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 самостоятельных 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 самостоятельных работ для студентов направления подготовки 12.03.04 – «Биотехнические системы и технологии» (бакалавр) / Юго-Зап. гос. ун-т; сост.: А.А.Кузьмин. - Курск, 2023. - 50 с.

Содержат методические рекомендации к проведению самостоятельных 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 25.08.23 Формат 60х84 1/16
Усо.печ.л. 2,9 . Уч.-изд.л. 2,6 .Тираж 30 экз. Заказ: 1074 .Бесплатно.
Юго-Западный государственный университет.
305040. г. Курск, ул. 50 лет Октября, 94.

Работа №1
Программирование вложенных циклических вычислительных процессов

1. Цель работы:

приобрести навыки программирования, отладки и тестирования вложенных циклических вычислительных процессов.

2. Условия:

даны целые числа A1..An. Получить все числа, которые входят в последовательность по одному разу.

3. Листинг программы

```
#include <conio.h>
#include <iostream.h>
int main()
  {
   clrscr();
      int array[10];              //исследуемый массив
      int mas_little[10];         //массив одновстречающихся чисел
      int m=0,n=0,k=0;            //вспомогательные переменные
      cout<<"Введите количество элементов последовательности: ";    // не больше 10и
      cin>>n;
      for(int i=0;i<n;i++)
          {
           cout<<"Введите ["<<i+1<<"] элемент массива: ";
           cin>>array[i];
          }

      for(i=0;i<n;i++)
        {
          for(int j=0;j<n;j++)
              if(array[i]==array[j]) k++;  //k-количество совпадений
```

```
            if(k==1)
              {
                mas_little[m]=array[i];
                k=0;
                m++;
              }
             else k=0;
            }

      //вывод массива на экран
            cout<<"\t\t\tИСХОДНЫЙ МАССИВ:"<<'\t'<<'\n';
              for(int j=0;j<n;j++)
                cout<<array[j]<<'\t';
            cout<<'\n';

            cout<<"ЭЛЕМЕНТЫ, ВХОДЯЩИЕ В
ПОСЛЕДОВАТЕЛЬНОСТЬ 1 РАЗ:"<<'\t'<<'\n';
            for(i=0;i<m;i++)
              cout<<mas_little[i]<<'\t';

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

4. Комментарии к тексту программы

Объявлен массив array из 10 элементов целого типа, и массив mas_little из 10 элементов целого типа, а также ряд переменных (m, n, k) целого типа.

В цикле:

```
for(int i=0;i<n;i++)
    {
     cout<<"Введите ["<<i+1<<"] элемент массива: ";
     cin>>array[i];
    }
```

реализуется ввод элементов массива array. Затем организуется двойной цикл (внешний по i и внутренний по j), реализующий

действия, определённые в условиях задачи. Происходит последовательное сравнение i-го элемента массива array с j-ым элементом того же массива. Количество совпадений подсчитывается в переменной k, затем, если k=1 (т.е. совпадение числа с самим собой), то в массив mas_little записывается элемент из массива array:

mas_little[m]=array[i];

В противном случае запись не происходит, а переменная k обнуляется. Переменная m определяет индекс элемента массива mas_little.

Последние операторы цикла организуют вывод на экран соответственно исходного и результирующего массивов.

Те же действия можно реализовать, используя указатели. Ниже представлен текст функции main с использованием указателей.

```
int main()
  {
   clrscr();
      int array[10];                  //исследуемый массив
      int mas_little[10];             //массив одновстречающихся
чисел
      int n=0,k=0;                    //k-количество совпадений
      int* p_array=array;
      int* p_mas_little=mas_little;
      cout<<"Введите количество элементов последовательности:
";
      cin>>n;
      for(int i=0;i<n;i++)
          {
           cout<<"Введите ["<<i+1<<"] элемент массива: ";
           cin>>*p_array++;
          }
      p_array=array;                  //вернулись к началу массива
      for(i=0;i<n;i++)
        {
         for(int j=0;j<n;j++)
             if(*(p_array+i)==*(p_array+j)) k++;

         if(k==1)
```

```
                {
                  *p_mas_little++=*(p_array+i);
                  k=0;
                }
              else k=0;
            }
      //вывод массива на экран
            cout<<"\t\t\tИСХОДНЫЙ МАССИВ:"<<'\t'<<'\n';
               for(i=0;i<n;i++)
                  cout<<*p_array++<<'\t';
            cout<<'\n';

            cout<<"ЭЛЕМЕНТЫ, ВХОДЯЩИЕ В
ПОСЛЕДОВАТЕЛЬНОСТЬ 1 РАЗ:"<<'\t'<<'\n';

            for(i=1;i<=p_mas_little-mas_little;i++)
              cout<<*(p_mas_little-i)<<'\t';

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

Объявлены указатели на массив array и mas_little. В дальнейшем указатели используются для ссылки и выборки элементов из массива.

В конструкции

*p_array++

вначале выполняется операция разыменовывания (т.е. чтение элемента по адресу), а затем инкрементирование адреса указателя на единицу длины типа данных.

В конструкции

*(p_array+i)

адрес указателя увеличивается на значение i, а затем по сформированному адресу читается из массива элемент.

Правильной является запись вида:

p_array=array;

т.к. имя массива ассоциируется компилятором с адресом нулевого элемента массива array.

Конструкция

p_mas_little++=(p_array+i);

реализует следующие действия: из массива array в соответствии с адресом указателя p_array выбирается элемент и записывается в массив по адресу, содержащемуся в указателе p_mas_little. Затем адрес указателя инкрементируется на единицу длины типа данных.

Необходимо слева использовать операцию разыменовывания, иначе запись вида:

p_mas_little++=*(p_array+i);

будет неправильной, т.к. правая часть выражения - численное значение, а левая – адрес, следовательно, будет несоответствие типов. Напомню, что адрес байта памяти – это два двухбайтных поля, определяющих кодовый сегмент и смещение внутри сегмента.

