |
ФАМИЛИЯ | АДРЕС СЛЕДУЮЩЕГО ЭЛЕМНТА |\n";
           for (int i=1;i<73;i++)
             {
               gotoxy(i,wherey());
               cout<<"=";
             }
           cout<<'\n';
          return 0;
         }

       //просмотр списка
        VIEW()
        {
         unsigned int i=0;//количество выведенных на экран данных
         active_spisok=first_spisok;//переход на начало списка
         clrscr();
         TABLICA();
         while(active_spisok!=NULL)
          {
           i++;
           //поэкранный вывод информации
           if (fmod(i,11)==10)//если выведено 10 элементов, то сделать
паузу
             {
               cout<<"\t\tДля продолжения просмотра нажмите любую
клавишу\n";
               getch();
               clrscr();
               TABLICA();
             }
```

```
   cout<<"|   "<<active_spisok;
   gotoxy(18,wherey());
   cout<<"|   "<<active_spisok->nomer;
   gotoxy(35,wherey());
   cout<<"|"<<active_spisok->FIO;
   gotoxy(45,wherey());
   cout<<"|   "<<active_spisok->next_adres<<'\n';
   for (int j=1;j<73;j++)
     {
       gotoxy(j,wherey());
       cout<<"-";
     }
   cout<<'\n';
   active_spisok=active_spisok->next_adres;
  }
 cout<<"\n\t\tДля возврата в меню нажмите любую клавишу";
 getch();
return 0;
}

//поиск
 POISK()
 {
  char key[10];           //ключ поиска
  unsigned short vibor,proverka=0,key_nomer;
  clrscr();
  //вывод меню поиска
  cout<<"\n\n\t\tПОИСК ПО ЗАДАННОМУ КЛЮЧУ\n\n\n\n\n\n";
  cout<<"\t\t 1) ПО ФАМИЛИИ СТУДЕНТА \n\n";
  cout<<"\t\t 2) ПО НОМЕРУ СТУДЕНТА \n\n";
  cout<<"\t\t 3) ВЫХОД ИЗ ПОИСКА \n\n\n\n";
  cout<<"\t\tВВЕДИТЕ НОМЕР ОПЕРАЦИИ:  ";
  cin>>vibor;
  if(vibor==1)
   {
    active_spisok=first_spisok;//переход на начало списка
    cout<<"\t\tВВЕДИТЕ ФАМИЛИЮ СТУДЕНТА :  ";
```

```
          cin>>key;
          clrscr();
          cout<<"\n\n\t\t\tРЕЗУЛЬТАТЫ ПОИСКА:\n\n\n\n";
          while(active_spisok!=NULL) //до конца списка
           {
            if(stricmp(key,active_spisok->FIO)==0) //сравниваем с ключом
             {
               cout<<"\n\n\t\t\tНОМЕР СТУДЕНТА: "<<active_spisok-
>nomer<<"\n";
               cout<<"\t\t\tФАМИЛИИ СТУДЕНТА: "<<active_spisok-
>FIO;
               proverka=1;       //данные обнаружены
               getch();
             }
            active_spisok=active_spisok->next_adres;
           }
          if(proverka==0)      //данные не обнаружены
           {
             cout<<"\t\t\tДАННЫЕ НЕ ОБНАРУЖЕНЫ";
             getch();
           }
         }
        if(vibor==2)
         {
          active_spisok=first_spisok;
          cout<<"\t\tВВЕДИТЕ НОМЕР СТУДЕНТА: ";
          cin>>key_nomer;
          clrscr();
          cout<<"\n\n\t\t\tРЕЗУЛЬТАТЫ ПОИСКА:\n\n\n";
          while(active_spisok!=NULL)  //до конца списка
           {
            if(key_nomer==active_spisok->nomer)    //сравниваем с
ключом
              {
               cout<<"\n\n\t\t\tНОМЕР СТУДЕНТА: "<<active_spisok-
>nomer<<"\n";
```

```
                cout<<"\t\t\tФАМИЛИИ СТУДЕНТА: "<<active_spisok->FIO<<"\n";
                getch();
                proverka=1;     //данные обнаружены
               }
             active_spisok=active_spisok->next_adres;
            }
           if(proverka==0)     //данные не обнаружены
            {
              cout<<"\t\t\tДАННЫЕ НЕ ОБНАРУЖЕНЫ";
              getch();
            }
         }
        return 0;
        }

