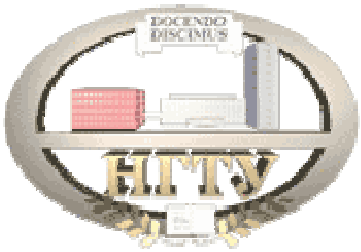
Министерство образования Российской Федерации
Новосибирский Государственный Технический Университет



Курсовой проект по дисциплине "Программирование"

Таблица данных произвольных типов в памяти

Факультет: *АВТФ*
Группа: *АП-319*
Выполнил: *Глухов С. А.*
Проверил: *Романов Е. Л.*

Новосибирск
2005

Содержание

Задание

Необходимо разработать интерфейс для объединения в структуру данных множества объектов различных классов - абстрактный базовый класс объектов object, для которого предусмотреть виртуальные методы: загрузки объекта из текстовой строки, выгрузки объекта в текстовую строку в динамической памяти, добавления объекта в последовательный двоичный файл, чтения объекта из последовательного двоичного файла, возврата уникального идентификатора класса, возврата указателя на строку с именем класса, сравнения двух объектов, “сложения“ (объединения) двух объектов, создание динамической копии объекта. Сделать классы хранимых объектов производными от абстрактного базового класса object. Предусмотреть создание заголовка таблицы со столбцами объектов выбранных типов, добавление, удаление, редактирование строк, сортировку по любому столбцу, сохранение и загрузку таблицы (структуры и содержимого) в текстовом файле.

Структурное описание разработки

Данная программа представляет собой реализацию таблицы данных произвольных типов в памяти и предлагает интерфейс для работы с этой таблицей. Чтобы реализовать в памяти таблицу с динамически изменяемым количеством строк и столбцов используется двумерный динамический массив. При необходимости добавления / удаления новых строк или столбцов, динамический массив перераспределяется, используя стандартную функцию Си `realloc`. Также в программе используется два вспомогательных одномерных динамических массива в которых хранятся названия столбцов и названия их типов. Эти вспомогательные массивы используются для формирования шапки таблицы. Увеличение или уменьшение количества столбцов таблицы влияет на размерность этих массивов. Добавление же или удаление строки таблицы на размерность вспомогательных массивов не влияет.

Класс Table представляет собой:

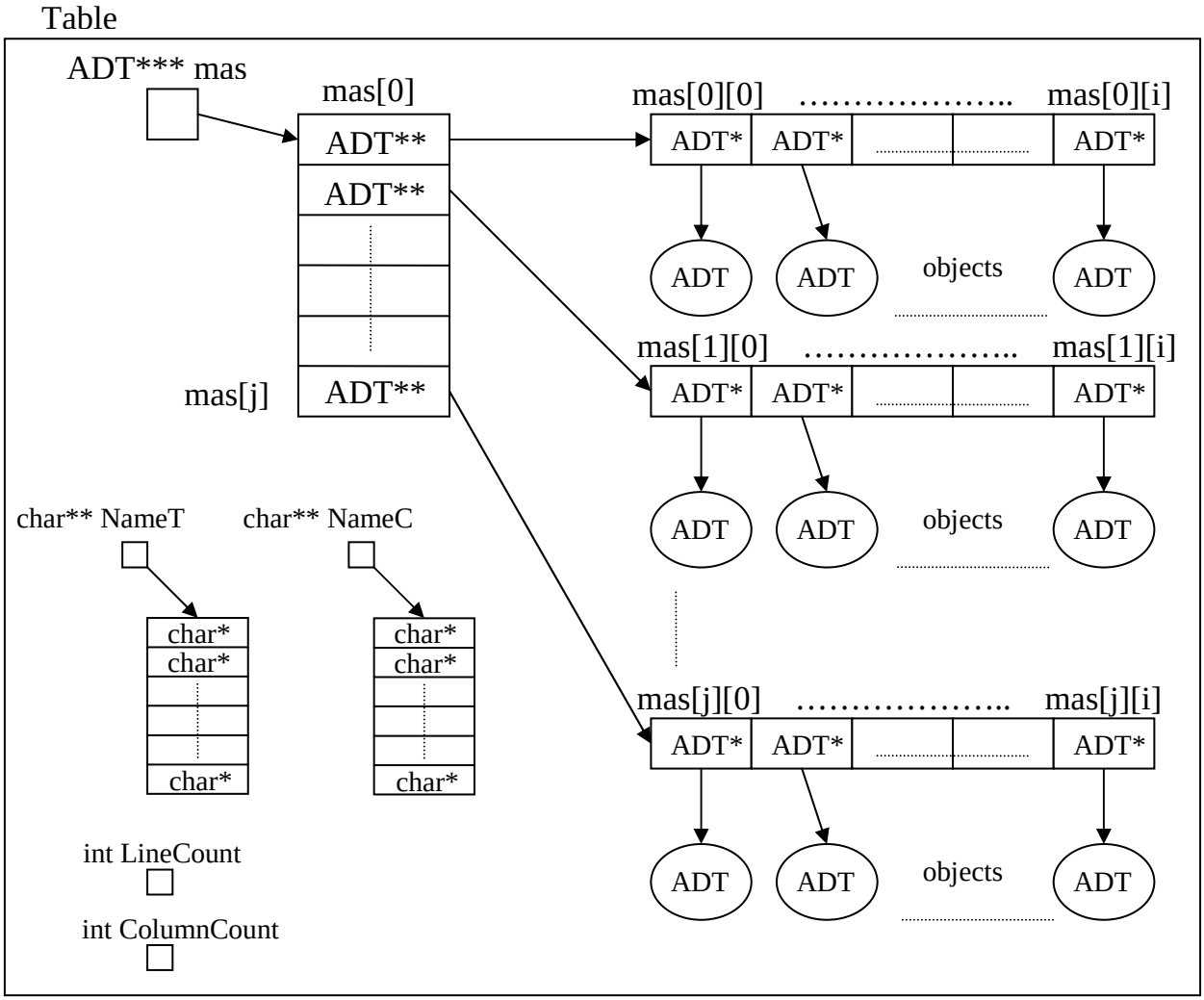


Рис. 1

Table – класс таблицы, mas – массив указателей типа ADT**, каждый элемент является, своего рода, заголовком столбца таблицы и ссылается на массив указателей типа ADT*, каждый из которых является элементом столбца таблицы и содержит ссылку на объект базового класса ADT. NameT – динамический массив указателей на строки в котором хранятся названия типов столбцов таблицы, NameC – динамический массив указателей на строки в котором хранятся названия столбцов, LineCount – количество строк в таблице, ColumnCount – количество столбцов в таблице, Objects - объекты, производные от абстрактного БК.

Различные классы хранимых данных являются производными от абстрактного БК. Внутри каждого их низ переопределяются методы загрузки объекта из текстовой строки, выгрузки объекта в текстовую строку в динамической памяти, добавления объекта в последовательный двоичный файл, чтения объекта из последовательного двоичного файла, возврата уникального идентификатора класса, возврата указателя на строку с именем класса, сравнения двух объектов, “сложения“ (объединения) двух объектов, создание динамической копии объекта.

При создании новой таблицы или чтении ее из файла вначале задается количество строк и столбцов, считывается идентификатор каждого столбца, название типа столбца и название самого столбца.

Переполнения в программе возникнуть не может поскольку при добавлении записи в таблицу сначала перераспределяется двумерный динамический массив на необходимую величину. Принцип работы при добавлении новой строки показан на Рис. 2