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

1. Что представляет собой указатель в C++?
2. Каков формат объявления указателя?
3. Что определяет следующее объявление?

   Char *ps = &symvol;
4. Какую операцию необходимо использовать, чтобы получить значение, записанное в некоторой области памяти, на которую ссылается указатель?
5. Какие арифметические операции применимы над указателями?
6. Что объявлено в следующем предложении?

   Int **pPtrInt;
7. Если объявлен указатель на указатель (**), то сколько раз должен быть разыменован такой указатель?
8. Объявлен указатель типа:

   Int *ptr;

   а) ++ptr;

   б) ptr++;

   в) --ptr;

г) ptr--;

Работа №2
Программирование матричных операций

1. Цель работы:

приобрести навыки программирования, отладки и тестирования операций с матрицами.

2. Условия:

дана матрица размерностью m*n. Получить последовательность B1…Bn, где Bi – сумма элементов в i-ой строке, расположенных за первым отрицательным элементом.

3. Листинг программы

```
#include <conio.h>
#include <iostream.h>
int main()
  {
   clrscr();
      int array[10][10];          //матрица
      int mas[10];                //массив суммы
      int m=0,n=0;                //количество строк и столбцов
      cout<<"Введите количество строк: ";
      cin>>m;
      cout<<"Введите количество столбцов: ";
      cin>>n;
      for(int i=0;i<m;i++)        //обнуление массива суммы
        mas[i]=0;
      for(i=0;i<m;i++)
         for(int j=0;j<n;j++)
          {
            cout<<"Введите [";
            cout<<i+1<<"][";
            cout<<j+1;
```

```
            cout<<"] элемент матрицы: ";
            cin>>array[i][j];
           }

        for(i=0;i<m;i++)
          for(j=0;j<n;j++)                    //          <--
            if(array[i][j]<0)               //            |
              {                           //            |
               for(j+1;j+1<n;j++)         //            |
                 mas[i]+=array[i][j+1];   //            |
               break;             //выход из внутреннего цикла
              }
    //вывод матрицы и массива на экран
        cout<<"\t\t\tИСХОДНАЯ МАТРИЦА:"<<'\t'<<'\n';
        for(i=0;i<m;i++)
         {
          for(j=0;j<n;j++)
            cout<<array[i][j]<<'\t';
          cout<<'\n';
         }
        cout<<"МАССИВ СУММ ЭЛЕМЕНТОВ В i-ой СТРОКЕ
ПОСЛЕ 1-ГО ОТРИЦАТЕЛЬНОГО ЭЛЕМЕНТА:"<<'\t'<<'\n';
        for(i=0;i<m;i++)
          cout<<mas[i]<<'\t';

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

4. Комментарии к тексту программы

Объявлена матрица int array[10][10]. Таким образом, двумерный массив – это массив, каждый элемент которого представляет собой массив из 10 элементов.

Обращение к элементу двумерного массива – указание номера строки и столбца. Каждый индекс заключается в квадратные скобки (смотри двойной цикл ввода элементов матрицы array).

После ввода элементов матрицы организуется вложенный цикл для реализации условий задачи.

В цикле проверяется условие:

if(array[i][j]<0)

Если оно выполняется, то вычисляется сумма элементов за первым отрицательным числом. При этом используется оператор присваивания со сложением. По завершении этого цикла выполняется вывод исходной матрицы и результирующего массива mas.

Несколько иной вид имеет программа при использовании указателей. Использование указателей считается эффективным для адресации элементов массива.

```
int main()
  {
   clrscr();
      int array[10][10];            //матрица
      int mas[10];                  //массив суммы
      int m=0,n=0;                  //количество строк и столбцов
      int* p_array=(int*)array;     //двойное разыменовывание
      int* p_mas=mas;
      cout<<"Введите количество строк: ";
      cin>>m;
      cout<<"Введите количество столбцов: ";
      cin>>n;
      for(int i=0;i<m;i++)          //обнуление массива суммы
        *(p_mas++)=0;
      p_mas=mas;                    //возвращаем указатель в начало
массива
      for(i=0;i<m;i++)
         for(int j=0;j<n;j++)
          {
            cout<<"Введите [";
            cout<<i+1<<"][";
            cout<<j+1;
            cout<<"] элемент матрицы: ";
            cin>>*(p_array+i*10+j);
          }
```

```
        for(i=0;i<m;i++)
           for(j=0;j<n;j++)                    //                <--
             if(*(p_array+i*10+j)<0)          //                   |
               {                              //                   |
                 for(j+1;j+1<n;j++)           //                   |
                   *(p_mas+i)+=*(p_array+i*10+j+1);                |
                 break;              //выход из внутреннего цикла
               }
    //вывод матрицы и массива на экран
        p_mas=mas;
        cout<<"\t\t\tИСХОДНАЯ МАТРИЦА:"<<'\t'<<'\n';
        for(i=0;i<m;i++)
          {
           for(j=0;j<n;j++)
             cout<<*(p_array+i*10+j)<<'\t';
           cout<<'\n';
          }
        cout<<"МАССИВ СУММ ЭЛЕМЕНТОВ В i-ой СТРОКЕ
ПОСЛЕ 1-ГО ОТРИЦАТЕЛЬНОГО ЭЛЕМЕНТА:"<<'\t'<<'\n';
        for(i=0;i<m;i++)
          cout<<*(p_mas++)<<'\t';

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

Помимо объявления матрицы и массива, объявлены указатели на матрицу (p_array) и массив (p_mas). Указатели инициируются начальным адресом матрицы array и массива mas. Доступ к элементам многомерного массива через указатели осуществляется немного сложнее. Поскольку двумерный массив array [10][10] может быть представлен как одномерный (array [10]), каждый элемент которого также является одномерным массивом (array [10]), указатель на двумерный массив p_array, ссылаясь на элемент массива array [10][10], по сути, указывает на массив array [10] в массиве array [10]. Таким образом, для доступа к содержимому ячейки памяти указатель p_array приходиться разыменовывать дважды.