       //УДАЛЕНИЕ ЭЛЕМЕНТА СПИСКА
        DELETE()
        {
          //указатель на элемент, содержащий адрес удаляемого элемента
          SPISOK *pred_spisok;
          unsigned short proverka=0,key_nomer;
          clrscr();
          cout<<"\tВВЕДИТЕ НОМЕР СТУДЕНТА, ЧЬИ ДАННЫЕ ВЫ ХОТИТЕ УДАЛИТЬ:  ";
          cin>>key_nomer;

          active_spisok=first_spisok;
          while(active_spisok!=NULL)  //до конца списка
           {
            if(key_nomer==active_spisok->nomer)    //сравниваем с ключом
             {
               cout<<"\n\n\t\t\tЭТИ ДАННЫЕ УДАЛЕНЫ: \n";
               cout<<"\n\n\t\t\tНОМЕР СТУДЕНТА: "<<active_spisok->nomer<<"\n";
```

```
            cout<<"\t\t\tФАМИЛИИ СТУДЕНТА: "<<active_spisok->FIO<<"\n";
            getch();
            proverka=1;       //данные обнаружены
            break;
           }
          active_spisok=active_spisok->next_adres;
         }
        if(proverka==0)       //данные не обнаружены
         {
          cout<<"\t\tДАННЫЕ НЕ ОБНАРУЖЕНЫ,-> УДАЛЕНИЯ НЕ ПРОИЗОШЛО";
          getch();
          proverka=0;
          return 0;               //выход
         }
        pred_spisok=first_spisok;
      //переход на элемент, содержащий указатель на удаляемый элемент
        while(pred_spisok!=NULL)  //до конца списка
         {
          if(pred_spisok->next_adres==active_spisok)
            break;       //да, элемент с этим указателем есть
          pred_spisok=pred_spisok->next_adres;
         }

        //если удаляется единственный элемент
        if(active_spisok==first_spisok && active_spisok->next_adres==NULL)
         {
          first_spisok=NULL;
          end_spisok=NULL;
          free(active_spisok);
          free(pred_spisok);
          nomer_spiska=0;
          return 0;       //выход
         }
```

```
          //удаление первого элемента списка
          if(pred_spisok->next_adres!=active_spisok)
           {
            active_spisok=first_spisok;
            first_spisok=first_spisok->next_adres;
            free(active_spisok);
            nomer_spiska--;
            return 0;       //выход
           }
        //если удаляемый элемент находится после 1го элемента списка
           pred_spisok->next_adres=active_spisok->next_adres;
           //удаление последнего элемента
           if(active_spisok->next_adres==NULL)
             end_spisok=pred_spisok;
           free(active_spisok);
           nomer_spiska--;
         return 0;
         }

         //создание меню
          MENU()
          {
           clrscr();
           cout<<"\n\n\n\n\n";
           cout<<"\t\t\t   ГЛАВНОЕ МЕНЮ:\n\n\n";
           cout<<"\t\t\t1) ДОБАВЛЕНИЕ ДАННЫХ В СПИСОК\n";
           cout<<"\t\t\t2) ПРОСМОТР СПИСКА\n";
           cout<<"\t\t\t3) ПОИСК В СПИСКЕ\n";
           cout<<"\t\t\t4) УДАЛЕНИЕ ЭКЗЕМПЛЯРА СПИСКА\n";
           cout<<"\t\t\t5) ВЫХОД\n\n\n\n";
           cout<<"\t\tВЫБЕРИТЕ НОМЕР ОПЕРАЦИИ И НАЖМИТЕ
*Enter*: ";
           cin>>MenuItem;
           return 0;
          }

