Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Кафедра ЭВМ

Лабораторный практикум
по дисциплине «Основы алгоритмизации и программирования»
для студентов всех форм обучения в 2-ч частях
Часть 2

Минск – 2006

Л а б о р а т о р н а я р а б о т а №1

Структуры

Цель работы: Разработать алгоритмы и написать программы для работы со структурами

Краткие теоретические сведения

Структура – это одна или несколько переменных, возможно различных типов, которые для удобства работы с ними сгруппированы под одним именем. В некоторых языках структуры называются записями. Структуры удобно использовать для организации сложных данных, так как позволяют группу связанных между собой переменных трактовать не как множество отдельных элементов, а как единое целое. Например, данные: имя, фамилия, год рождения, оценки по экзаменам можно поместить в структуру данных о студенте.

Структуры могут копироваться, над ними могут выполняться операции присваивания, взятия адреса с помощью &, осуществление доступа к ее элементам, их можно передавать функциям в качестве аргументов, а функции могут возвращать их в качестве результатов Структуры нельзя сравнивать. Структура отличается от массива тем, что к ее элементам необходимо обращаться по имени.

Объявление структуры начинается с ключевого слова struct, за которым следуют ее тип и список элементов, заключенных в фигурные скобки , например:

```
struct date {int day,    // объявление записи типа date
             month,
             year;};
```

Структуры могут также включать в себя другие структуры, при необходимости можно объявить массив структур, например:

```
struct person { char fam[30],
                     im[20],
                     otch[40];
                int weight,
                   height;
                struct date birthday; } bguir[20], *sptr=&bguir[0];
```

Здесь объявлен массив структур типа person с именем bguir, содержащий данные о студентах: фамилию, имя, отчество, вес, рост и дату рождения. Последняя уже является структурой типа date, которая объявлена выше.

Существует два способа обращения к элементам структуры: прямой доступ к элементу (оператор “.”) и доступ по указателю (оператор “->”). Например:

```
bguir[10].weight=60;  //  Используя прямой доступ к элементу,
                      //  присваиваем зхначение выбранной
                      //  переменной.
strcpy(bguir[10].fam, “Иванов”);
```

В предыдущем примере в объявлении массива структур bguir[] мы дополнительно объявили указатель sptr, которому присвоили адрес начала массива структур. Предыдущий пример иначе можно реализовать так:

```
(sptr+10)->weight=60;          //  Используя доступ по указателю
                               //  на структуру, присваиваем
                               //  значение выбранной переменной.
strcpy((sptr+10)->fam, “Иванов”);
```

Порядок выполнения работы

1. Изучить краткие теоретические сведения.
2. Составить блок-схему алгоритма.
3. По разработанной блок-схеме алгоритма написать программу.
4. Отладить и выполнить программу.

1. В головном модуле ввести структуры. Элементами структуры является вторая структура (фамилия, имя, отчество студента и номер семестра) и объединение. В объединение, в зависимости от семестра, записаны номера экзаменов и результаты, а список наименований экзаменов хранится отдельно. В зависимости от запроса в командной строке вывести информацию о студентах за определенный семестр.

2. В головном модуле в зависимости от запроса в командной строке выполнить ввод структуры или ее обработку в зависимости от ключа.

Ключи:

/e – ввод структуры;

/p – распечатать структуру;

/x – поиск максимальной средней оценки;

/n – поиск минимальной средней оценки.

Структура включает:

а) фамилию студента, оценки;

б) номер группы, номер семестра;

в) оценки за экзамены. Список экзаменов зависит от номера семестра.

3. Для каждого факультета по каждой специальности подготовить приказы об окончании студентами высшего учебного заведения. Обработку информации по факультету прекратить, когда на запрос введено слово «Выход». Специальностей на факультете не более пяти, групп на специальности не более десяти, студентов в группе не более тридцати, количество сданных экзаменов не более двадцати пяти. Списки студентов по специальности вводятся по группам в алфавитном порядке.

В приказе первыми в алфавитном порядке следуют выпускники, получающие диплом с отличием (нет троек, четверок менее 25%), затем в алфавитном порядке следуют остальные выпускники данной специальности. Исходные массивы с данными сохранить, но выходных массивов, копирующих исходные данные, не создавать. Глобальные переменные, системные функции, кроме ввода/вывода, не использовать. Ввод данных, вывод, поиск отличников, расположение по алфавиту выполнить в отдельных функциях.

4. В массиве структур хранится информация о детях детского сада. Элементом структуры является объединение, в котором хранится следующая информация: последнее заболевание ребёнка (грипп, ангина и т.д.) и имя участкового врача. Если ребенок находился в больнице, то и номер больницы, ее адрес и фамилия лечащего врача в больнице. По запросу из командной строки вывести в алфавитном порядке фамилии детей, болевших данным заболеванием.

5. Обработать информацию о фирмах городов. Обработку прекратить, когда на запрос будет введено слово «end». Фирм в городе не более пятидесяти. Информация следующая:

- название фирмы (не более тридцати знаков);
- величина налогообложения (не более 1 млн. р. – в виде строки);
- дата (месяц – в виде строки) последнего срока внесения налога;
- дата его фактического внесения (строка).

В одной функции внести названия фирм, в другой – величину налога, в третьей – предельную дату внесения налога и дату, когда налог погашен (если не внесен, то вводится нуль). В головном модуле для заданной даты (месяц) вывести в алфавитном порядке пять фирм, имеющих максимальную задолженность. Глобальные переменные, системные функции, кроме функций ввода-вывода, не использовать. Исходный массив сохранить, новых массивов структур не создавать. Можно объявлять и вводить другую необходимую информацию.

6. В головном модуле вводится список сотрудников (не более 25): фамилия, имя, отчество и зарплата в виде чисел по месяцам (не более

12) для каждого сотрудника. В первой функции рассортировать список каждого отдела по алфавиту. Во второй функции для каждого отдела создать матрицу из полученного списка сотрудников. Строка матрицы соответствует одному сотруднику: сначала идет фамилия и инициалы, а затем зарплата по месяцам. Полученную матрицу вывести на экран в головном модуле. Глобальные переменные, системные функции, за исключением ввода-вывода, не использовать.

