Лабораторная работа № 1

Обработка данных в виде массива структур средствами языка C++

Цель работы: научиться конструировать пользовательские типы данных — структуры, создавать массивы структур в динамической памяти, осуществлять ввод/вывод из файла.

Задание

Написать программу, представляющую собой простейшую систему для управления данными. Исходные данные для программы хранятся в текстовом файле в виде строк. Для создания файла с исходными данными следует использовать простейший текстовый редактор или редактор инструментальной среды разработки. Количество строк, требования к содержимому строки, способ обработки определяются вариантом задания. В соответствии со своим вариантом следует сконструировать структуру и на ее основе создать массив в динамической памяти.

Действия, выполняемые программой:

1. Чтение данных из файла в динамический массив.
2. Просмотр данных, хранящихся в массиве.
3. Корректировка данных заданной строки.
4. Вычисления.
5. Сохранение данных в новом файле (имя файла вводится с клавиатуры).

Алгоритм работы программы должен предусматривать однократное чтение данных с проверкой, многократное редактирование, просмотр записей, вычисления. Работа программы завершается после сохранения данных в новом файле.

Варианты заданий

В текстовом файле с исходными данными находится таблица, состоящая из n строк, в каждой по m слов, образующих, соответственно, m столбцов. Между словами расстояние — 1 пробел. Тип данных в каждом столбце должен соответствовать заданию.

1 вариант

Количество строк: 4. Столбцы: Название программы, Разработчик, Версия, Год выпуска. Определить самую новую программу.

2 вариант

Количество строк: 3. Столбцы: Номер школы, Название школы, Специализация, Количество учащихся. Вычислить общее количество учащихся.

3 вариант

Количество строк: 3. Столбцы: Название товара, Категория товара, Цена, Количество. Вычислить общую стоимость товара.

4 вариант

Количество строк: 4. Столбцы: Номер банковской карты, Фамилия владельца, Год окончания действия, Остаток на счете. Определить владельца карты с минимальным остатком средств.

5 вариант

Количество строк: 5. Столбцы: Фамилия, Количество отработанных дней, Тариф. Вычислить сумму заработной платы.

6 вариант

Количество строк: 4. Столбцы: Марка машины, Мощность двигателя, Объем бака, Цвет кузова . Вычислить машину с самым мощным двигателем.

7 вариант
Количество строк: 4. Столбцы: Марка монитора, Максимальное разрешение, Цена. Вычислить среднюю цену.
8 вариант
Количество строк: 5. Столбцы: Фамилия студента, Предмет, Оценка. Вычислить количество двоек.
9 вариант
Количество строк: 3. Столбцы: Марка принтера, Формат бумаги, Скорость печати, Цена. Определить самый дешевый принтер.
10 вариант
Количество строк: 4. Столбцы: Название турфирмы, Маршрут, Количество оставшихся путевок. Вычислить общее количество оставшихся путевок.
11 вариант
Количество строк: 3. Столбцы: Фамилия, Имя, Должность, Оклад. Определить самого высокооплачиваемого сотрудника.
12 вариант
Количество строк: 5. Столбцы: Станция отправления, Станция прибытия, Время в пути. Определить маршрут с наименьшим временем в пути.
13 вариант
Количество строк: 4. Столбцы: Фамилия спортсмена, Вид спорта, Разряд, Название спортивного клуба. Вычислить количество спортсменов, имеющих первый разряд.
14 вариант
Количество строк: 5. Столбцы: Название книги, Автор, Год издания. Определить самое старое издание.
15 вариант
Количество строк: 3. Столбцы: Фамилия, Отдел, Год поступления на работу, Образование. Определить средний стаж работы.
16 вариант
Количество строк: 4. Столбцы: Фамилия студента, Название вуза, Курс, Факультет. Определить количество студентов второго курса.
17 вариант
Количество строк: 5. Столбцы: Фамилия абонента, Продолжительность разговора в мин., Стоимость минуты разговора. Вычислить стоимость всех разговоров.
18 вариант
Количество строк: 3. Столбцы: Фамилия, Имя, Род занятий (сотрудник, студент), Год поступления. Вычислить сотрудника, принятого на работу последним.
19 вариант
Количество строк: 4. Столбцы: Название предмета, Преподаватель, Количество лекций, Количество лабораторных работ. Вычислить количество часов занятий по всем предметам (лекции и лабораторные работы имеют продолжительность 2 часа).
20 вариант
Количество строк: 5. Столбцы: Фамилия, Место жительства, Год рождения. Определить средний возраст.
21 вариант
Количество строк: 4. Столбцы: Название фирмы, Адрес, Телефон, Электронный адрес. Вычислить количество фирм, не указавших электронный адрес.
22 вариант
Количество строк: 3. Столбцы: Фамилия, Номер договора, Стоимость заказа, Срок исполнения . Вычислить среднюю стоимость заказа.
23 вариант
Количество строк: 5. Столбцы: Название журнала, Номер, Год выпуска. Вычислить количество

журналов, выпущенных в текущем году.

24 вариант

Количество строк: 4. Столбцы: Название группы, Факультет, Количество студентов, Количество успевающих студентов. Вычислить процент успевающих студентов по всем факультетам.

25 вариант

Количество строк: 3. Столбцы: Марка телефона, Фирма изготовитель, Вес, Цена. Определить самый легкий телефон.

Справочный материал

Структуры

Структура — это тип данных языка С, определяемый пользователем. Структуры также используются в языке С++. Структура сможет содержать произвольное количество полей в виде переменных различных типов, как стандартных, так и пользовательских. Исключение: структура не может содержать поле того же типа, что и она сама. Под структуру выделяется область памяти, равная сумме областей памяти, необходимых для хранения каждого поля.