В строке:

int* p_array=(int*)array;

идентификатор array уже является указателем на элемент с индексом 0, однако, поскольку массив двумерный, требуется его повторное разыменовывание.

В строке:

cin>>*(p_array+i*10+j);

осуществляется модификация адреса обращения к элементу массива. Введенное значение записывается в соответствующую позицию матрицы.

Аналогичные конструкции используются в дальнейшем при поиске первого отрицательного элемента, вычислении суммы и выводе исходной матрицы на экран.

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

6. Контрольные вопросы

1. Определите понятие массива.
2. Определите понятие элемента массива.
3. Какие средства обращения к элементам массива предлагает C++?
4. Объясните механизм доступа к элементам массива через указатель.
5. Пусть задано объявление:

   Char ArrayOfChar[]={‘w’,’o’,’r’,’l’,’d’};

   Char* pArr = ArrayOfChar;

   На какой элемент ссылается указатель, если pArr+=3; ?
6. Объясните различие в смысле конструкций:

   Char (*Array)[10]; и

   Char *Array[10];
7. Чему соответствует выражение:

   ((Array+i)+j) ?
8. Поясните порядок инициализации многомерных массивов.

Работа №3

Табулирование функций

1. Цель работы:

приобрести навыки программирования, отладки и тестирования вложенных циклических вычислительных процессов и табулирования функции.

2. Условия:

Вычислить и вывести на печать:

$$Z_{ij} = X_j^2 \cdot \arcsin(Y_i \cdot \sqrt{1 - X_j})$$

$0.1 \le Yi \le 1.0$ $\Delta Y = 0.1$

$0.1 \le Xj \le 0.2$ $\Delta X = 0.05$

3. Листинг программы

```
#include <math.h>     //необходим для ф-ии asin()-
#include <conio.h>
#include <iostream.h>
int main()
  {
   clrscr();
      float x=0.1,y=0.1;
      int i=1,j=1;        //для вывода индекса Z
      float *pX=&x,*pY=&y;
      cout<<"\t\t\t\t\t ____"<<'\n';
      cout<<"Программа вычисляет : Zij=Xi^2*arcsin(Yjv1-Xi)
"<<'\n';
      cout<<"\tгде:        0.1<=Yj<=1.0     ΔY=0.1 "<<'\n';
      cout<<"\t\t     0.1<=Xi<=0.2     ΔX=0.05 "<<'\n';
    while(*pY<1.1)
     {
      while(*pX<0.21)
       {
```

```
                cout<<"Z["<<i<<"]["<<j<<"] =
"<<(*pX)*(*pX)*asin(*pY*sqrt(1-*pX))<<'\n';
                *pX+=0.05;
                j++;
               }
              *pX=0.1;
              j=1;
              if(i==5)  //приостанавливает работу, когда выведено
                                  //5*3=15 элементов
               {
                cout<<"Для продолжения нажмите любую
клавишу..."<<'\n';
                getch();
               }
              *pY+=0.1;
              i++;
             }

            getch();
            return 0;
           }
```

4. Комментарии к тексту программы

Для реализации математических функций необходимо подключать файл math.h.

Указатели pX и pY инициируются соответственно адресом переменной x и y. Группа операторов cout выводит на экран условия задачи. Управляющий символ ‘\t’ – табуляция (перевод курсора на 5 позиций вправо); ‘\n’ – перевод строки.

Для табулирования значений функции необходима организация вложенного цикла. Поскольку аргументы функции имеют дробные значения, необходимо использование операторов цикла типа while или do…while.

Напомню, что конструкция while(*pY<1.1) означает: цикл выполняется до тех пор, пока значение, находящееся по адресу в указателе pY, будет меньше 1,1.

Аналогично рассматривается выражение while(*pX<0.21).

Запись *pX+=0.05 – оператор присваивания со сложением. Значение, находящееся по адресу в указателе pX, увеличивается на 0,05.

* - операция косвенного обращения или разыменовывания. Можно записать: *pX=*pX+0.05;

Аналогично следует рассматривать запись: *pY+=0.1 или *pY=*pY+0.1. Операторы i++ и j++ инкрементируют значение i и j на 1, соответственно. Оператор вывода имеет многокомпонентную структуру:

cout<<"Z[""	<<i=	<<"]["	<<j	<<"]="	<<(*pX)*(*pX)*…	<<'\n'
1	2	3	4	5	6	7

1 – текст: z [→ z [;
2 – значение переменной: i → z [число
3 - текст:][→ z [число][
4 – значение переменной: j → z [число][число
5 - текст:]=→ z [число][число]=
6 – вычисляемое значение функции: →z [число][число]= число
7 – перевод строки.

Так как результат работы программы не помещается на экране (выводит 30 значений), то необходимо приостановить вывод результата пока пользователь не нажмёт ENTER. Это проверяется в конструкции:

```
if(i==5)
  {
    cout<<"Для продолжения нажмите любую
клавишу..."<<'\n';
    getch();
  }
```

Оператор цикла может быть записан без использования указателей (тогда в объявлении float *pX=&x,*pY=&y нет необходимости):

```
…
while(y<1.1)
    {
```

```
            while(x<0.21)
              {
                cout<<"Z["<<i<<"]["<<j<<"] = "<<x*x*asin(y*sqrt(1-
x))<<'\n';
                x+=0.05;
                j++;
              }
            x=0.1;
            j=1;
            if(i==5) //приостанавливает работу, когда выведено
                                    //5*3=15 элементов
              {
                cout<<"Для продолжения нажмите любую
клавишу..."<<'\n';
                getch();
              }
            y+=0.1;
            i++;
          }

        getch();
        return 0;
       }
```

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

6. Контрольные вопросы

1. Задан цикл for(; ;); что определяет данный цикл?
2. В каком из циклов while или do…while сначала выполняется тело цикла, а затем уже осуществляется проверка выражения на истинность?
3. Определите понятие функции в С++.
4. Что понимается под прототипом функции?