7. Реализовать программу премирования по каждому факультету десяти студентов, которые учатся на «4» и «5» и имеют наивысший средний балл. Обработку данных прекратить после того, как введено слово «вывод». Специальностей на факультете не более пяти, групп на каждой специальности не более десяти, студентов в группе не более тридцати, экзаменов не более пяти. Списки вводятся по алфавиту, кодировка оценок: 0 – неявка, 1 – недопуск, 2,3,4,5 – остальные. Список к оплате для каждого факультета вывести в алфавитном порядке. Новых списков не составлять. Ввод, вывод, поиск данных реализовать в отдельных функциях. Исходный список сохранить.

Л а б о р а т о р н а я р а б о т а №2

Объединения. Поля бит.

Цель работы: Изучить типы данных, основные операции языка С и форматированные функции ввода/вывода. Рассмотреть структуру простой программы.

Теоретические сведения

Объединение – это переменная, которая может содержать (в разные моменты времени) объекты различных типов и размеров. Объединенияпозволяют хранить разнородные данные в одной и той же области памяти. Синтаксис объединения аналогичен синтаксису структур.

```
union example {   int I;
                  float f;
                  char ch[10];
              };
```

Переменная example должна быть достаточно большой, чтобы в ней поместилась любая переменная из указанных трех типов. Значение одного их этих трех типов может быть присвоено переменной example и

далее использовано в выражениях, если тип взятого значения совпадает с типом последнего присвоенного ей значения.

Доступ к элементам объединений такой же, как и для элементов структур.

Фактически объединение – это структура, все элементы которой имеют нулевое смещение относительно ее базового адреса и размер которой позволяет поместиться в ней самому большому ее элементу. К объединением применяются те же операции, что и к структурам.

Поле битов – это группа соседних двоичных разрядов (бит), расположенных в области памяти переменной целого типа.. Использование данных полей делает возможным доступ к отдельным битам более крупных объектов, например байтов или слов. Поля бит удобно использовать тогда, когда для хранения информации в структуре данных достаточно нескольких бит.

Общий синтаксис описания битового поля :

тип [имя] : ширина;

В C++ для определения битового поля разрешено использовать любой тип, интерпретируемый как целый: char, short, int, long (signed или unsigned), перечисления. В полях типа signed крайний левый бит является знаковым. Каждому полю выделяется точно столько бит, сколько указано в поле «ширина». Ссылка на битовое поле выполняется по имени, указанному в поле «имя». Если имя в данном поле не указано,то запрошенное количество бит будет выделено, но доступ к ним будет невозможен.

Порядок выполнения работы

1. Изучить краткие теоретические сведения.
2. Составить блок-схему алгоритма.
3. По разработанной блок-схеме алгоритма написать программу.
4. Отладить и выполнить программу.

Варианты заданий

1. В головном модуле ввести структуры. Элементами структуры являются вторая структура (фамилия, имя, отчество студента и номер семестра) и объединение. В объединении в зависимости от семестразаписаны номера экзаменов и результаты, а список наименований экзаменов хранится отдельно. В зависимости от запроса в командной строке вывести информацию о студентах за определенный семестр.

2. В командной строке задаются два признака. Первый признак определяет тип вводимой информации: просто структура или структура, включающая объединение и имена функций, которые надо выполнить. Функции выводят информацию (из просто структуры или из структуры, включающей объединение), в зависимости от второго признака в командной строке.

В структурах хранится информация о студентах: фамилия, имя, отчество.

В объединении структура хранит один из типов информации:

а) отец, мать, брат;

б) отец, мать, брат, сестра;

в) мать, брат, сестра.

3. В структурах хранится информация о студентах. Одним из элементов структуры является объединение, в котором, в зависимости от места жительства (иногородний студент или нет), информация задается в виде:

а) Минск, ул. …, д. …, кв. …;

б) область, город …, ул. …, д. …, кв. …;

в) область …, район …, город …, ул. …, д. …, кв. …;

г) область …, район …, деревня …, дом … .

По запросу из командной строки, в зависимости от вида информации, вывести данные о тех или иных студентах, используя указатели на функции.

4. Имеется массив структур с информацией о студентах. В структуре в качестве подструктуры задаются фамилия, имя, отчество студентов. Их медицинские параметры задаются в виде объединения в этой же структуре. По запросу из командной строки выдать информацию о студентах с соответствующими признаками.

Объединения включают:

а) рост, вес;

б) рост, вес, два-три других параметра.

Л а б о р а т о р н а я р а б о т а №3

Динамические структуры данных. Стек, очередь, кольцо.

Цель работы: Изучить принципы построения динамических структур данных. Организация данных в виде стека, очереди, кольца. Разработать алгоритмы и написать программы для работы со стеком, очередью, кольцом.

Теоретические сведения

Список – совокупность объектов (элементов списка), размещаемых в динамической памяти, в которой каждый объект содержит

информацию о местоположении связанного с ним другого объекта. В простейшем случае элемент списка представляет собой структурную переменную, содержащую указатель (указатели) на следующий элемент и любое число других полей (информационных).

Списки бывают линейными и кольцевыми, односвязными и двусвязными.

Список называется односвязным (однонаправленным), если движение от элемента к элементу списка возможно только в одном из направлений. Список имеет начальную точку такого движения, элемент такого списка включает только указатель на следующий элемент.

В случае двусвязного (двунаправленного) списка возможно движение от элемента к элементу в обоих направлениях, при этом элемент содержит два указателя: на предыдущий и последующий элементы списка.

В кольцевом списке последний элемент содержит указатель на первый элемент списка.

Самыми распространенными случаями линейного односвязного списка являются стек и очередь.

Очередь – список, который организован так, что новые элементы добавляются в конец списка, а извлекаются из начала, то есть организованный по принципу «первым пришел – первым вышел» (FIFO – First In, First Out). Чтобы не ограничивать максимальное число элементов в очереди, целесообразно строить ее в виде односвязного списка. Добавление новых элементов происходит всегда в начало списка (в «голову»). Последний элемент помечается особым образом, например поле указателя на следующий элемент равно NULL. Извлечение элемента из очереди требует прохода списка для поиска последнего элемента, который и удаляется из списка.