         //основная программа
```

```
main()
 {
  do
    {
      MENU();
       if(MenuItem==1) INSERT();
       if(MenuItem==2) VIEW();
       if(MenuItem==3) POISK();
       if(MenuItem==4) DELETE();
    }
  while(MenuItem!=5);
  free(first_spisok);
  free(active_spisok);
  free(end_spisok);
  return 0;
 }
```

4. Комментарии к тексту программы

Основой данной динамической структуры является однонаправленный список. Поэтому в элементе списка наряду с полями данных должны присутствовать адресные поля. Для однонаправленного списка необходимо одно адресное поле, содержащее адрес следующего элемента списка (SPISOK *next_adres).Очевидно, что последний элемент списка в этом поле будет иметь значение NULL.

Для работы со списком вводятся три указателя: на первый элемент списка (first_spisok), на последний элемент (end_spisok), на текущий элемент (active_spisok).

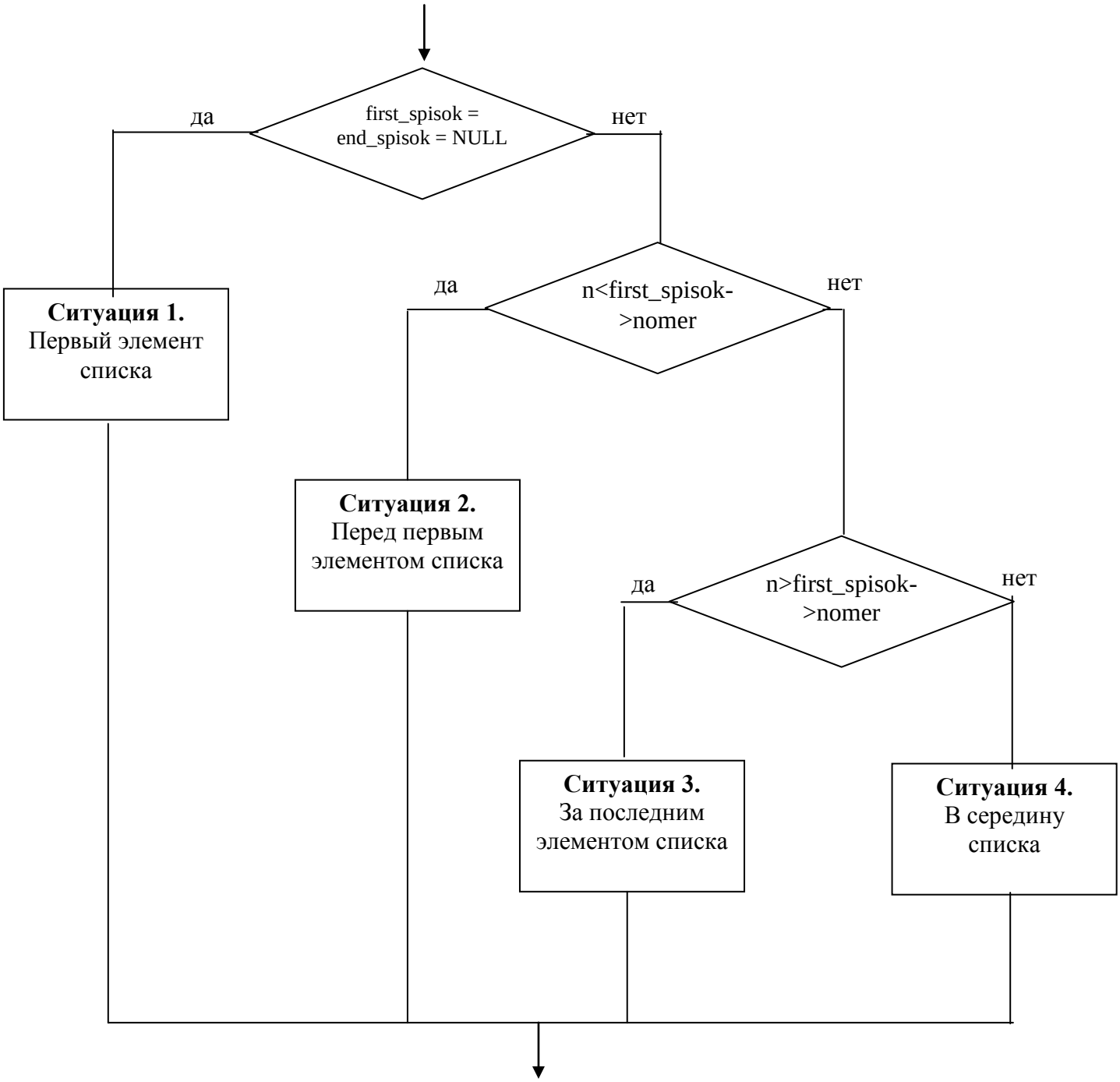
Наиболее сложные действия приходится выполнять при добавлении и исключении элемента списка.

При добавлении необходимо, чтобы элементы списка шли в порядке возрастания (или убывания) значения поля nomer. В программе объявлена структура элемента списка:

```
struct spisok
  {
   unsigned short int nomer;       //номер студента
```

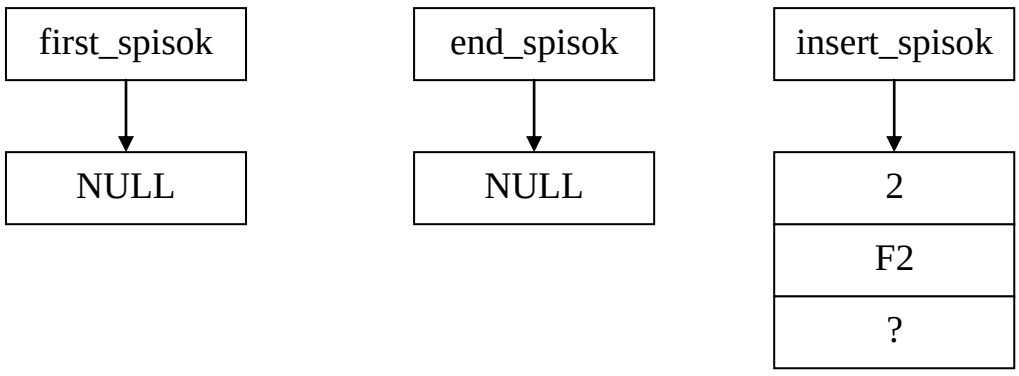
```
        char FIO[10];              //ФИО студента
        SPISOK *next_adres; //указатель на следующий элемент
списка
      };
```

Рассмотрим функцию добавления нового элемента списка (INSERT( )). Логика выбора варианта добавления реализуется в соответствии со схемой (n – значение поля nomer добавляемого элемента списка):

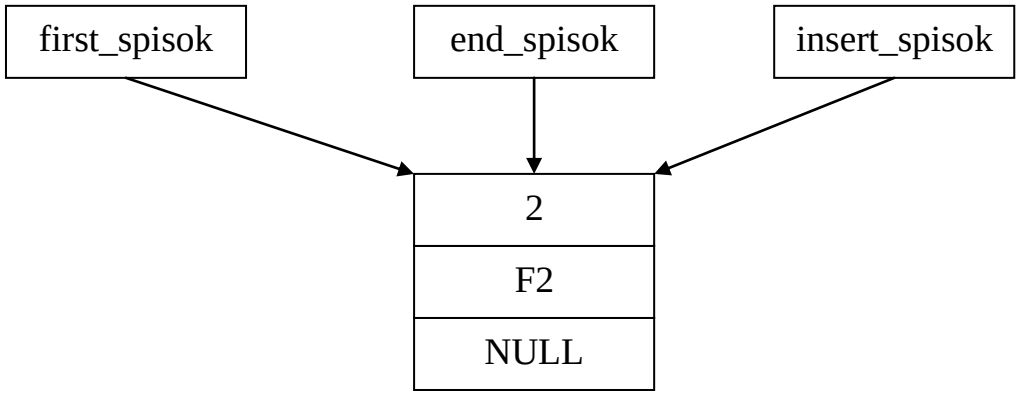


Графически ситуацию можно отобразить следующим образом.
Ситуация 1.

Исходное состояние:



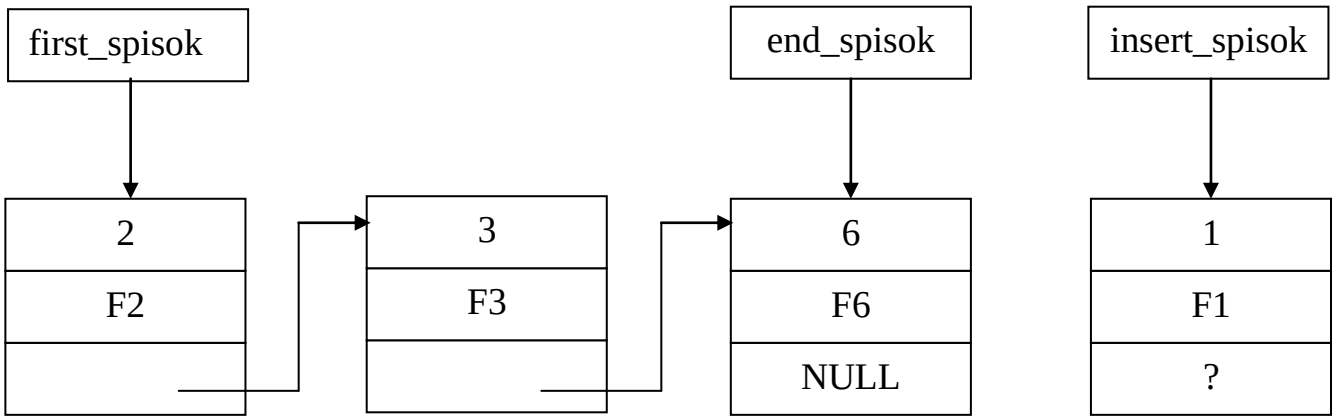
Конечное состояние:



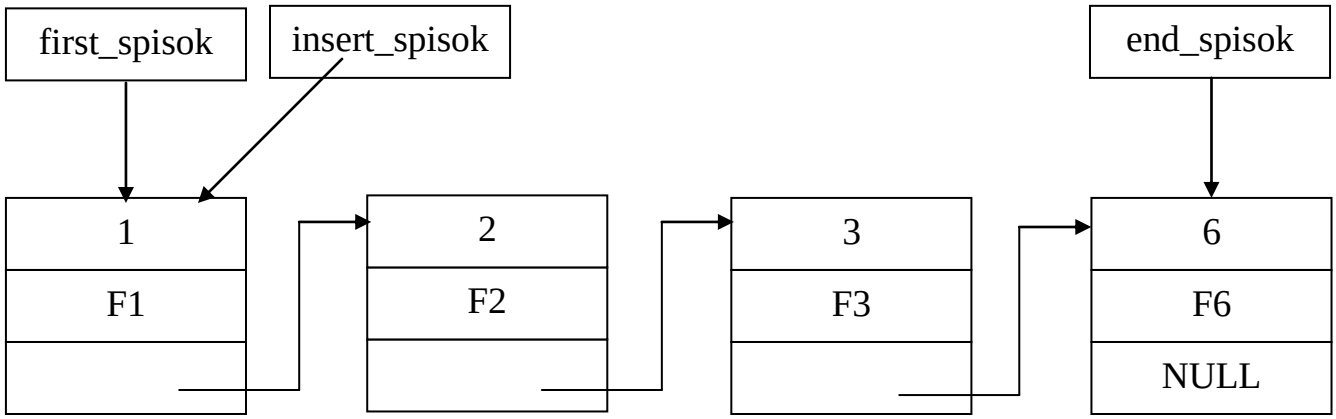
