ЛАБОРАТОРНАЯ РАБОТА № 16
СТРУКТУРЫ И ФАЙЛЫ

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Цель работы – научиться работать со структурами и массивами структур, освоить формат доступа к элементам структуры и к элементам массива структур. Научится создавать текстовые файлы с массивом структур, овладеть техникой ввода и вывода элементов структур.

Примеры функций для работы со структурами

При создании программного обеспечения часто приходится обрабатывать информацию, в которой присутствуют разные типы данных. Например, информация о студенте в базе данных деканата включает в себя фамилию, имя, отчество студента, дату рождения, дату поступления в университет, курс, группу, средний балл и т.д. Для хранения этой информации удобно использовать структуры, которые позволяют объединить под одним именем подобную разнотипную информацию.

Кроме объявления структуры, нужно уметь создавать массивы структур, сохранять эту информацию в файле, читать ее из файла, отображать на дисплее и т.д.

Структуры были подробно описаны в лабораторной работе 13, а запись информации в файл и чтение из файла – в лабораторной работе 15.

Структура – это создаваемый программистом новый тип данных, следовательно его надо предварительно объявить в начале программы. При этом в программе еще не создается никаких переменных, поскольку это только описание типа. Например, для программы, которая позволяла бы хранить данные о пациентах (фамилия, возраст, вес, рост), удобно было бы объявить такую структуру:

```
struct ManInfo
{
  char      Family[50]; // Фамилия
  short     Age;        // Возраст
  float     Weight;     // Вес
  float     Height;     // Рост
};
```

Под поле «Фамилия» отводим массив из 50 символов, под поле «Возраст» - переменную типа short, под «Вес» и «Рост» – переменные типа float. Имя структуры – `ManInfo` – будет *именем нового типа данных*, и теперь в программе можно создавать переменные *этого типа*:

```
ManInfo a; // Объявление переменной a типа ManInfo
```

Программа выделяет в памяти под переменную `a` 60 байт (суммарный размер всех полей структуры). Для обращения ко всей переменной используется ее *имя*, а для обращения к ее отдельному полю – *оператор точка и имя поля*.

С полем структуры можно работать так же, как и с переменной соответствующего типа: числовые переменные могут участвовать в арифметических выражениях, для работы со строками используют соответствующие функции.

```
a.Age = 32;
a.Weight = 64.5;
a.Height = 175;
// a.Family = "Иванов А.А.";  ОШИБКА!!!
```

Поле `Family` задать простым присваиванием нельзя, т.к. это строка. Можно использовать библиотечную функцию копирования строк (про строки подробно написано в лабораторной работе 14):

```
strcpy(a.Family, "Иванов А.А.");
```

Аналогично можно объявить массив структур:

```
ManInfo Array[10]; // Массив из 10 переменных типа ManInfo
```

Можно объявить указатель на структуру и затем создать динамический массив с помощью оператора new:

```
ManInfo* Arr;
. . . .
Arr = new ManInfo[n];
```

Поля у элементов массива заполняются аналогично: берем нужный элемент массива (например, *i*-тый) и с помощью оператора точка обращаемся к требуемому полю структуры.

```
Arr[i].Age = 45;
Arr[i].Weight = 69.5;
Arr[i].Height = 177;
strcpy(Arr[i].Family, "Петров С.А.");
```

Чтобы заполнять поля структур, удобно написать функцию, в которую передавать указатель на структуру. Функция не имеет возвращаемого значения, поэтому заголовок будет выглядеть так:

```
void SetInfo(ManInfo* pman)
```

В самой функции будем задавать по очереди все поля, но, так как pman – это указатель на структуру, то для обращения к ее отдельному полю нужно использовать *оператор стрелка и имя поля:*

```
pman->Family
```

Для ввода информации с клавиатуры будем использовать функцию scanf():

```
void SetInfo(ManInfo* pman) // Задает поля структуры по указ. pman
{
  printf("Фамилия: ");
  scanf("%s", pman->Family);
  printf("Возраст: ");
  scanf("%d", &pman->Age);
  printf("Вес: ");
  scanf("%f",&pman->Weight);
  printf("Рост: ");
  scanf("%f", &pman->Height);
}
```

Здесь поле pman->Family – указатель на строку, поэтому символ & перед ним не нужен.