Стек – список, организованный так, что новые элементы добавляются в начало списка и извлекаются для обработки тоже из начала списка, то есть по принципу «первым пришел – последним вышел» (FILO – First In, Last Out), т.е. первый занесенный в стек элемент будет обработан последним.

```
// однонаправленная очередь
#include <stdio.h>
#include <alloc.h>
#include <conio.h>
#include <string.h>
struct zap {char inf[50];
        struct zap *nx;};
void add(zap **);
void del(zap **);
```

```
void del_any(zap **,char *);
void see(zap *);
void sort(zap **);
zap* sort2(zap *);

void main(void)
{ zap *h,*t;
  char l,*st;
  st=(char *)malloc(10);
  h=NULL;
  clrscr();
  while(1)
  { puts("вид операции: 1-создать очередь");
    puts("          2-вывод содержимого очереди");
    puts("          3-удаление элемента из очереди");
    puts("          4-удаление любого элемента из очереди");
    puts("          5-сортировка очереди");
    puts("          0-окончить");
    fflush(stdin);
    switch(getch())
    { case '1': add(&h); break;
      case '2': see(h); break;
      case '3': if(h) del(&h); break;
      case '4': if(h)
                { fflush(stdin);
                  gets(st);
                  del_any(&h,st);
                } break;
      case '5': if (h) //sort2(h); break;
                       sort(&h); break;
      case '0': return;
      default: printf("Ошибка, повторите \n");
    }
  }
}

// функция создания очереди
void add(zap **h)
{ zap *n;
  puts("Создание очереди \n");
  do
  { if(!(n=(zap *) calloc(1,sizeof(zap))))
    { puts("Нет свободной памяти");
```

```
      return;
    }
    puts("Введите информацию в inf");
    scanf("%s",n->inf);
    if (!*h)      // очерель еще не создана
        *h=n;          // хвост очереди
    else
    { n->nx=*h;        // очередной элемент очереди
      *h=n;            // передвигаем указатель на хвост
    }
    puts("Продолжить (y/n): ");
    fflush(stdin);
  } while(getch()=='y');
}

// функция вывода содержимого очереди
void see(zap *h)
{ puts("Вывод содержимого очереди \n");
  if (!h)
  { puts("Очередь пуста");
    return;
  }
  do
  { printf("%s\n",h->inf);
    h=h->nx;
  } while(h);
  return;
}

// функция удаления первого элемента очереди
void del(zap **h)
{ zap *f,*pr;
  if (!*h)
  { puts("Очередь пуста");
    return;
  }
  if (!(*h)->nx)         // в очереди только один элемент
  { free(*h);
    *h=NULL;             // очередь пуста
    return;
  }
  pr=*h;
  while(pr->nx->nx)
```

```
  pr=pr->nx;
  free(pr->nx);          // удаление первого элемента из очереди
  pr->nx=NULL;
  return;
}

// функция удаления любого элемента очереди
void del_any(zap **h,char *st)
{ zap *f,*pr;
  if (!*h)
  { puts("Очередь пуста");
    return;
  }
  if (!(*h)->nx &&       // в очереди только один элемент
     (!strcmp((*h)->inf,st) || *st=='\0'))
  { free(*h);
    *h=NULL;             // очередь пуста
    return;
  }
  pr=*h;
  f=pr;                  // далее f предыдущий к  pr элемент
  while(pr)
  { if (!strcmp(pr->inf,st)) // найден элемент со строкой  st
    {  if (pr==*h) {*h=pr->nx; free(pr); f=pr=*h;}
       else
       { f->nx=pr->nx;   // обходим элемент  pr
         free(pr);       // удаляем элемент  pr очереди
         pr=f->nx;       // выбор следующего элемента очереди pr
       }
    }
    else {f=pr; pr=pr->nx;}  // переход к следующему элементу
  }
}

// функция сортировки содержимого очереди
void sort(zap **h)
{ zap *s1=*h,*s2,*s3,*s4,*hh=NULL;
  s1=*h;                          // ссылка на хвост очереди
  s4=( zap *) calloc(1,sizeof(zap));
  for(; s1->nx; s1=s1->nx)            // выбор исходного элемента очереди
  { for(s2=s1->nx; s2; s3=s3->nx, s2=s2->nx) // перебор последующих за S1
    { if (s2==s1->nx) s3=s1;          // S3-элемент расположенный перед S2
      if(strcmp(s1->inf,s2->inf)>0)    // найдено новое меньшее значение
```

```
    { if (s1->nx==s2) s1->nx=s2->nx; // S1 и S2 - соседние элементы
      else s3->nx=s2->nx;          // S3 находится между S1  и  S2
      s2->nx=s1;                   // элемент с min становится перед S1
      s4->nx=s2;                   // S4- элемент расположенный перед S1
      s1=s2;                       // новый адрес S1- (после замены S1<->S2)
     }
   }
   if(!hh)
   { hh=s1;    // модификация текущего указателя на хвост очереди
     free(s4);
   }
   s4=s1;
  }
  *h=hh;            // возврат возможно измененного указателя на хвост
  puts("Сортировка выполнена");
}
```

Порядок выполнения работы

1. Изучить краткие теоретические сведения.
2. Составить блок-схему алгоритма.
3. По разработанной блок-схеме алгоритма написать программу. Память под

строки выделять динамически.

4. Отладить и выполнить программу.

Варианты заданий

1. В командной строке выполнить. Одна из функций в качестве параметра получает указатель на очередь, содержащую массив указателей на стеки. Элементами стеков являются структуры. Путем слияния исходных стеков создать результирующий стек, упорядоченный в алфавитном порядке строк.задаются имена функций, которые необходимо

2. Имеется стек, элементами которого являются указатель на очередь и признак упорядочения очереди. Элементом записи очереди является фамилия студента. На экране с задержкой поочередно вывести состояние стека и очередей. При нажатии F1 вывод информации на экран приостанавливается и выполняется следующая операция: если «+» - добавляются элементы в очередь, если знак «-«, то удаляются заданные элементы, «!» - наращивается стек.