Необходимо: в указатели first_spisok и end_spisok записать адрес, содержащийся в указателе insert_spisok. В адресное поле элемента записать NULL.

Ситуация 2.

Исходное состояние:



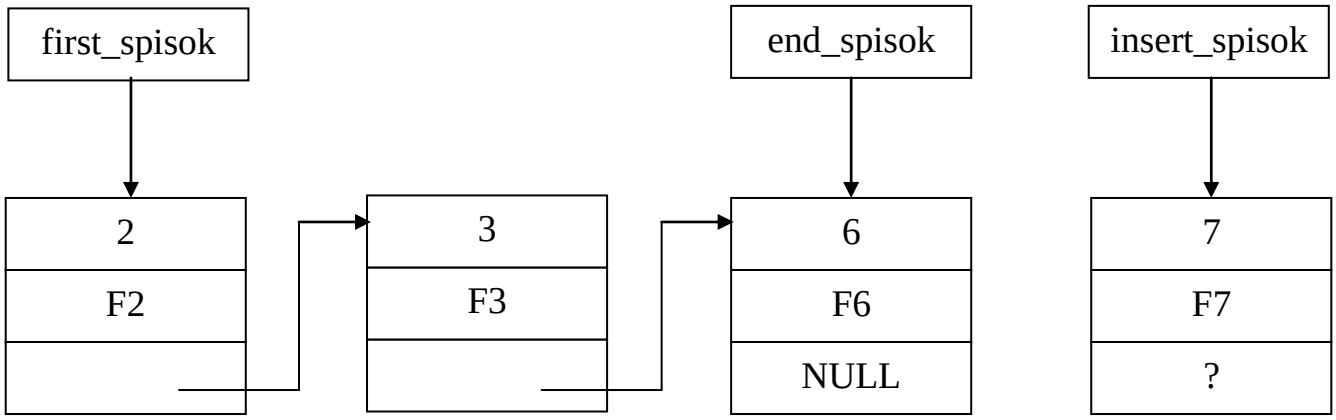
Конечное состояние:



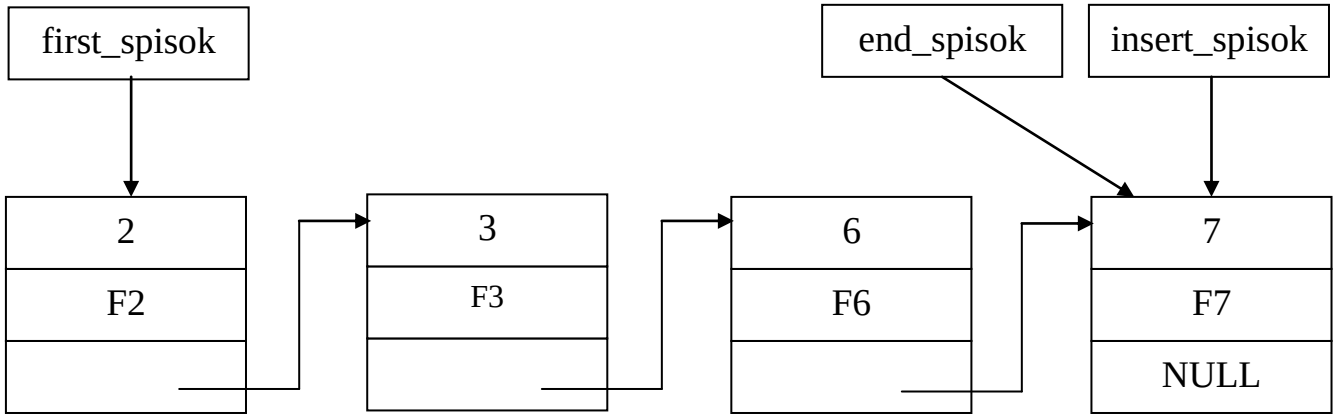
Необходимо: в поле адреса нового элемента записать содержимое указателя first_spisok, присвоить указателю first_spisok значение указателя insert_spisok.

Ситуация 3.

Исходное состояние:



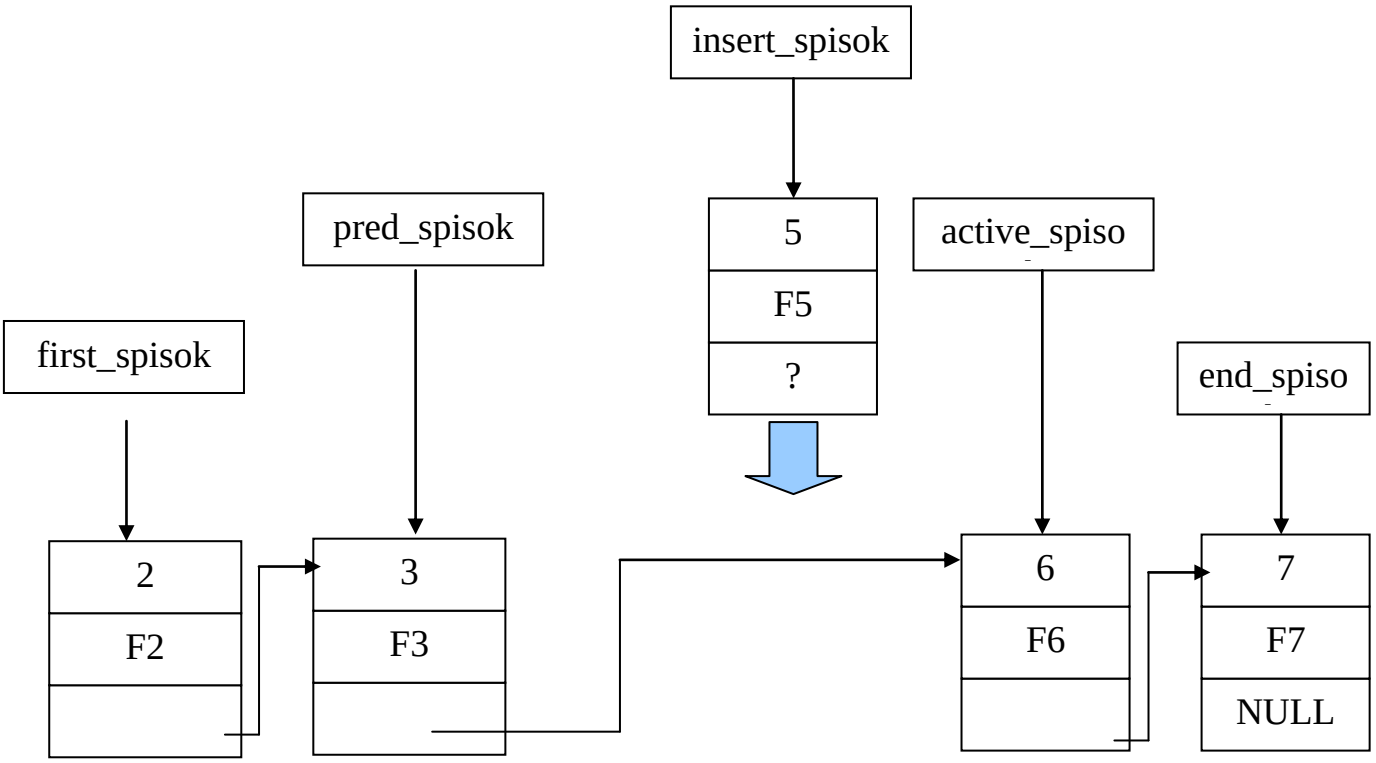
Конечное состояние:



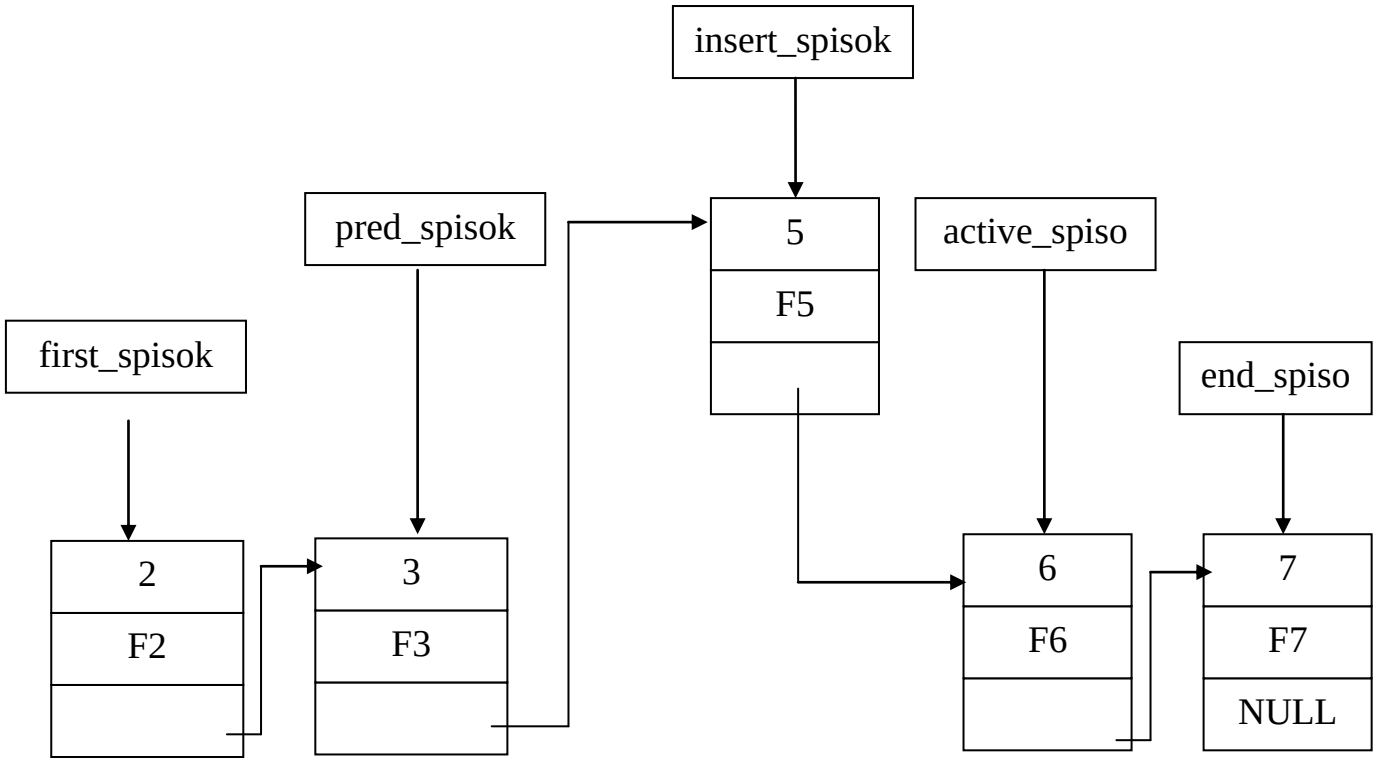
Необходимо: в поле адреса последнего элемента записать вместо NULL содержимое указателя insert_spisok; указателю end_spisok присвоить значение insert_spisok; в поле адреса нового (последнего) элемента записать NULL.

Ситуация 4.

Исходное состояние:



Конечное состояние:



Необходимо: определить элемент списка после которого необходимо вставить новый элемент (или элемент списка, перед которым необходимо вставить новый элемент); в поле адреса нового элемента записать содержимое поля адреса предыдущего элемента; в поле адреса предыдущего элемента записать значение указателя insert_spisok.

В функции INSERT( ) объявлен указатель *pred_spisok на предыдущий элемент, указатель insert_spisok – на вставляемый элемент. Запрашивается номер студента и вводятся поля структуры элемента списка.

Функция malloc(sizeof(SPISOK)) выделяет из динамической памяти требуемый объём (sizeof(SPISOK)), адрес выделяемой памяти записывается в указатель insert_spisok. Если память по каким-либо причинам не может быть выделена, то выдаётся сообщение «Недостаточно памяти». После получения адреса заполняются поля элемента списка.

В соответствии с выше описанными ситуациями программируются необходимые действия. При реализации вставки в середину (между первым и последним элементами списка) необходимо определить позицию вставки. Поиск осуществляется следующим образом: в указатель active_spisok записывается адрес, содержащийся в поле адреса элемента с указателем first_spisok (т.е. переход на 2-ой элемент списка). Указателю pred_spisok присваивается значение указателя first_spisok. Далее организуется цикл поиска позиции вставки. С этой целью сравнивается значение поля nomer нового элемента с аналогичным полем текущего элемента. При выявлении ситуации «новый < текущего»

