Работа № 2

Записи с вариантами. Обработка таблиц

Цель работы – приобрести навыки работы с типом данных «запись» (структура), содержащим вариантную часть (объединение, смесь), и с данными, хранящимися в таблицах, произвести сравнительный анализ реализации алгоритмов сортировки и поиска информации в таблицах, при использовании записей с большим числом полей, и тех же алгоритмов, при использовании таблицы ключей; оценить эффективность программы по времени и по используемому объему памяти при использовании различных структур и эффективность использования различных алгоритмов сортировок.

Краткие теоретические сведения

Параметры реальных объектов бывает сложно описать, используя только числовые, либо только символьные типы данных. Часто требуется комбинация разных типов. Для этого используется тип «запись» (структура). Запись – это структурированный тип, состоящий из фиксированного числа компонентов различного типа. Компоненты записи называются полями записи. Каждое поле имеет свое собственное имя и тип. Тип поля может быть любым, кроме файлового. Обращение к полю записи осуществляется с помощью идентификатора переменной и идентификатора поля, разделенных точкой. Такая структура называется составным именем.

Объем памяти, необходимый для размещения записи, складывается из размера памяти под каждое из ее полей.

Запись может содержать поля, которые, в свою очередь, также являются записями, то есть, возможна структура вложенных записей.

Тип «запись» (**Record**) или структура (**struct)** объявляется следующим образом:

```
Type
  "имя записи1" = Record
     "поле 1" : "тип 1";
     "поле 2" : "тип 2";
   …
     "поле n" : "тип n"
  End;
```

```
struct  "имя записи1"
        {  "тип 1"  "поле 1";
           "тип 2"  "поле 2";
            …
           "тип 3"  "поле 3";
         };
```

Пример вложенной записи:

```
"имя записи2" = Record
```

```
struct  "имя записи2"
```

```
    "поле 1" : "тип 1";                    {  "тип 1"  "поле 1";
    "поле 2" : "тип 2";                       "тип 2"  "поле 2";
    "поле 3" : "имя записи1"                   struct "имя записи1" "поле 3"
  End;                                      };
```

Запись может включать в себя вариантную часть, которая позволяет определить тип, содержащий описания нескольких вариантов структуры. Вариантное поле в записи может быть только одно и оно всегда последнее. В свою очередь последнее поле в вариантной части может быть вариантным, то есть вариантное поле так же может быть вложенным.

Таким образом, записи с вариантами обеспечивают средства объединения записей, которые похожи, но не идентичны по форме. Объем памяти, необходимый для записи с вариантами складывается из длин полей фиксированной части и максимального по длине поля вариантной части. Тип данных в вариантной части при компиляции не проверяется, поэтому, контроль за правильностью ее использования возлагается на программиста.

Объявление вариантной записи:

```
Type
  "имя записи1" = Record
     "поле 1": "тип 1";
     "поле 2": "тип 2";
   …
     "поле n-1": "тип n-1";
      case "вариант"  of
        "вариант 1": ("поле 11": "тип 11", …, "поле 1n":"тип 1n")
        "вариант 2": ("поле 21": "тип 21", …, "поле 2n":"тип 2n")
….
        "вариант k": ("поле k1": "тип k1", …, "поле kn":"тип kn")
End;
```

Объявление записи с вложенной вариантной частью:

```
  "имя записи1" = Record
     "поле 1" : "тип 1";
     "поле 2" : "тип 2";
   …
     "поле n-1" : "тип n-1";
     case "вар."  of
```

```
    "вар.1" : ("поле11": "тип11", …, "поле1n":"тип1n")
    "вар.2" : ("поле21": "тип21", …, "поле n":"тип2n")
….
    "вар.k-1" : ("полек-11": "типk-11", …, "полек-1n":"типk-1n")
    case "вариант"  of
      "вариант1" : ("поле11": "тип11", …, "поле1n":"тип1n")
      "вариант2" : ("поле21": "тип21", …, "поле2n":"тип2n")
….
     "вариантk" : ("полек1": "типk1", …, "полекn":"типkn")
End;
```

Приведем пример. Пусть необходимо обеспечить вывод в заданную точку экрана монитора с координатами x и y следующих геометрических фигур: квадрата, прямоугольника, окружности и точки. При этом квадрат задается длиной своей стороны, прямоугольник – длинами двух сторон, окружность – координатами центра, которые совпадают с точкой (x,y) и радиусом. При желании мы можем еще закрасить окружность, что потребует дополнительно указания стиля и цвета закраски. Тогда можно определить следующий тип:

```
type
    figure=(sq,rec,cir,pix);                    //тип фигуры
    param = record                              //параметры вывода
          x,y: integer;                         //координаты
          fig: figure;                          //фигура
  case figure of                                //в зависимости от типа фигуры:
        sq:(side: word);                        // сторона квадрата
        rec:(side1, side2:word);                // стороны прямоугольника
        pix:();                                 // точка
        cir:(pc:record                          // окружность
              radius:word;
             stl:boolean;                  // закрасить?
             case boolean of                            //закраска – по желанию
                true: (st,cl:word);                     //стиль и цвет закраски
                false:();
             end;)
    end;
```

В языке Си вариантные записи реализуются структурами с объединениями Union (см. лекции).