3. Реализовать программу учета занятых и свободных мест в общежитии, а также лиц, нуждающихся в общежитии. Одним из элементов созданной очереди является номер этажа, другим – общее количество мест на этаже, количество свободных мест, список лиц, проживающих на этаже.

Если свободных мест нет, то студент ставится в очередь на вселение на соответствующий этаж.

4. Имеется перечень (очередь) автостоянок города. В очереди указываются общее количество мест, количество свободных мест и список номеров машин на этой автостоянке. Обеспечить функционирование автостоянок. По клавише F1: если «+» - появляется новая машина и ставится на свободное место, если «-« - машина убывает. Въезд – выезд со стоянки единственный. По клавише F2 создается новая автостоянка. При нажатии на F1 с клавиатуры задается номер автостоянки, на которую желаем поставить автомобиль. При отсутствии мест машина ставится в очередь ожидания свободного места на соответствующей стоянке.

5. Имеется очередь с информацией о больницах. Элементами очереди являются количество всех мест в больнице, количество свободных мест и указатель на функцию, вычисляющую (условно) расстояние от больного до больницы. Программа ведет учет свободных мест и распределяет больных в ближайшие больницы. Реализовать операции приема и выписки больного: если 0 – больной выписывается, 1 – поступает в ближайшую больницу, задается фамилия больного. По F4 – выдается состояние мест в больнице, поF5 – завершить выполнение программы.

6. Ввести «m», «n» (n<= 200) и строку из n символов (строка оканчивается либо «.», либо»:», либо «;»). В строке среди знаков могут встречаться открывающие и закрывающие круглые, квадратные и фигурные скобки. Проверить, предшествует ли каждая открывающая скобка соответствующей закрывающей, т.е. правильность расстановки скобок во всех m строках.

7. Дана очередь. Одним из ее элементов является указатель на кольцо. Реализовать следующие операции с указанными списками: в заданное кольцо подсоединить новый элемент, удалить элемент, вывести на экран содержимое кольца, удалить элемент из очереди, элементы заданного кольца распределить по другим кольцам.

8. Дано кольцо с одной связью. Элемент кольца содержит указатель на строку с названием прибора и указатель на текст с описанием прибора. На старом месте получить новое кольцо, в котором направление обхода поменять на противоположное. Новые массивы, списки не создавать.

9. Дано кольцо. Элементом кольца является указатель на стек, а элементом стека – указатель на текстовую информацию. Объединить стеки в единую цепочку так, чтобы при вхождении в n-й стек последовательно проходились все последующие стеки.

10. Элементами кольца являются указатель на массив указателей на строки и число, задающее количество строк. Рассортировать строки в массиве в алфавитном порядке, а элементы кольца (после первой сортировки) – в порядке возрастания первой строки массива (в алфавитном порядке).

Лабораторная работа №4

Бинарные деревья

Деревья представляют собой нелинейные структуры, встречающиеся в вычислительных алгоритмах. Существуют различные классы деревьев, наиболее популярны бинарные (двоичные) деревья.

Дерево – это неориентированный связный граф. Дерево можно определить как конечное множество T, состоящее из одного или более узлов, таких, что:

- есть один обозначенный узел, называемый корнем данного дерева;

- остальные узлы содержатся в m>=0 попарно непересекающихся множествах T1,…,Tm, каждое из которых в свою очередь является деревом. Деревья T1,…,Tm называются поддеревьями данного дерева.

В бинарном дереве каждый узел имеет только два поддерева,- левое и правое.

Узел бинарного дерева можно описать в виде структуры:

```
struct node
    {  int key;                    //  ключ
       …                        //  данные
       struct node *left;//  ссылка на левое поддерево (узел)
          struct node *right;  //  ссылка на правое поддерево (узел)
       };
```

Для обхода дерева используют ключи, - специальные переменные,облегчающие доступ к узлам дерева. Это некоторое значение, хранящееся в каждом узле дерева и удовлетворяющее следующим условиям. Например, все ключи в левом поддереветекущего узла меньше ключа в нем, а все ключи в правом поддереве – больше или равны текущему.