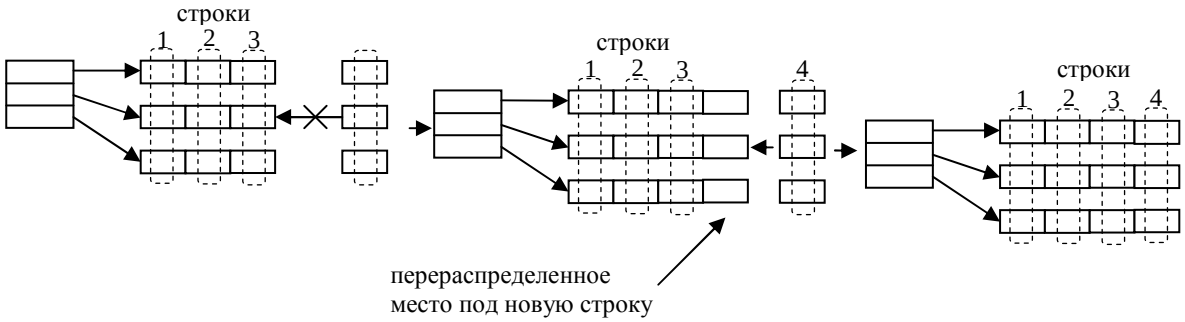


Рис. 2

Принцип работы при добавлении столбца Рис. 3

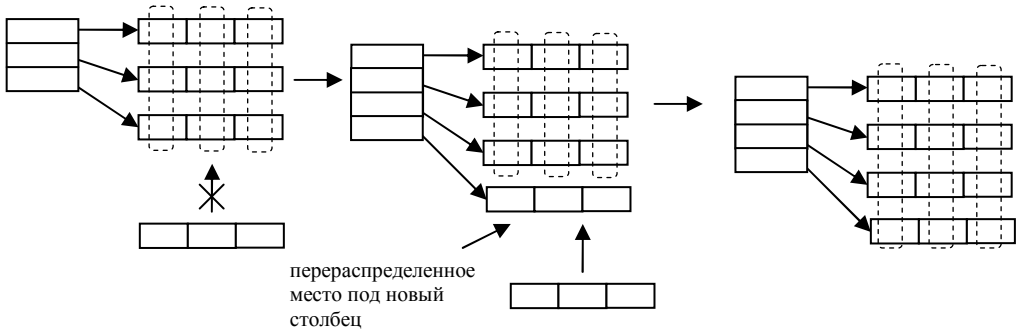


Рис. 3

При сохранении таблицы в файл записывается количество строк, столбцов, типы каждого из столбцов, названия столбцов, а затем строки самой таблицы.

При открытии существующей таблицы читается количество строк, столбцов типы каждого из столбцов, их названия. Эти данные служат для создания новой таблицы, количество строк которой равняется считанному количеству строк. Сначала создается таблица считанной размерности, после чего каждому столбцу присваивается считанный тип, а затем считываются сами строки таблицы.

В программе реализованы 3 типа сортировки. Предусмотрены запись строки в конец таблицы, редактирование строк по логическому номеру, удаление строки по логическому номеру, добавление столбца в конец таблицы, удаление столбца по логическому номеру, вывод строки по логическому номеру с шапкой таблицы.

Удаление строки или столбца по логическому номеру предусматривает “уплотнение“ массива указателей и уменьшение количества строк или столбцов для дальнейшей корректной работы программы.

Функциональное описание

Функция перераспределения строк таблицы

```
void ResizeLine(int);
```

В функцию передается новое количество строк в таблице, после чего соответствующим образом вызываются функции перераспределения памяти и таблица подготавливается для добавления новой строки.

Функция перераспределения столбцов таблицы

```
void ResizeColumn(int);
```

В функцию передается новое количество столбцов в таблице, после чего соответствующим образом вызываются функции перераспределения памяти и таблица подготавливается для добавления нового столбца.

Функция создания новой таблицы:

```
void CreateTable();
```

Изначально в функцию не передается никаких параметров. В процессе своей работы функция запрашивает у пользователя желаемое количество строк и столбцов, которое будет в таблице. Получив эти данные, вызываются функции перераспределения памяти для строк таблицы и для столбцов. Также пользователь должен будет ввести типы всех столбцов и указать их названия, после чего нужно будет ввести данные каждой строки таблицы.

Функция добавления строки в конец таблицы:

```
void InsertLine();
```

Изначально в функцию не передается никаких параметров. В процессе своей работы функция запрашивает у пользователя данные, которые будут введены в строку таблицы. Но перед этим перераспределяется память под новую строку в таблице, типы элементов строки определяются автоматически, для этого просматриваются первые элементы всех столбцов и элементам новой строки присваивается такой же тип, как и у соответствующих первых элементов в столбце таблицы. После этого данные считываются в строку таблицы.

Функция добавления столбца в конец таблицы:

```
void InsertColumn();
```

Изначально в функцию не передается никаких параметров. В процессе своей работы функция запрашивает у пользователя тип столбца, его название, которые будут введены в новый столбец таблицы. Перед этим перераспределяется память под новый столбец в таблице, количество элементов в новом столбце автоматически становится равным числу строк в таблице, типы элементов столбца создаются в соответствии с указаниями пользователя. После того, как выделена память под новый столбец и заданы типы элементов, пользователю будет предложено ввести данные столбца.

Функция удаления строки из таблицы по указанному логическому номеру:

```
void DelLine(int);
```

В функцию передается логический номер строки, которую нужно удалить. После этого с помощью цикла выполняется поиск строки в массиве. Затем происходит уплотнение массива, в результате чего требуемая строка удаляется. Уплотнение строк происходит путем уплотнения элементов каждого столбца. После удаления строки также происходит перераспределение памяти и счетчик количества строк уменьшается.

Функция удаления столбца из таблицы по указанному логическому номеру:

```
void DelColumn(int);
```

В функцию передается логический номер столбца, который нужно удалить. После этого с помощью цикла выполняется поиск столбца в массиве. Затем происходит уплотнение массива, в результате чего требуемая столбец удаляется. После удаления столбца также происходит перераспределение памяти и счетчик количества столбцов уменьшается.