На физическом уровне информация о переменной типа «запись» содержится в *дескрипторе* или *заголовке записи*. Дескриптор записи может содержать имя записи, количество полей, их имена, а также указатели значений элементов. Поля записи занимают в памяти непрерывную область, поэтому в дескрипторе достаточно иметь один указатель на начало записи, а в описании полей – смещение относительно начала. Смещение вычисляется при компиляции программы, что повышает эффективность доступа к полям записи.

При обработке записей с вариантами программисту необходимо следить за правильностью хранения и обработки данных, содержащихся в вариантной части.

Тип «запись» очень часто используется в информационно-поисковых системах, когда приходится хранить и обрабатывать большие объемы данных. Очень часто эти данные группируются в таблицы.

Таблица – это конечное множество элементов, имеющих одну и ту же организацию. Таблица в оперативной памяти обычно представляется в виде массива (в том числе, массива записей). Доступ к элементу таблицы осуществляется по его номеру (индексу). Часто для поиска или обработки данных в таблице используется одно (или несколько) полей записи. В этом случае такое поле называется ключевым полем или ключом записи.

Пусть, имеется таблица, содержащая данные из адресной книги: фамилию, год рождения, адрес и телефон человека (табл.1). Назовем ее исходной или основной.

Таблица1

Индекс	**Фамилия**	**Год рождения**	**Адрес**	**Телефон**
1	Сидоров	1990	Ул. Первая, 1	1234567
2	Петров	1989	Ул. Вторая, 2	2345678
3	Иванов	1980	Ул. Первая, 3	3456789

Если необходимо осуществлять частый поиск информации в основной таблице по фамилии, то ключевым полем будет являться фамилия. Если же при поиске нас больше интересует возраст человека (например, если эти же сведения используются в военкомате), то роль ключа может играть год рождения.

При больших размерах таблиц поиск данных, имеющих указанный ключ, может потребовать больших затрат времени. Если же помимо поиска требуется произвести сортировку данных, то временн**ые** затраты многократно возрастут, так как потребуется осуществлять их перестановку (перемещение). В этом случае можно уменьшить время обработки за счет создания дополнительного массива – таблицы

ключей, содержащей индекс элемента в исходной таблице и выбранный ключ. Если в качестве ключевого поля используется фамилия, то таблица ключей будет выглядеть как в табл. 2.

Таблица2

№п/п	Индекс исходной таблицы	Фамилия
1	1	Сидоров
2	2	Петров
3	3	Иванов

При необходимости произвести сортировку данных в основной таблице, (см. табл.1) осуществляется упорядочивание соответствующих ключей в дополнительном массиве (табл.3). При этом расположение записей в основной таблице не меняется.

Таблица3

№п/п	Индекс исходной таблицы	Фамилия
1	3	Иванов
2	2	Петров
3	1	Сидоров

Для того чтобы вывести информацию из основной таблицы (см. табл.1) в алфавитном порядке следования фамилий, надо данные из нее выбирать в том порядке, в котором они находятся в дополнительном массиве - в таблице ключей (см. табл.3). То есть, в нашем примере, первой должна быть напечатана третья запись из табл. 1, потом вторая, а потом первая.

Таким образом, если мы сортируем таблицу ключей, то экономится время, поскольку перестановка записей в исходной таблице, которая иногда может содержать достаточно большое число полей, отсутствует. Этот выигрыш во времени особенно заметен при большой размерности таблиц и при правильно подобранных ключах. При этом, правда, надо помнить, что для размещения таблицы ключей требуется дополнительная память. Кроме того, следует учитывать, что если в качестве ключа используется символьное поле записи, то это влечет за собой необходимость посимвольной обработки данного поля в цикле, и, следовательно, приводит к увеличению времени выполнения любых операций. Выбор данных из основной таблицы в порядке, определенном таблицей ключей, так же замедляет вывод этих данных.

Отметим еще, что при подобной организации вычислительного процесса разделяются понятия хранения данных и их структурирования и обработки. Например, добавим в табл.1 еще несколько записей и получим табл.4:

Таблица4

Индекс	**Фамилия**	**Телефон**	**Год рождения**	**Адрес**
1	Сидоров	1234567	1990	Ул. Первая,1
2	Петров	2345678	1989	Ул. Вторая,2
3	Иванов	3456789	1980	Ул. Первая,3
4	Петрова	4567892	1985	Ул. Вторая,3
5	Ющенко	5678912	1996	Ул. Третья,1
6	Антонов	6789123	1988	Ул. Первая,23

Эти записи вносятся, в конец таблицы. Тогда после упорядочивания по фамилии табл. 2 будет выглядеть как табл.5.