При работе с бинарными деревьями возникают задачи создания дерева, добавления или удаления очередного узла, поиска узла с требуемй информацией, вывод информации, содержащейся в дереве.

```
// бинарное дерево
#include <stdio.h>
#include <alloc.h>
#include <string.h>
#include <conio.h>
#include <dos.h>
#define N 20
#define M 20
struct der {char *inf;      // информационное поле*/
```

```
        int n;          // число встреч инф. поля в бинарном дереве*/
      struct der *l,*r;};  // указатель на левое и правое поддерево*/

void see_1(struct der *);
void see_2(struct der *);
der *sozd(struct der *);
void add(struct der *);
der *del(struct der *);

void main(void)
{ der *dr;
  dr=NULL;       // адрес корня бинарного дерева

  clrscr();
  while(1)
  { puts("вид операции: 1-создать дерево");
    puts("          2-рекурсивный вывод содержимого дерева");
    puts("          3-нерукурсивный вывод содержимого дерева");
    puts("          4-добавление элементов в дерево");
    puts("          5-удаление любого элемента из дерева");
    puts("          6-выход");
    fflush(stdin);
    switch(getch())
    { case '1': dr=sozd(dr); break;
      case '2': see_1(dr); getch(); break;
      case '3': see_2(dr); getch(); break;
      case '4': add(dr); break;
      case '5': del(dr); break;
      case '6': return;
    }
    clrscr();
  }
}

// создание бинарного дерева
der *sozd(struct der *dr)
{ if (dr)
  { puts("Бинарное дерево уже создано");
    return (dr);
  }
  if (!(dr=(struct der *) calloc(1,sizeof(struct der))))
  { puts("Нет свободной памяти");
    getch();
```

```
   return (NULL);
  }
  puts("Введите информацию в корень дерева");
  dr->inf=(char *) calloc(1,sizeof(char)*N);
  gets(dr->inf);
  dr->n=1;      // число повторов информации в дереве
  dr->l=NULL;   // ссылка на левый узел
  dr->r=NULL;   // ссылка на правый узел
  return(dr);
}

// функция добавления узлов в бинарное дерево
void add(struct der *dr)
{ struct der *dr1,*dr2;
  char *st;             // строка для анализа информации
  int k;                // результат сравнения двух строк
  int ind;
  if (!dr)
  { puts("Нет корня дерева \n");
    getch();
    return;
  }
  do
  { puts("Введите информацию в очередной узел дерева  (0 - выход)");
    st=(char *) calloc(1,sizeof(char)*N);  //память под символьную инф.
    gets(st);
    if(!*st) return;     // выход в функцию  main
    dr1=dr;
    ind=0;              // 1 - признак выхода из цикла поиска
    do
    { if(!(k=strcmp(st,dr1->inf)))
      { dr1->n++;       // увеличение числа встреч информации узла
        ind=1;          // для выхода из цикла do ... while
      }
      else
      {
        if (k<0)        // введ. строка  <  строки в анализируемом узле
        { if (dr1->l!=NULL) dr1=dr1->l; // считываем новый узел дерева
          else ind=1;   // выход из цикла do ... while
        }
        else            // введ. строка  >  строки в анализируемом узле
        { if (dr1->r!=NULL) dr1=dr1->r; // считываем новый узел дерева
          else ind=1;   // выход из цикла do ... while
```

```
      }
     }
    } while(ind==0);
    if (k!=0)     // не найден узел с аналогичной информацией
    { if ((dr2=(struct der *) malloc(sizeof(struct der)))==NULL)
     { puts("Нет свободной памяти");
       return;
     }
     if (k<0) dr1->l=dr2;   // ссылка в dr1 налево
     else dr1->r=dr2;       // ..... направо
     dr2->inf=st;           // заполнение нового узла  dr2
     dr2->n=1;
     dr2->l=NULL;
     dr2->r=NULL;
    }
//    free(st);
  } while(1);        // любое условие т.к. выход из цикло по return
}

// рекурсивный вывод содержимого бинарного дерева
void see_1(struct der *dr1)
{ if(dr1)
  { printf("узел содержит :   %s , число встреч %d\n",dr1->inf,dr1->n);
    if (dr1->l) see_1(dr1->l);     // вывод левой ветви дерева
    if (dr1->r) see_1(dr1->r);     // вывод правой ветви дерева
  }
}

// нерекурсивный вывод содержимого бинарного дерева
// используя стек для занесения адресов узлов дерева
void see_2(struct der *dr1)
{ struct stek {der *d;
               stek *s;} *st,*st1=NULL;
  int pr=1;
  for(int i=0;i<2;i++)   // в стек заносятся 2 элемента содержащие указатель
на
  { st=(stek *)calloc(1,sizeof(stek));  // корень дерева для прохода по
    st->d=dr1;       // левому и правому поддеревьям
    st->s=st1;       // указатель на стек вниз
    st1=st;
  }
    printf("узел содержит :   %s , число встреч %d\n",dr1->inf,dr1->n);
    while(st)
```

```
   { do
     { if(pr && dr1->l) dr1=dr1->l;    // переход на узел слева
       else if (dr1->r) dr1=dr1->r;    // переход на узел cправа
       pr=1;                    // сброс принудительного движения вправо
       if(dr1->l && dr1->r)          // узел с 2 связями вниз
       { st1=st;
         st=(stek *)calloc(1,sizeof(stek));
         st->d=dr1;                 // указатель на найденый узел
         st->s=st1;                 // указатель на стек вниз
       }
       printf("узел содержит :   %s , число встреч %d\n",dr1->inf,dr1->n);
     } while(dr1->l || dr1->r);
     dr1=st->d;                 // возврат на узел ветвления
     st1=st->s;                 // в стеке адрес узла выше удаляемого
     free(st);                  // удаление из стека указателя на узел
                                // после прохода через него налево
     st=st1;
     if(dr1->r) pr=0;           // признак принудительного перехода на
                                // узел расположенный справа от dr1, т.к.
                                // dr1->inf уже выведен при проходе слева
   }
}


// функция удаления узла дерева
struct der *del(struct der *dr)
{ struct der *dr1,*dr2,*dr3;
  char *st;                 // строка для анализа информации
  int k;                    // результат сравнения двух строк
  int ind;
  if(!dr)
  { puts("Дерево не создано \n");
    getch();
    return NULL;
  }
  puts("Введите информацию в для поиска удаляемого узла");
  st=(char *) malloc(sizeof(char)*N);
  gets(st);                 //строка для поиска узла в дереве
  if(!*st) return NULL;          // выход в функцию  main
  dr2=dr1=dr;
  ind=0;                    // 1 - признак выхода из цикла поиска
  do                 // блок поиска удаляемого из дерева узла
  { if (!(k=strcmp(st,dr1->inf)))
```

```
      ind=1;        // узел со строкой st найден
    if (k<0)        // введ. строка  <  строки в анализируемом узле
    { if (dr1->l!=NULL)
      { dr2=dr1;      // запоминаем текущий узел
        dr1=dr1->l;   // считываем новый левый узел дерева
      }
      else ind=1;     // выход из цикла do ... while
    }
    if (k>0)        // введ. строка  >  строки в анализируемом узле
    { if (dr1->r!=NULL)
      { dr2=dr1;      // запоминаем текущий узел
        dr1=dr1->r;   // считываем новый правый узел дерева
      }
      else ind=1;     // выход из цикла do ... while
    }
  } while(ind==0);
  free(st);
  if (ind==0)
  { puts("Требуемый узел не найден \n");
    getch();
    return dr;
  }
  else
  { k=strcmp(dr1->inf,dr2->inf);
    dr3=dr1;
    if (k<0)          // удаляемая вершина < предыдущей
    { dr3=dr1->r;       // поиск ссылки NULL влево
      while(dr3->l!=NULL) dr3=dr3->l;
      dr2->l=dr1->r;    // сдвиг ветви начинающейся с адреса dr1->r влево
      dr3->l=dr1->l;
    }
    else              // удаляемая вершина > предыдущей
    { dr3=dr1->l;       // поиск ссылки NULL вправо
      while(dr3->r!=NULL) dr3=dr3->r;
      dr2->r=dr1->l;    // сдвиг ветви начинающейся с адреса dr1->l вправо
}
      dr3->r=dr1->r;
    }
  }
}
```

Порядок выполнения работы

1. Изучить краткие теоретические сведения.
2. Составить блок-схему алгоритма.
3. По разработанной блок-схеме алгоритма написать программу. Память под

строки выделять динамически.