Пример объявления типа структуры:

```
struct Book
{
     int ID; // идентификатор, поле целого типа
     char Author[30]; // текстовое поле (строка в виде
массива символов)
     char Title[50];
     float Price;
} ;
```

Тип структура (`struct Book`) используется для создания структурной переменной:

```
struct Book MyBook;
```

Переменная `MyBook` занимает в памяти 4 (int ID)+30(char Author[30])+50(char Title[50]) +4(float Price) = 88 байтов.

Создание массива структур:

```
struct Book Library[10];
```

Обработка структурных переменных сводится обычно к действиям над отдельными полями структуры в соответствии с их типом.

Обращение к полям структуры через имя структурной переменной использует операцию точка (.):

```
MyBook.ID=12345;
strcpy(MyBook.Author, «A.S.Pushkin»);
Library[3].ID=MyBook.ID;
strcpy(Library[3].Author,MyBook.Author);
```

Для обращения к полям структуры через указатель используется операция стрелка (→):

```
(Library+3)->ID=54321;
```

Функции для обработки строк в языке С

Объявление строки в виде массива символов:

```
char st[20];
```

Объявленная таким образом строка может хранить не более 19 символов.

Ввод строки с клавиатуры:

```
scanf(''%s'', st);
```

Функции обработки строк (заголовочный файл string.h):

`strlen (str)` – возвращает фактическое количество символов в строке `str`;

`strcpy (str1, str2)` – копирует символы из строки `str2` в строку `str1`, строка `str2` при этом не меняется;

`strcmp (str2, str2)` – сравнивает две строки, возвращает 0, если строки одинаковые, иначе функция возвращает ненулевое значение.

`strcat (str1, str2)` – объединяет две строки, при этом символы из `str2` помещаются в конец строки `str1`, в конец объединенной строки добавляется символ конца строки `'\0'`, строка `str2` при этом не меняется.

Замечание. Функция `strcpy()` относится к небезопасным функциям, т.к. она не контролирует переполнение строки-приемника. Избежать ошибки во время работы программы, связанной с использованием функции `strcpy()`, можно, если вместо `strcpy()` использовать функцию `strncpy(str1, str2, n)`, копирующую не более чем n первых символов из строки `str2` в строку `str1`, или выполнить посимвольное копирование из буфера в выделенную область памяти с проверкой на переполнение. Аналогично, вместо небезопасной функции `strcat()` можно использовать функцию `strncat(str1,str2,n)`, которая присоединит к строке `str1` не более `n` символов из строки `str2`.

Динамическая память

Динамическая память — это память, свободно распределяемая программистом. Для сохранения данных динамическая память должна быть запрошена в достаточном объеме (выделена), по окончании работы с этими данными память должна быть освобождена, т.е. возвращена в общее пользование и может быть в дальнейшем повторно выделена для других данных. Невыполнение освобождения памяти с последующим выделением приводит к накоплению ненужных данных и к переполнению области динамической памяти, что приводит к сбоям в работе программы («утечка памяти»).

В C++ для работы с динамической памятью используются операторы `new, delete, new[ ], delete[ ]`. Оператор new используется для выделения памяти под одну переменную, delete для освобождения памяти из-под одиночной переменной. Для массивов эти операторы используются в форме `new[ ]` и `delete[ ]`. Для хранения адреса выделенной области памяти требуется переменная-указатель. Сама динамическая переменная не имеет имени.

```
int* pNumber;
pNumber =new int; // Выделено 4 байта для хранения переменной типа  int
char* pName;
pName = new char[10]; // Выделено 10 байтов для хранения массива символов

*pNumber = 4;
strcpy(pName, «Александр»); //Надо помнить, что
                // в выделенную под переменную память
                //можно скопировать не более 9 символов
std::cout << *pNumber<< ” “ << pName << std::endl;
                //Освобождение динамической памяти
                //из-под простой переменной и массива
                //и обнуление указателя, чтобы
          //предотвратить обращение к освобожденной памяти
delete pNumber;
pNumber = NULL;
delete[ ] pName;
```

```
pName = NULL;
```

Файловый ввод/вывод

Работа с файлами в C++ основана на создании пользователем объектов-потоков, которые образуют канал между программой и устройством чтения/записи. Для создания таких объектов используется заголовок `fstream`. Также существуют стандартные объекты-потоки (заголовок `iostream`), осуществляющие консольный ввод (`cin`), вывод (`cout`), вывод сообщений об ошибках (`cerr`).

Алгоритм работы с объектами-потоками:

1. Создание объекта-потока и открытие его в заданном режиме.
2. Выполнение действий с данными.
3. Закрытие потока.

Создание объекта-потока для чтения из файла:

```
std::ifstream fin; // Создание входного потока
fin.open(“myfile.txt”); // Открытие входного потока
```

Создание объекта-потока для записи в файл:

```
std::ofstream fout; // Создание выходного потока
fout.open(“yourfile.txt”); // Открытие выходного потока
```

Операции чтения/записи:

```
int a;
fin >> a; // Чтение из файла в переменную
fout << a; // Запись в файл
```

Закрытие потоков:

```
fin.close();
fout.close();
```

Открытие входного потока может быть неуспешным, например, если входной файл не будет найден. В этом случае нормальная работа программы прерывается. При открытии файла на чтение следует обязательно проверить факт открытия файла:

```
if (fin)
{
     …
     fin.close( );
}
else
     std:: cout <<” Файл не найден!“ << std:: endl;
```