Таблица5

№п/п	**Индекс исходной таблицы**	**Фамилия**
1	6	Антонов
2	3	Иванов
3	2	Петров
4	4	Петрова
5	1	Сидоров
6	5	Ющенко

Итак, расположение данных в исходной таблице, т.е. в табл.1, осталось прежним. Следовательно, удалось снизить затраты на их структурирование.

Таблицы является достаточно распространенной структурой данных для многих программно–информационных систем. Например, их широко используют при разработке компиляторов для создания таблиц операций, идентификаторов, ошибок и т.д.

Принцип разделения хранения и структурирования информации часто применяется при хранении больших объемов информации в файлах, например, для хранения изображений. В этом случае создаются реестры (дополнительные массивы записей), содержащие ссылки на изображения, играющие роль ключевого значения, и имя файла.

Задание

Создать таблицу, содержащую не менее 40 записей с вариантной частью. Произвести поиск информации по вариантному полю. Упорядочить таблицу, по возрастанию ключей (где ключ – любое невариантное поле по выбору программиста), используя: **а)** исходную таблицу; **б)** массив ключей, используя 2 разных алгоритма сортировки (простой, ускоренный). Оценить эффективность этих алгоритмов (по времени и по используемому объему памяти) при различной реализации программы, то есть, в случаях **а)** и **б).** Обосновать выбор алгоритмов сортировки. Оценка эффективности должна быть относительной (в %).

Указания к выполнению работы

Интерфейс программы должен быть понятен неподготовленному пользователю. При разработке интерфейса программы следует предусмотреть:

- указание формата и диапазона данных при вводе и (или) добавлении записей;
- указание операций, производимых программой;
- наличие пояснений при выводе результата;
- возможность добавления записей в конец таблицы и удаления записи по значению указанного поля.
- просмотр **отсортированной** таблицы ключей при **несортированной** исходной таблице;
- вывод упорядоченной исходной таблицы;
- вывод исходной таблицы в упорядоченном **виде,** используя **упорядоченную** таблицу ключе
- вывод результатов сравнения эффективности работы программы при обработке данных в исходной таблице и в таблице ключей;
- вывод результатов использования различных алгоритмов сортировок.

Одним из результатов работы программы должна быть **количественная** информация (лучше представить в виде таблицы) с указанием времени, затраченного на обработку исходной таблицы и таблицы ключей двумя алгоритмами сортировки (при этом, не забыть оценить так же время выборки данных из основной таблицы с использованием таблицы ключей), а так же - объем занимаемой при этом оперативной памяти.

При тестировании программы необходимо:

- проверить правильность ввода и вывода данных (в том числе, отследить попытки ввода неверных по типу данных в вариантную часть записи);
- обеспечить вывод сообщений при отсутствии входных данных («пустой ввод»);
- проверить правильность выполнения операций;
- отследить переполнение таблицы.

При хранении исходных данных в файлах (что приветствуется) необходимо также проверить наличие файла и изменения информации в нем при удалении и добавлении данных в таблицу.

Содержание отчета

В отчете по лабораторной работе должна быть обоснована целесообразность применения типа «запись» с вариантной частью. Там же должен быть проведен сравнительный анализ и сделан вывод о том, какие преимущества и недостатки (с учетом времени, затрачиваемого на обработку данных, и необходимого объема памяти) влечет за собой применение дополнительного массива ключей при работе с таблицами. Вывод должен быть подтвержден числовыми данными, полученными в процессе тестирования программы на таблицах разной размерности. Кроме того, отчет по лабораторной работе должен содержать ответы на следующие вопросы:

1.Как выделяется память под вариантную часть записи?

2.Что будет, если в вариантную часть ввести данные, несоответствующие описанным?

3.Кто должен следить за правильностью выполнения операций с вариантной частью записи?

4.Что представляет собой таблица ключей, зачем она нужна?

5.В каких случаях эффективнее обрабатывать данные в самой таблице, а когда – использовать таблицу ключей?

6.Какие способы сортировки предпочтительнее для обработки таблиц и почему?

Отчет представляется в электронном или печатном виде.

Список рекомендуемой литературы

1. *Вирт Н.* Алгоритмы и структуры данных: Пер. с англ. СПб.: Невский диалект, 2001. С. 37-44, .67-69.

2. *Баунси Кен* Основные концепции структур данных и реализация в C++: Пер. с англ. - М.: Издат. Дом «Вильямс», 2002. С. 37-44.
3. *Иванова Г. С.* Основы программирования. М.: Изд-во МГТУ им. Н.Э. Баумана, 2001. С. 140-143