4. Отладить и выполнить программу.

Варианты заданий

1. Написать рекурсивную функцию поиска узла бинарного дерева с максимальным значением (число с плавающей точкой). Ключ дерева – число с фиксированной точкой.

2. Дано бинарное дерево. Определить количество узлов на N-м уровне бинарного дерева (N ввести с клавиатуры).

3. Дано бинарное дерево. Записи бинарного дерева содержат указатель на функцию и указатель на стек, хранящий строку знаков. Одна из функций определяет правильность расстановки скобок в строке. Вторая функция переставляет самую длинную и самую короткую строки в соответствующих стеках.

4. Дан массив бинарных деревьев. Элементеми дерева являются фамилия, оценки, адрес, указатель на текст с характеристикой студента, признак – военнообязанный или нет. Сделать копии бинарных деревьев с целью воссоздания информации, если исходные деревья и вся принадлежащая им информация будут уничтожены. Структура новых деревьев должна полностью совпадать со структурами исходных деревьев. Рекурсивные функции и глобальные переменные не использовать.

5. Дана ссылка на бинарное дерево, записи которого содержат указатель на стек, в котором посимвольно хранится строка, являющаяся ключом бинарного дерева. Сформировать кольцо из двадцати пяти записей,в которых находятся первые в алфавитном порядкеключи бинарного дерева (ключ не более десяти знаков). Указатели на стеки, содержащие выбранные строки (ключи бинарного дерева), записать в файл. Имя файла ввести. Рекурсии и системные функции не использовать.

6. Дан стек. Элементами стека являются число N и указатель на бинарное дерево. В узле бинарного дерева записаны указатели на строку и на массив целых чисел. На N-м уровне каждого дерева найти его средний элемент (узел) и , используя рекурсию, на старом месте записать строку знаков в обратном порядке. Рекурсивно вывести строки с хвоста. Узлов на одном уровне не более пятидесяти. Глобальные переменные не использовать.

7. Дано бинарное дерево, в качестве ключа используется строка знаков (длина не более десяти знаков). В записях дерева хранятся

указатели на кольцо. В записях кольца есть указатели на строки знаков (не более ста знаков). Слова в строке разделены произвольным количеством пробелов. Определить частоту встречаемости слов во всех строках кольца. Вывести в файл слова и соответствующую им частоту. Имя файла для каждого кольца задается с клавиатуры. Глобальные переменные, системные функции не использовать, исходные строки сохранить. Самое длинное слово не более восьми знаков.

8. Дано бинарное дерево, в качестве ключа используется строка, ее длина не более пяти знаков. Записи дерева содержат указатель на слово, его длина не более пятнадцати знаков. В поддереве минимальной длины определить самое длинное слово.

Лабораторная работа №5

Файлы. Текстовые файлы.

Цель работы: Изучить структуру текстовых файлов, функции для работы с текстовыми файлами. Разработать алгоритмы и написать программы для работы с текстовыми файлами.

Теоретические сведения

Файл – это именованный объект, хранящий данные (программа или любая другая информация) на каком-либо носителе. Файл, как и массив, - это совокупность данных. В отличии от массивов, файлы располагаются не в оперативной памяти, а на жестких дисках или внешних носителях, файл не имеет фиксированной длины, т.е. может увеличиваться и уменьшаться. Перед работой с файлом его необходимо открыть, а после работы – закрыть..

Различают два вида файлов – текстовые и бинарные.
Текстовые файлы представляют собой совокупность ASCII-символов. Эта последовательность символов разбивается на строки, каждая строка заканчивается двумя кодами:13,10 (0xD,0xA). К текстовым файлам относятся, например, *.bat, *.c, *.asm.

Библиотека языка С содержит функции для работы как с текстовыми, так и с бинарными файлами.

Перед работой с файлом его необходимо открыть. Для этого используется функция fopen(), которая при успешном открытии возвращает указатель на структуру типа FILE, называемый указателем на файл. Эта структура связана с физическим файлом и содержит всю необходимую информацию для работы с ним (указатель на текущую позициюв файле, тип доступа и др.).

Функция открытия файла fopen() содержит два параметра, оба являются строковыми литералами.

FILE *fopen(char *filename, char *mode);

Первый параметр задает физическое местонахождение (путь) и имя открываемого файла, а второй – тип доступа к файлу.
Значения второго параметра могут принимать следующие значения:

«r» - открыть файл для чтения.

«w» - открыть файл для записи. Если файл существует, то его содержимое теряется.

«a» - открыть файл для записи в конец файла. Если файл не существует, то он создается.

«r+» - открыть файл для чтения и записи. Файл должен существовать.

«w+» - открыть файл для чтения и записи. Если файл существует, то его содержимое теряется.

«a+» - открыть файл для чтения и записи в конец файла. Если файл не существует, то он создается.

К перечисленным комбинациям могут быть добавлены также «t» либо «b»:

«t» - открыть файл в текстовом режиме.

«b» - открыть файл в бинаргом режиме.

Возможны следующие режимы доступа: «w+b», «wb+», «rw+», «w+t», «rt+» и др. Если режим не указан, то по умолчанию файл открывается в текстовом режиме.

После работы с файлом он должен быть закрыт функцией fclose(). Для этого необходимо в указанную функцию передать указатель на FILE, который был получен при открытии функцией fopen(). При завершении программы незакрытые файлы автоматически закрываются системой. Последовательность операторов для открытия и закрытия файлов следующая:

```
#include<stdio.h>
…
FILE *f;
If(!(f=fopen(“readme.txt”,”r+t”)))
{
      printf(“Невозможно открыть файл\n”); return;
}
…   //Работа с файлом
fclose(f);
```

Рассмотрим функции. Которые использует язык С для работы с текстовыми файлами. Для записи в файл и чтения из файла

используются функции fprintf(), fscanf(), fgets(), fputs(). Отличие от функций printf(), scanf(), gets(), puts() состоит в том, что для работы с файлами добавлен параметр, который является указателем на структуру FILE.