Функция редактирования строки по логическому номеру:

```
void EditLine(int);
```

В функцию передается логический номер требуемой строки, после чего выполняется поиск в массиве с помощью цикла. Затем старые данные строки заменяются на данные введенные пользователем.

Функция вывода строки на экран по логическому номеру :

```
void ShowLine(int);
```

В функцию передается логический номер строки, которую требуется вывести на экран. Выполняется поиск строки в массиве с помощью цикла. После того как строка найдена она выводится на экран. Для удобства восприятия вместе со строкой на экран выводится шапка таблицы.

Функция вывода всей таблицы на экран:

```
void ShowTable();
```

Чтобы выводимая на экран таблица смотрелась красиво и все столбцы были ровными, функция сначала выполняет поиск максимального по длине элемента в каждом столбце таблице, включая шапку таблицы, и заносит максимальные длины элементов в каждом столбце в динамический массив. Далее начинается вывод строк таблицы с учетом того, что между каждыми элементами строки добавляется нужное количество разделителей, вычисленное на основе ранее определенных максимальных длин элементов каждого столбца.

Функция сортировки рекурсивным разделением:

```
void RekSort(int,int,int);
```

Функция получает логический номер столбца, по которому будет происходить сортировка, логический номер первой строки, последней строки. Происходит разделение в одном массиве указателей на строки с использованием алгоритма на основе обмена. Сравнение объектов указанного столбца происходит с концов массива указателей (i=a, j=b) к середине (i++ или j--), причем “укорочение“ происходит только с одной из сторон. После каждой перестановки меняется тот конец, с которого выполняется “укорочение“. В результате этого массив указателей разделяется на две части относительно первого объекта указанного столбца, который и становится медианой. Медиана делит массив указателей на две неравные части. Алгоритм выполняется итерационно. В каждом шаге итерации медиана двигается к середине массива.

Функция сортировки "пузырьком":

```
void Sort(int);
```

В функцию передается логический номер столбца по которому производится сортировка.

Суть ее заключается в следующем: производятся попарное сравнение соседних элементов 0-1, 1-2 ... и перестановка, если пара расположена не в порядке возрастания. Просмотр повторяется до тех пор, пока при пробегании массива от начала до конца перестановок больше не будет.

Функция однонаправленной Шейкер-сортировки:

```
void SheikerSort(int);
```

В функцию передается логический номер столбца по которому производится сортировка.

Усовершенствованная сортировка методом "пузырька", отличие заключается в том, что запоминается индекс последней перестановки, и передается во внешний цикл в качестве крайней границы. Условие окончания –граница сместится к началу массива.

Функция сохранения в файл:

```
void SaveToFile(char*);
```

В функцию передается имя файла.

Сохранение в файл происходит точно так же, как и вывод на экран, за исключением того, что в начало файла записывается размерность таблицы, а затем идентификаторы всех столбцов таблицы, далее сохраняются самис троки таблицы.

Функция загрузки из файла:

```
void LoadFromFile(char*);
```

В функцию передается имя файла.

Сначала считывается размерность таблицы и с помощью вспомогательных функций под эту таблицу распределяется память. Затем идентификаторы типов таблицы, далее считываются названия типов столбцов и названия самих столбцов. Затем считываются сами строки таблицы.

Описание работы программы на контрольном примере и выводы

Создается таблица, состоящая из 3 столбцов: типы столбцов – integer. Значения столбцов файлов получены при помощи дополнительной программы Generator, которая использует функцию генерации случайных величин rand().

Число строк	Время сортировки "Пузырек", сек.	Время Шейкер-сортировки, сек.	Время рекурсивной сортировки, сек.	Время вывода на экран, сек.	Время загрузки, сек.
1 000 (19 КБ)	0,172	0,093	0,15	0,062	0,015
10 000 (186 КБ)	35,625	17,625	0,031	0,578	0,0125
100 000 (1,85 МБ)	-	-	0,563	6,172	1,437
500 000 (9,26 МБ)	-	-	3,906	31,594	41,469

Из таблицы видно, что наиболее эффективная сортировка – это сортировка рекурсивным разделением.

Литература

1. Романов Е.Л. Язык СИ++ в задачах, вопросах и ответах. Новосибирск: Изд-во НГТУ, 2003.
2. Архангельский А.Я. Программирование в C++Builder 6. – М.: "Издательство БИНОМ", 2003 г. – 1152 с.: ил.