Однако, тестирование функции показывает, что информация не всегда вводится корректно:

```
int main()
{
  ManInfo man;
  setlocale(0, "Russian");

  SetInfo(&man);
  printf("%s\n", man.Family);
  printf("%d\n", man.Age);
```

```
    printf("%f\n", man.Weight);
    printf("%f\n", man. Height);

    _getch();
    return 0;
}
```

Например, если на запрос фамилии ввести “Ivanov A.A.”, то программа пропускает ввод остальных полей. Это происходит из-за того, что функция scanf() вводит символы *только до первого пробела*, а остальные остаются во входном буфере. Поэтому при следующем вызове функции scanf() эти символы берутся из входного буфера, не дожидаясь ввода пользователя. Чтобы ввод корректно работал, нужно после каждого вызова scanf() очищать буфер.

Аналогичная ситуация будет при выполнении следующего фрагмента программы:

```
char c;
char s[32];
puts("Введите символ: ");
scanf(“%c”, &c);
puts("Введите строку: ");
scanf(“%s”, s);
```

Если на запрос «Введите символ» ввести с клавиатуры, например, "y" и нажать Enter, входной буфер будет содержать "y\n". При этом функция scanf(“%c”, &c) берет только первый символ из входного буфера ("y"), а "\n" остается в буфере, поэтому следующая функция scanf(“%s”, s) берет его из буфера и возвращает пустую строку, не дожидаясь ввода пользователя.

Чтобы избавиться от этого, нужно перед вызовом scanf(“%s”, s) очистить входной буфер. Это можно сделать функцией fflush(stdin), но в последних версиях Visual Studio она не работает.

Для корректной очистки потока вместо fflush(stdin) лучше очищать буфер функцией getchar(), вызывая ее в цикле до тех пор, пока не встретится символ ‘\n’:

```
while (getchar() != '\n');
```

Теперь при выполнении следующего фрагмента программы все будет работать правильно:

```
char c;
char s[32];
puts("Введите символ: ");
scanf(“%c”, &c);
while (getchar() != '\n');
puts("Введите строку: ");
scanf(“%s”, s);
while (getchar() != '\n');
```

В функции SetInfo() тоже желательно после каждого ввода очищать буфер. Кроме того, чтобы можно было вводить строку с пробелами (“Ivanov A.A.”) нужно вместо функции scanf() использовать gets_s().

Функция SetInfo() будет выглядеть так:

```
void SetInfo(ManInfo* pman) // Задает поля структуры
{                           // по указателю pman
    printf("Фамилия: ");
    gets_s(pman->Family, 31);
    printf("Возраст: ");
    scanf("%d", &pman->Age);
    while (getchar() != '\n');
    printf("Вес: ");
```

```
    scanf("%f", &pman->Weight);
    while (getchar() != '\n');
    printf("Рост: ");
    scanf("%f", &pman->Height);
    while (getchar() != '\n');
  }
```

Теперь в функции main() можно объявить массив структур и последовательно их заполнить. При вызове функции SetInfo() передаем в нее в качестве аргумента адрес очередного элемента структуры:

```
  ManInfo* MArr;
  int size;
  // здесь нужно задать size
  MArr = new ManInfo[size];
  for(int i=0; i<size; i++)
  {
    printf("Задайте данные %d пациента:\n", i + 1);
    SetInfo(&MArr[i]);
  }
```

Для отображения данных, записанных в структурах тоже лучше написать отдельную функцию. В качестве параметров будем передавать в нее указатель на структуру (имя массива), и размер массива. В функции отображаем данные в виде таблицы, для форматирования используем возможности функции printf() (про форматированный вывод информации подробно написано в лабораторной работе 4).

```
void ViewInfo(ManInfo *Arr, int Len) // Вывод инф-и на дисплей
{
  printf("=============================================================\n");
  printf("|  N |        Фамилия        | Возраст |   Вес   |   Рост  |\n");
  printf("=============================================================\n");
  for (int i = 0; i < Len; i++)
    printf("| %2d | %-20s |   %2d    |  %5.1f  |  %5.1f  |\n",
    i+1,Arr[i].Family, Arr[i].Age, Arr[i].Weight, Arr[i].Height);
  printf("=============================================================\n");
}
```