5. Что понимается под определением функции?
6. Что понимается под вызовом функции?
7. Какие переменные называются локальными?
8. Какие переменные называются глобальными?
9. Что понимается под областью действия переменной?
10. Что понимается под областью видимости переменной?
11. Что называется временем жизни переменной?
12. Объясните использование модификаторов auto, register, static, extern для переменных?
13. Что понимается под встраиваемой функцией?
14. Приведите пример рекурсивной функции.

Работа №4

Программирование файловых операций

1. Цель работы:

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

2. Условия:

сформировать символьный файл F1, который состоит из строк, содержащие слова, разделённые символами: пробел, точка с запятой, точка. Создать файл F2, содержащий количество слов в каждой строке файла F1, и вывести на экран.

3. Листинг программы

```
#include <conio.h>
#include <iostream.h>
#include <stdio.h>
#include <math.h>
int main()
  {
      char File1[]="F1.txt";
      char File2[]="F2.txt";
      FILE* F1=fopen(File1,"wt");     //создание файла для записи;
      FILE* F2=fopen(File2,"wt");     //создание файла для записи;
      clrscr();
      char string[75];          //строка символов
      do
        {
          int i=0,n=0;                  //количество слов
          cout<<"\nВведите строку символов:\n";
          cin.getline(string,sizeof(string));
          fputs(string,F1);
```

```
        putc(13,F1);          //перевод на новую строку файла
        while(string[i]!='\0')
          {
            if(string[i]!=' ' && string[i]!=';' && string[i]!='.'
            && (string[i+1]==' ' || string[i+1]==';' || string[i+1]=='.'
            || string[i+1]=='\0'))
              {
                n++;
                i++;
              }
            else  i++;
          }
        cout<<'\n'<<"Количество слов равно "<<n;
        if (n>9)
          {
           putc((n/10+48),F2);
           putc((fmod(n,10)+48),F2);
          }
        else
          putc((n+48),F2);   //запись количества слов в F2
        putc(13,F2);          //перевод на новую строку файла
        cout<<'\n'<<"Будете ещё вводить строку?(y-да/n-нет) ";
      }
    while(getche()=='y');
    fclose(F1);
    fclose(F2);
    getch();                  //ждём нажатия клавы
    return 0;
}
```

4. Комментарии к тексту программы

Функции ввода-вывода объявлены в заголовочном файле <stdio.h>, там же объявлен модуль <math.h>, из которого используется функция fmod() – остаток от деления числа.

При программировании файловых операций используются стандартные функции работы с файлами.

Объявлены переменные File1 и File2, содержащие соответственно имя исходного (F1.txt) и результирующего файла (F2.txt). Длина переменных File1 и File2 определяется при инициализации (квадратные скобки не содержат конкретного значения).

Так же объявлены файловые указатели F1 и F2, которые инициируются функцией открытия файлов fopen().

Функция fopen() открывает указанный файл и определяет характер операций с файлом. Комбинация wt определяет операцию записи в файл в текстовом режиме. Операция чтения и записи в существующий файл должна быть определена как ‘rt+’. Функция fopen() возвращает указатель, используемый для идентификации потока в последующих операциях.

Для реализации определённых в задании действий используется оператор цикла типа do…while. Условие while(getche()=='y') означает, что цикл будет выполняться до тех пор, пока функция getche() возвращает значение ‘y’. Функция getche() отличается от getch() тем, что выводит символ нажатой клавиши.

Тело цикла содержит объявление локальных (действующих только внутри цикла) переменных i, n. Вводится строка символов с экрана в string с помощью функции getline() (смотри лаб. №4).

Введённая строка выводится в файл F1 функцией fputs(). Функция fputs() – это функция вывода строки в файловый поток. Символ перехода на новую строку не добавляется, и завершающий строку нуль-символ в файловый поток не копируется. Для того, чтобы в файле F2.txt числа не были расположены в одну строку, необходимо осуществлять переход на следующую строку. Для этого используем функцию putc(), которая выводит символ в файловый поток. В нашем случае это символ с кодом 13 (код перевода строки ‘ENTER’).

Далее введённая строка string анализируется с целью определения количества слов. Эти действия выполняются в цикле while до тех пор, пока не будет достигнут конец строки (то есть символ ‘\0’).

В условном операторе if проверяется следующее условие: если i-ый символ строки не равен пробелу и точке с запятой, и точке,(т.е. не является символом-разделителем) и (i+1)ый символ строки равен

пробелу или точке с запятой, или точке, или символу конца строки (т.е. хотя бы одному символу-разделителю), то счетчик числа слов (n) увеличивается на 1 (n++). Переменная i определяет позицию символа в строке.

Функция putc() выводит в файл F2 символьное представление количества слов (в виде одной цифры) в строке. Известно, что ASCII код цифр – это коды от 48 (цифра 0) до 57 (цифра 9). Поэтому в операторе putc() к элементу n необходимо прибавить 48 тем самым перевести числовое представление количества слов в символьное:

```
putc((n+48),F2);
```

Но если количество слов больше 9 (т.е. в виде двух цифр), то надо выводить отдельно каждую цифру числа (десятки и единицы), для этого используем оператор if и функцию fmod():

```
if (n>9)
  {
   putc((n/10+48),F2);            записывает десятки в F2
   putc((fmod(n,10)+48),F2);   записывает единицы в F2
  }
else
  putc((n+48),F2);    записывает единицы в F2
```

Для работы с файлами имеется достаточно много других функций. Во многом они подобны известным функциям языка Pascal.