После открытия каждый файл имеет так называемый указатель на текущую позицию в файле (УТПФ). Все операции над файлами (чтение и запись) работают с данными, на которые указывает УТПФ. При каждом выполнении функции чтения или записи УТПФ смещается на количество записанных или прочитанных байт. Так реализуется *последовательный доступ к данным.*

Для организации чтения или записи данных в произвольном порядке необходимо реализовать установку указателя на некоторую заданную позицию в файле, для этого используется функция fseek().

Int fseek(FILE *stream, long offset, int whence);

Параметр offset задает количество байт, на которое необходимо сместить указатель в направлении, указанном whence. Параметр whence может принимать следующие значения:

SEEK_SET 0 - смещение выполняется от начала файла;
SEEK_CUR 1 – смещение выполняется от текущей позиции указателя;
SEEK_END 2 – смещение выполняется от конца файла.

Величина смещения может быть как положительной, так и отрицательной. Такой доступ к данным в файле называется *произвольным*.

…Рассмотрим пример, реализующий ввод с клавиатуры чисел и добавление их в текстовый файл по возрастанию.

```
#include<stdio.h>
voidmain(void)
{FILE*f;
inta=5,b=7,c=10,d=11,e=14;
inti,j,k,kk;
fpos_tl1,l2;//типпозициявфайле
f=fopen("aaa","w+");
fprintf(f,"%3d%3d%3d%3d%3d",a,b,c,d,e);//форматзаписи:2пробела1цифра
while(1)
{scanf("%d",&i);
if(i==999)break;//выходизпрограммы
rewind(f);    //установкаУТПФвначалофайлаисброс
do//признакаконцафайла
{fgetpos(f,&l1);//УТПФнаначалополясчитываемогочисла
fscanf(f,"%d",&j);
if(feof(f)||j>i)break;//достигнутконецфайлаилисчитаночисло
```

```
                    //большеечемвведенноесклавиатуры.
}while(1);
if(feof(f)&&j<i)//EOFивфайленетчисла>чемвведенное
{rewind(f);
fseek(f,0,2);//выходнаконецфайла
kk=fprintf(f,"%3d",i);//дозаписьвконецфайла
continue;
}
rewind(f);
fseek(f,-3,2);//УТПФнапоследнийэлементфайла
k=1;
do
{fgetpos(f,&l2);//сдвигвсехчиселвмассиведо
fscanf(f,"%d",&j);//l2позицияУТПФчислакоторое>чеми
rewind(f);
l2+=3;
fsetpos(f,&l2);
fprintf(f,"%3d",j);
k++;
fseek(f,-3*k-3,2);
}while(l1!=l2);
fsetpos(f,&l2);
fprintf(f,"%3d",i);//записьiнаместочисласкоптороого
}    //произведенсдвигвсехчиселвниз
fclose(f);
}
```

Порядок выполнения работы

1. Изучить краткие теоретические сведения.
2. Составить блок-схему алгоритма.
3. По разработанной блок-схеме алгоритма написать программу.
4. Отладить и выполнить программу.

Варианты заданий

1. С клавиатуры вводится имя файла. Запись файла содержит ФИО и место жительства студента. Записать файл в память. Рассортировать записи в алфавитном порядке:
- в новый файл;
- в этом же файле.

Осуществить поиск информации об указанном студенте в файле.

2. С клавиатуры вводится имя файла, содержащего информацию о жителях района (ФИО, жилая площадь и количество человек). Написать программу, которая:
- сортирует файл по величине жилой площади, приходящейся на одного человека;
- добавляет, удаляет и выводит на экран информацию файла.

3. В записях файла хранятся наименование магазина, его номер, адрес, а также имя файла, в котором хранится информация об

имеющихся в нем книгах: количество, стоимость, издательство. Программа реализует следующие операции: выдачу справок, продажи,поступления новых книг, а также поиск требуемых книг.

4. Имя текстового файла задано в командной строке. Файл содержит не более десяти строк, каждая строка не превышает восьмидесяти символов. В тексте встречаются числа, все они записаны в восьмеричной системе счисления. Преобразовать эти числа в десятичную систему счисления и записать отредактированный текст обратно в файл.

5. В командной строке даны имена двух символьных файлов. Первый файл содержит произвольный текст, слова в тексте разделены пробелами и знаками препинания. Второй файл содержит не более сорока пар слов, разделенных запятыми. Эти слова образуют пары: каждое первое слово считается заменяемым, а второе – заменяющим. Найти в первом файле все заменяемые слова и заменить их на соответствующие заменяющие слова. Результат поместить во второй файл.

6. Создать файл, в котором хранится информация о видеофильмах (название, тип фильма, год выпуска). Программа реализует следующие операции:

- добавляет в файл новую информацию;
- исключает информацию о заданных фильмах;
- выводит список фильмов соответствующего типа.

7. Создать файл. В записи файла хранятся фамилия студента и список взятых им книг (не более двадцати). Студентов не более тридцати. Вывести список книг, взятых указанным студентом.

8. Программа ведет учет лекарств в аптеке:

- поступление новых лекарств;
- выдача лекарств, если они имеются;
- дата изготовления и контроль срока хранения лекарств;
- вывод на экран и удаление лекарств, срок хранения которых уже истек.

9. В файле, имя которого задано в командной строке, хранится текст. Длина строки в тексте не превышает восьмидесяти символов. Если в тексте встречаются числа, то их необходимо вывести на экран, а файл уплотнить.

10. Вычислить значения арифметических выражений, записанных в файл. Имя переменной – одна буква. Значения переменных в виде A=3,1415, B=-37,2 и т.д. записаны во второй файл. Имена файлов задаются в командной строке. В головном модуле вывести само выражение и его вычисленное значение.

Лабораторная работа №6

Бинарные файлы.