Еще одна проблема проявляется при вводе и **отображении русских букв**. Функция setlocale(LC_ALL, “Russian”) работает только *на вывод* символов на дисплей, а на ввод с клавиатуры – нет. Решить проблему можно по-разному, например после ввода строк, в которых могут быть русские буквы, добавить перекодировку с помощью функции OemToAnsi().

```
  OemToAnsi(s1, s2);
```

Функция преобразует символы строки s2 из кодировки OEM в кодировку ANSI. Результат сохраняется в строку s1:

В нашем примере это можно сделать так:

```
  OemToAnsi(pman->Family, pman->Family);
```

Но лучше поступить иначе. Вместо функции setlocale() можно использовать две функции: SetConsoleCP() и SetConsoleOutputCP().

Функция SetConsoleCP() устанавливает кодовую страницу **ввода**, используемую консолью, чтобы преобразовывать ввод информации с клавиатуры в соответствующие символьные значения.

Функция SetConsoleOutputCP() устанавливает кодовую страницу **вывода** данных, используемую консолью, чтобы преобразовать символьные значения, написанные различными функциями вывода информации в изображение, показываемое на экране в консольном окне.

Для установки кодовой страницы 1251 (с русскими буквами) нужно в начале функции main() вставить:

```
SetConsoleCP(1251);
SetConsoleOutputCP(1251);
```

тогда и ввод, и вывод будут корректно работать. Не забудьте подключить заголовочный файл #include <windows.h>.

Для демонстрации работы функций напишем простую программу:

```
int main()
{
  SetConsoleCP(1251);
  SetConsoleOutputCP(1251);

  ManInfo* MArr;
  int size = 3;

  printf("Введите число пациентов: ");
  scanf("%d", &size);
  while (getchar() != '\n'); // Сброс входного буфера

  MArr = new ManInfo[size];
  for (int i = 0; i < size; i++)
  {
    printf("Задайте данные %d пациента:\n", i + 1);
    SetInfo(&MArr[i]);
  }
  ViewInfo(MArr, size);

  _getch();
  return 0;
}
```

Пример работы программы:

```
Введите число пациентов: 3
Задайте данные 1 пациента:
Фамилия: Иванов А.Б.
Возраст: 23
Вес: 67
Рост: 170
Задайте данные 2 пациента:
Фамилия: Петров С.М.
Возраст: 35
Вес: 75
Рост: 180
Задайте данные 3 пациента:
Фамилия: Сидоров Н.И.
Возраст: 56
Вес: 75
Рост: 175
=============================================================
|  N |        Фамилия        | Возраст |   Вес   |   Рост  |
=============================================================
|  1 | Иванов А.Б.           |   23    |   67.0  |  170.0  |
|  2 | Петров С.М.           |   35    |   75.0  |  180.0  |
|  3 | Сидоров Н.И.          |   56    |   75.0  |  175.0  |
=============================================================
```

В программе желательно проверять корректность вводимой с клавиатуры информации и не позволять вводить заведомо неверные данные, например, чтобы нельзя было ввести возраст 1000 лет или вес – 1500 кг. При некорректном вводе нужно предлагать пользователю ввести данные заново.

Примеры функций для работы с файлами

Для хранения информации используют файловую систему компьютера. Структуры можно сохранять в текстовые или двоичные файлы. **В текстовых файлах** вся информация может быть добавлена или откорректирована в обычном текстовом редакторе. Для чтения из текстового файла удобно использовать функцию fscanf(), а для записи в текстовом формате – функцию fprintf() (описание работы функций см. в лабораторной работе 15).

```
FILE *f1;
ManInfo m;
SetInfo(&m); // Задаем поля структуры

f1 = fopen("list.dat", "a"); // открываем файл list.dat для дозаписи
fprintf(f1,"%s\n", m.Family); // пишем строку,
fprintf(f1,"%d\n", m.Age);    //       целочисленную переменную,
fprintf(f1,"%f\n", m.Weight); //       вещественную переменную,
fprintf(f1,"%f\n", m.Height); //       вещественную переменную
fclose (fp);  // закрываем файл

ManInfo a;
f1 = fopen("list.dat", "r");// открываем файл list.dat для чтения
fscanf(f1,"%s\n",&a.Family); // читаем
fscanf(f1,"%d\n",&a.Age);
fscanf(f1,"%f\n",&a.Weight);
fscanf(f1,"%f\n",&a.Height);
fclose (fp);  // закрываем файл
```

Записывать и читать приходится все поля по очереди. Чтобы организовать запись или чтение нескольких структур, нужно использовать цикл и массив структур. При чтении информации из файла количество записанных в нем структур заранее неизвестно, поэтому нужно придумать способ его определения.