По окончании работы с файлами их необходимо закрыть, используя функции fclose(F1) и fclose(F2). При этом все буферы, связанные с потоком, освобождаются.

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

6. Контрольные вопросы

1. В чём отличие двоичного файла от текстового?
2. Объясните термин «потоковый ввод-вывод».
3. Какие потоки ввода-вывода считаются стандартными?

4. Что возвращает следующие функции ввода-вывода: getc(), getchar(), putc(), putchar()?
5. Что возвращает следующие функции ввода-вывода: gets(),puts()?
6. Что понимается под форматированным вводом-выводом? Какие функции при этом используются?
7. Что определяет спецификатор формата при форматированном вводе-выводе? Приведите примеры спецификации формата.
8. В чём назначение функции fopen() ? Что возвращает функция fopen()?
9. Перечислите режимы работы с файлом.
10. В чём назначение функции fclose()? Что возвращает эта функция?
11. Приведите формат и пример записи функций, осуществляющих ввод-вывод символов в файловый поток.
12. Приведите формат и пример записи функций, осуществляющих ввод-вывод строк в файловый поток.
13. В чём назначение функции fseek() ? Какие параметры необходимо определить для её выполнения?

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\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?

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).

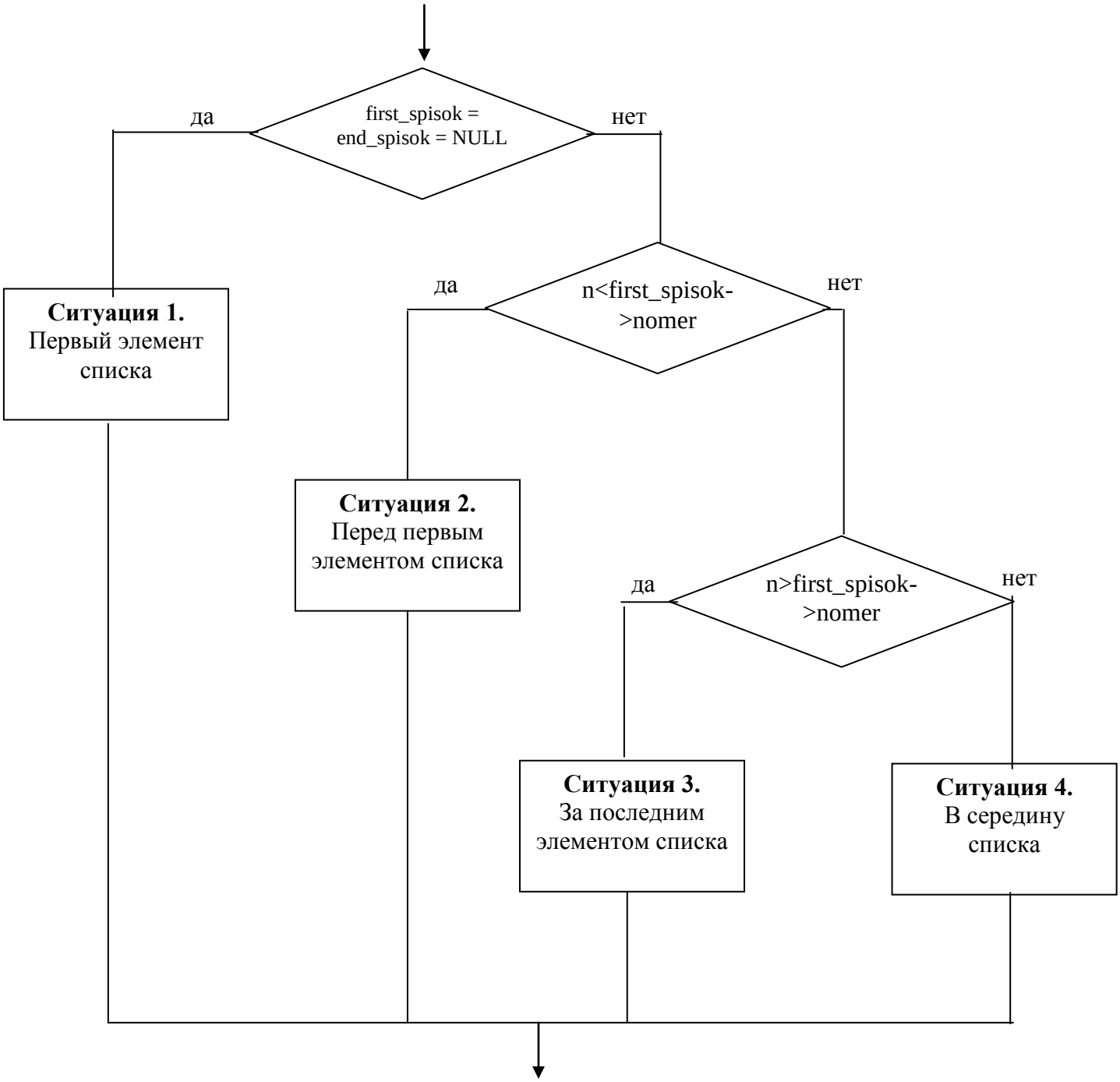
Наиболее сложные действия приходится выполнять при добавлении и исключении элемента списка.