Цель работы: Изучить структуру бинарных файлов, функции для работы с бинарными файлами. Разработать алгоритмы и написать программы для работы с бинарными файлами.

Теоретические сведения

Бинарные файлы – это файлы, которые не имеют структуры текстовых файлов. Каждая программа для своих бинарных файлов определяет собственную структуру.

Для чтения и записи в бинарный файл используются функции fwrite(), fread()/ эти функции без каких-либо изменений копируют блок данных из оперативной памяти в файл и из файла в оперативную память соответственно. Такой способ записи – чтения требует меньших затрат времени, и его целесообразно использовать и для текстовых файлов, если работать с блоками информации. Функции имеют следующий формат:

```
unsigned fread(void *ptr, unsigned size, unsigned n, FILE *stream)
unsigned fwrite(void *ptr, unsigned size, unsigned n, FILE *stream)
```

здесь

sze – размер блока информации в байтах;

n – количество блоков;

*stream – указатель на структуру FILE открытого файла.

Первым параметром передается указатель на буфер, в который будут помещены данные из файла функцией fread() или из которого данные будут прочитаны в файл функцией fwrite().

Нужно отметить, что нет способа различать бинарные и текстовые файлы, любой бинарный файл можно открыть для работы как текстовый и наоборот. Это может привести к ошибкам, следовательно необходимо работать с файлом в том режиме, в котором он был открыт.

Язык С обеспечивает возможность работы с файлами, используя дескрипторы. Дескриптор – это целое число, которое система ставит в соответствие открытому файлу ив дальнейшем использует в операциях работы с файлами. Существует целый ряд функций работы с файлами через дескриптор.

```
// записать в бинарный файл числа и выполнить их сортировку
//метод"черезотбор"
```

```
#include<stdio.h>
voidmain(void)
{FILE*f;
fpos_t*n1,*n2,*n3;
inti1,i2,i3,m[]={3,7,4,1,2,4};
if(!(f=fopen("aa","w+b")))
{puts("файлнеможетбытьсоздан");
return;
}
fwrite(m,sizeof(int),6,f);
rewind(f);
while(1)
{fgetpos(f,n1);//адресисходногочисла
fread(&i1,sizeof(int),1,f);//исходноечисло
if(feof(f))break;
*n3=*n1;i3=i1;//дляконтролянайденолиновое min значение
while(1)
{fgetpos(f,n2);//позициясчитываемогочисла
fread(&i2,sizeof(int),1,f);//значениечиславыбранногодлясравнения
if(feof(f))break;
if(i3>i2)//найденоновое(меньшее)значение
{*n3=*n2;
    i3=i2;
}
}
if(*n1!=*n3)
{fsetpos(f,n3);//
fwrite(&i1,sizeof(int),1,f);
fsetpos(f,n1);//
fwrite(&i3,sizeof(int),1,f);
}
*n1+=sizeof(int);
fsetpos(f,n1);//
}
rewind(f);
fread(m,sizeof(m),1,f);
for(i1=0;i1<sizeof(m)/sizeof(int);printf("%3d",m[i1++]));
fclose(f);
}
```

Порядок выполнения работы

1. Изучить краткие теоретические сведения.
2. Составить блок-схему алгоритма.
3. По разработанной блок-схеме алгоритма написать программу.
4. Отладить и выполнить программу.

Варианты заданий

1. В двоичном файле указан размер квадратной матрицы, а затем построчно записаны элементы матрицы целых чисел. Таких групп данных в файле несколько. Вычислить сумму элементов, расположенных ниже главной диагонали. Результат записать в коней файла. Имя файла задается в командной строке.

2. Удалить из бинарного файла чисел, содержащиеся в текстовом файле. Имена файлов получить из командной строки. В файлах содержится числовая информация.

3. Имеются два бинарных файла, упорядоченных по возрастанию. Переписать информацию в третий файл, упорядочивая ее по убыванию. Дополнительных массивов и файлов не использовать, сортировку не выполнять.

4. Из текстового файла удалить числа, содержащиеся в бинарном файле.

5. Выполнить сортировку бинарного файла методом вставок. Дополнительных массивов и файлов не использовать.

6. Для бинарного файла выполнить сортировку длинных целых чисел методом «через отбор». Дополнительных массивов и файлов не использовать.

7. В бинарном файле выполнить реверсивный переворот содержащихся в нем целых чисел. Дополнительных массивов и файлов не использовать.

8. В бинарном числовом файле выполнить перестановку местами *1* и *2*, *3* и *4* и т.д. элементов. Дополнительных массивов и файлов не использовать.

9. Из командной строки получить имя бинарного файла и границы диапазона чисел, которые нужно удалить из файла. Дополнительных массивов и файлов не использовать.

10. Для существующего упорядоченного бинарного файла ввести с клавиатуры несколько чисел, размещая их в файле без нарушения его упорядоченности по возрастанию. Дополнительных массивов и файлов не использовать.

Литература

1. Керниган Б., Ритчи Д. Фьэр А. Язык программирования С*(*С++*)*. Задачи по языку С*(*С++*)*. –М.*:* Финансы и статистика, *1985*.

2. Керниган Б., Ритчи Д. Язык программирования С*(*С++*)*. –М.*:* Финансы и статистика, *1992*.

3. Юлин В.А., Булатова И.Р. Приглашение к С*(*С++*)*. –Мн.*:* Выш. шк, *1991*.

4 . Касаткин А.И., Вальвачев А.Н. От *Turbo C* к *Borland C++*. Мн.*:* Выш. шк, *1992*.

5. Касаткин А.И. Управление ресурсами. Мн.*:* Выш. шк, *1992*.

6. Касаткин А.И. Системное программирование. Мн.*:* Выш. шк, *1993*.

7. Романовская Л.М., Русс Т.В., Свитковский С.Г. Программирование в среде С*(*С++*)* для ПЭВМ. М.*:* Финансы и статистика, *1992*.

10. Герберт Шилд. Программирование на *Borland C++*. –Мн.*:* ООО *"*Попурри*" 1998*г. *– 800*с.

11. Скляров В.А. Программирование на языках С*(*С++*)*. –М.*:* Высш. шк. *1999*г. *– 288* с.

Содержание