```
if(insert_spisok->nomer<active_spisok->nomer)
  {
   pred_spisok->next_adres=insert_spisok;
   insert_spisok->next_adres=active_spisok;
   return 0;  //выход
  }
```

формируется в указателе active_spisok адрес элемента списка, перед которым необходимо вставить новый элемент, а в pred_spisok хранится адрес элемента, после которого необходимо вставить новый элемент. Используя эти указатели, выполняются необходимые действия по включению. Этот отрывок программы можно немного переделать:

```
if(insert_spisok->nomer<active_spisok->nomer)
 {
  insert_spisok->next_adres= pred_spisok->next_adres;
  pred_spisok->next_adres=insert_spisok;
  return 0;  //выход
 }
```

Другие ситуации программируются достаточно просто.

Функция TABLICA ( ) формирует на экране заголовок таблицы для вывода полей и адреса структуры.

Функция VIEW ( ) реализует просмотр элементов списка. Для этого в указатель active_spisok записывается адрес начала списка (first_spisok) и осуществляется последовательный просмотр. Адрес

очередного элемента списка извлекается из адресного поля текущего элемента (active_spisok= active_spisok-> next_adres).

Функция POISK( ) реализует поиск и вывод на экран элемента списка в соответствии с фамилией или номером студента. Реализация аналогична действиям в функции VIEW ( ), только дополнительно проверяется условие на совпадение введенногозначения с соответствующим полем элемента списка.

Функция удаления элемента списка DELETE ( ). В этой функции вначале осуществляется поиск элемента с указанным номером. При нахождении такового выводятся на экран поля элемента списка, и в указателе active_spisok фиксируется адрес удаляемого элемента, а в указателе pred_spisok – адрес элемента списка, предшествующего удаляемому элементу.

Действия по удалению элемента состоят в следующем.

Если элемент в списке единственный, то в указатели first_spisok и end_spisok заносится нулевой адрес NULL и освобождается память, занятая этим элементом (free(active_spisok)).

Если элемент первый в списке, то в указатель first_spisok записывается адрес следующего элемента за первым, освобождается память, уменьшается значение числа элементов списка.

Если удаляемый элемент находится за первым элементом списка, то в адресное поле предыдущего элемента необходимо записать значение адресного поля удаляемого элемента (pred_spisok->next_adres=active_spisok->next_adres), а если элемент последний, то указателю end_spisok следует присвоить значение указателя pred_spisok. Затем освобождается память и уменьшается количество элементов списка.

5. Содержание отчёта:

- задание;
- листинг программы с комментариями;
- контрольный пример.

6. Контрольные вопросы

1. Какие поля необходимо включать в структуру элемента списка помимо информационных?

2. Запишите структуру элемента двунаправленного списка, очереди, стека.
3. Выразите графически ситуации удаления элементов списка.
4. Какая функция используется для определения размера элемента списка?
5. Какая функция используется для выделения динамической памяти?
6. Какие указатели следует использовать при организации списка?
7. Запишите операторы добавления элемента в очередь, стек.
8. Запишите операторы исключения элемента из очереди, стека.