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
```

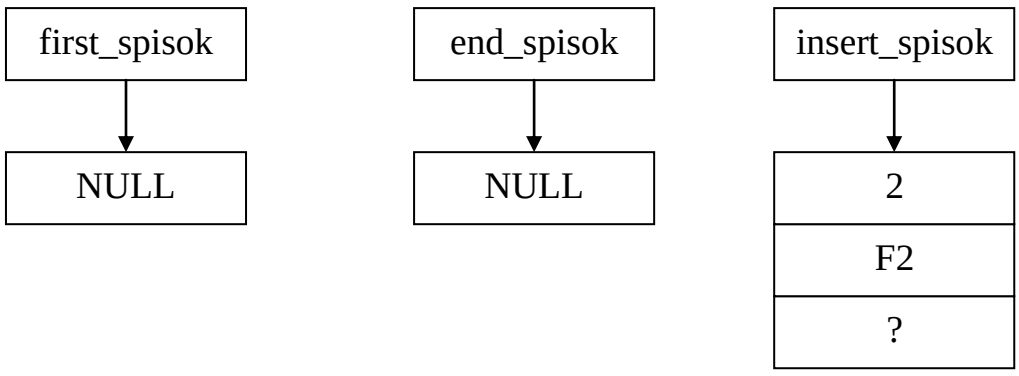
```
        char FIO[10];              //ФИО студента
        SPISOK *next_adres; //указатель на следующий элемент
списка
      };
```

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

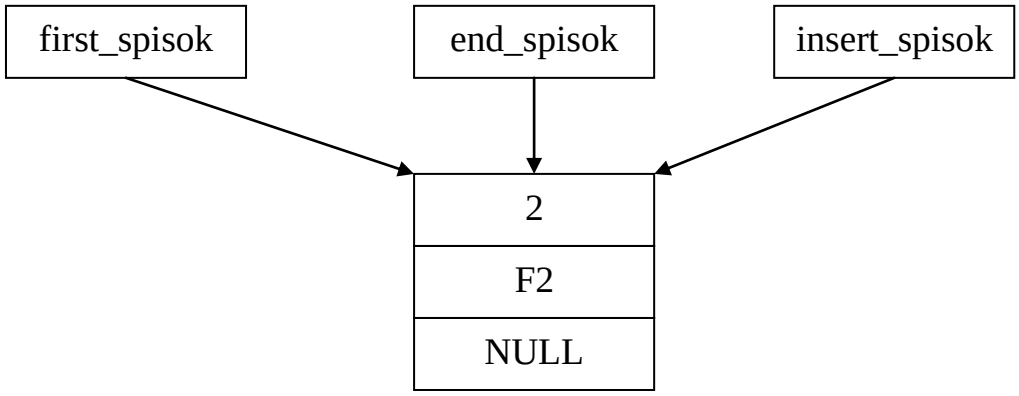


Графически ситуацию можно отобразить следующим образом.
Ситуация 1.

Исходное состояние:



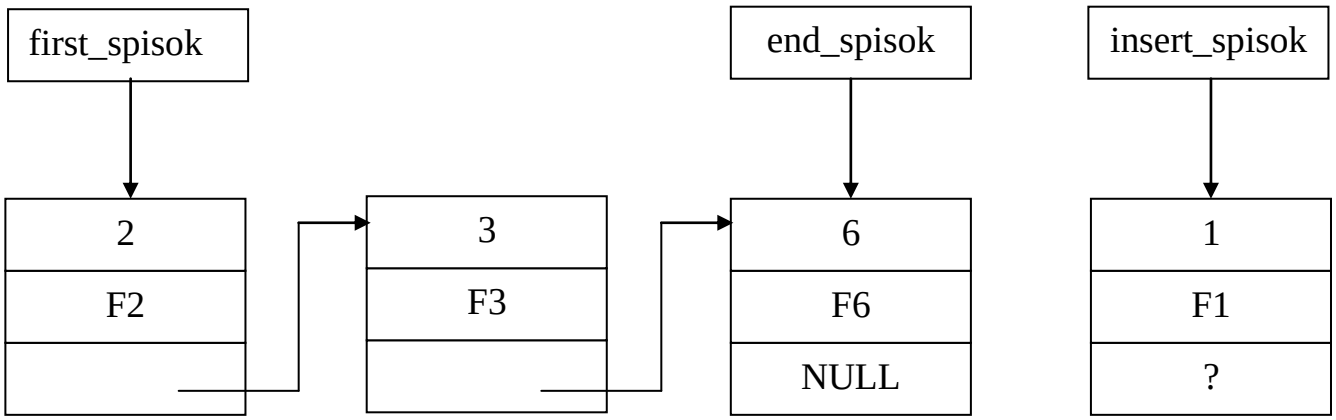
Конечное состояние:



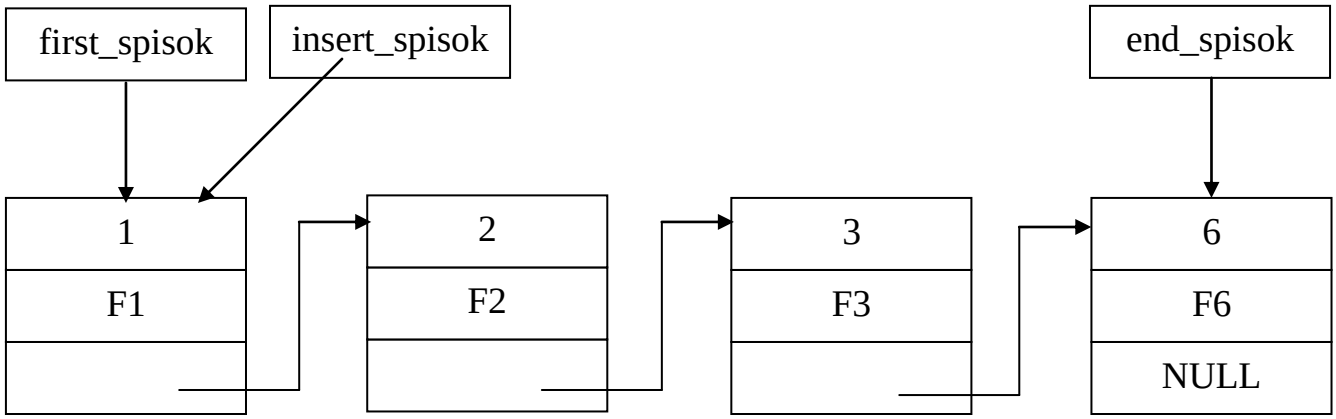
Необходимо: в указатели first_spisok и end_spisok записать адрес, содержащийся в указателе insert_spisok. В адресное поле элемента записать NULL.

Ситуация 2.

Исходное состояние:



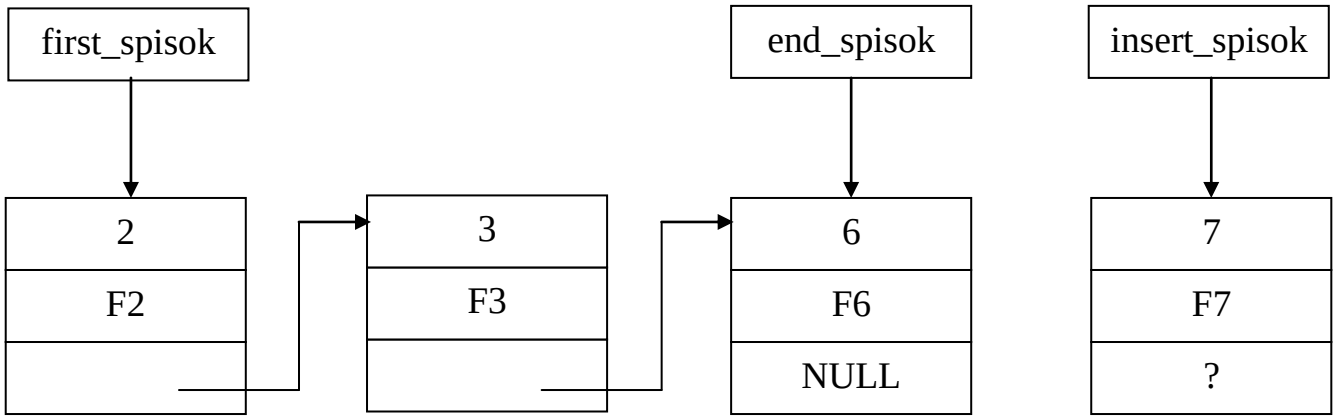
Конечное состояние:



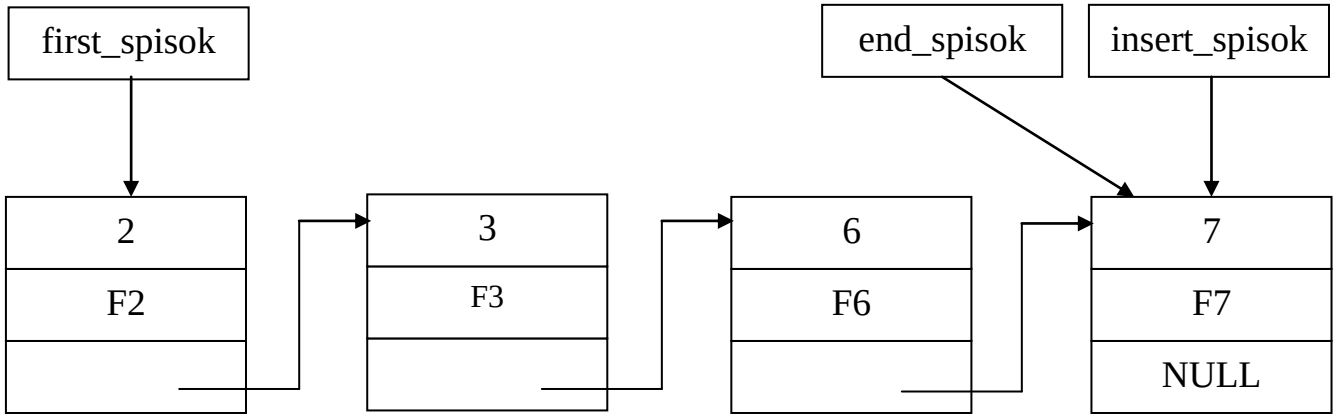
Необходимо: в поле адреса нового элемента записать содержимое указателя first_spisok, присвоить указателю first_spisok значение указателя insert_spisok.

Ситуация 3.

Исходное состояние:



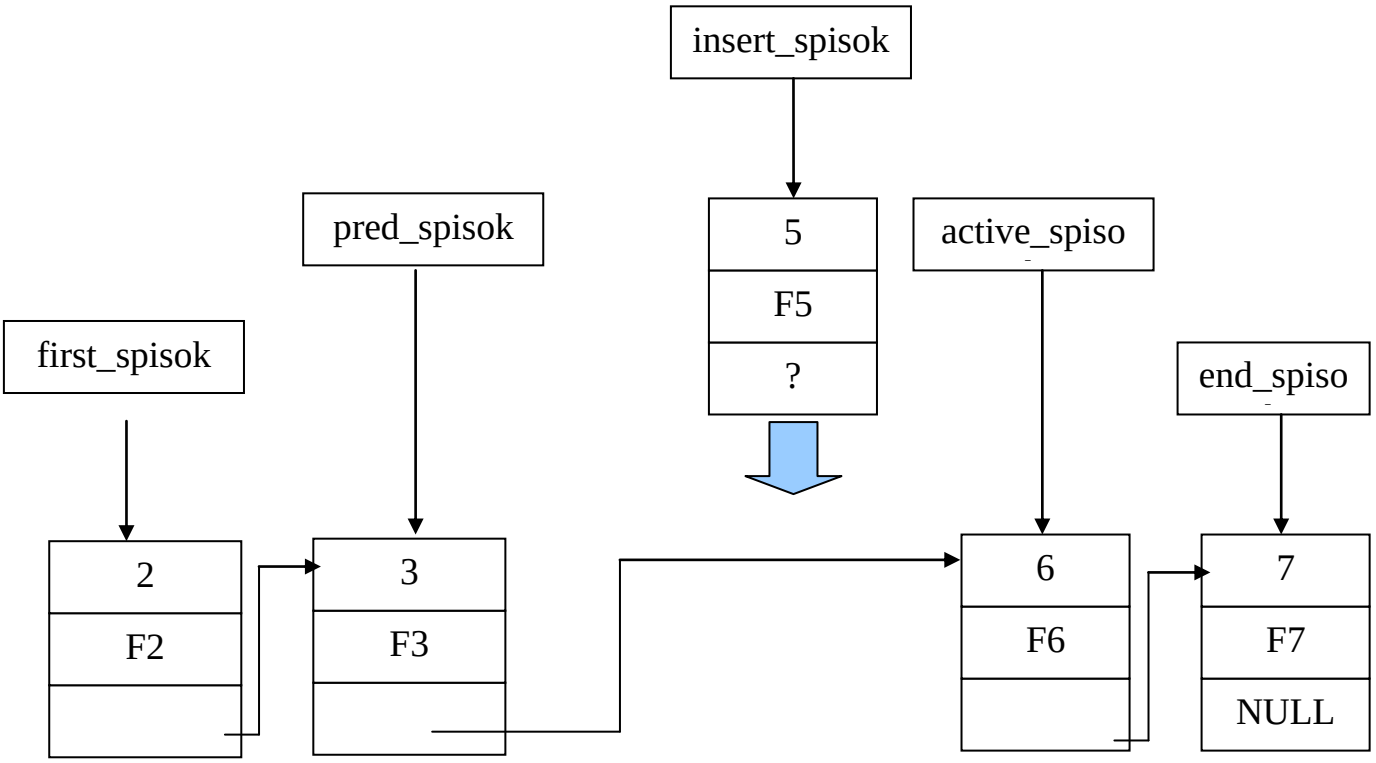
Конечное состояние:



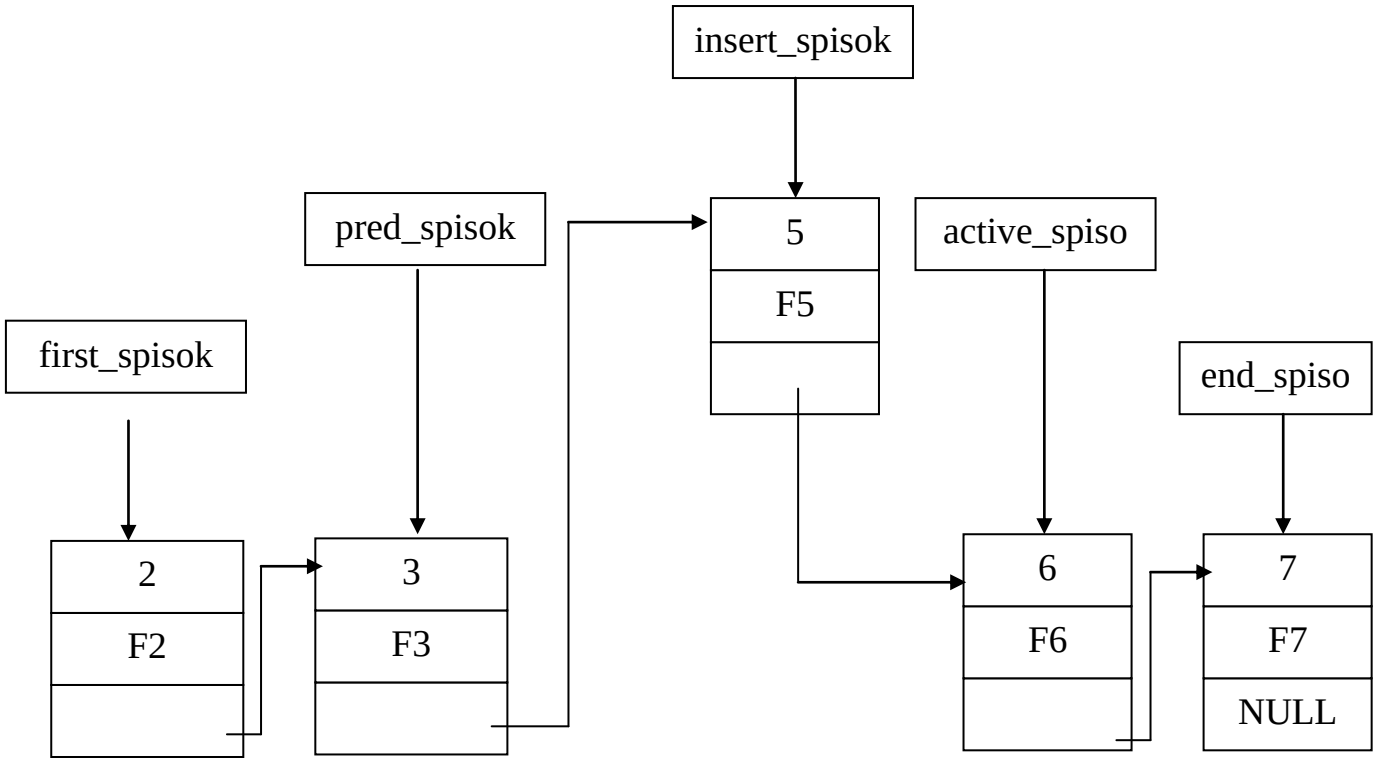
Необходимо: в поле адреса последнего элемента записать вместо NULL содержимое указателя insert_spisok; указателю end_spisok присвоить значение insert_spisok; в поле адреса нового (последнего) элемента записать NULL.

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.