Гораздо удобнее записывать структуры **в двоичные файлы**, поскольку можно за один раз прочитать или записать сразу одну или даже несколько структур. При чтении из двоичного файла используется функция fread(), в которую требуется передать адрес нужной области памяти (куда записать прочитанные данные), размер одного блока и количество таких блоков.

Для того, чтобы не вычислять *размер структуры* вручную (можно легко ошибиться), применяют оператор sizeof(), который можно использовать двумя способами: sizeof(ManInfo) или sizeof(man), поскольку запись man – это как раз один экземпляр структуры ManInfo.

```
ManInfo man;
int n;
FILE *fp;
fp = fopen("List.dat", "rb"); // открываем list.dat для чтения
                              // из бинарного файла
n = fread (&man, sizeof(ManInfo), 1, fp);
if ( n == 0 )
  printf ( "Ошибка при чтении из файла" );
fclose ( fp );
```

В данном примере функция fread() читает один блок размером sizeof(ManInfo), т.е. 60 байт, из файла, связанного с потоком fp, и записывает его по адресу &man. Функция возвращает число удачно прочитанных элементов (в нашем случае – структур). Поэтому если переменная n равна нулю, чтение закончилось неудачно и надо вывести сообщение об ошибке.

Для записи структуры в двоичный файл используют функцию fwrite(). Ее параметры – те же, что и у fread() Пример ниже показывает добавление информации из структуры a в конец двоичного файла.

```
ManInfo a;
FILE *fp;
SetInfo(&a);
fp = fopen("List.dat ", "ab");
fwrite(&a, sizeof(ManInfo), 1, fp);
fclose ( fp);
```

Для ввода с клавиатуры информации и записи в файл лучше написать отдельную функцию. В качестве параметра нее будем передавать имя файла, а возвращать она может число записанных структур. В функции откроем файл и в цикле будем вводить информацию о пациентах с клавиатуры и добавлять ее в файл.

```
int AddToFile(const char* FName) // Выводит инф-ю в файл
{                                // с именем FName
  FILE* f1;
  ManInfo dat;
  char key;
  int count = 0; // Для подсчета числа сохраненных структур

  printf("Запись информации в файл %s\n", FName);

  f1 = fopen(FName, "ab"); // открываем файл для дозаписи
  do
  {
    printf("Введите данные пациента: \n");
    SetInfo(&dat);
    printf("Добавить в файл (y/n)? \n");
    key = _getch();
    if (key == 'y' || key == 'Y')
    {
      int n = fwrite(&dat, sizeof(ManInfo), 1, f1);
      printf("Данные добавлены в файл %s\n", FName);
      count++;
    }
    printf("Повторить (y/n)? ");
    key = _getch();
    printf("\n");
  } while (key != 'n' && key != 'N');
  fclose(f1);
  return count;
}
```

Для чтения из файла всей информации о структурах необходимо предварительно определить, сколько структур там записано, затем выделить память под массив структур и считать из файла весь массив.

Для подсчета количества структур нужно определить размер файла (как это делать описано в лабораторной работе 15) и разделить его на размер самой структуры. Лучше оформить этот алгоритм в виде отдельной функции:

```
int CountStructs(FILE* fp)
{
  int len;
  fseek(fp, 0L, SEEK_END);  // Переходим в конец файла
  len = ftell(fp); // Читаем позицию в файле - это будет длина файла
  fseek(fp, 0L, SEEK_SET);    // Переходим снова в начало файла
  return len / sizeof(ManInfo);  // Кол-во структур
}
```

Чтение всего файла будет выглядеть совсем просто:

```
  FILE* f1 = fopen("Struct.dat", "rb");
  size = CountStructs(f1);  // Определяем кол-во структур в файле
  MArr = new ManInfo[size];  // Выделяем память под массив структур
  size = fread(MArr, sizeof(ManInfo), size, f1); // читаем все
  if (size == 0)
    printf("Ошибка при чтении из файла");
  else
    ViewInfo(MArr, size); // Выводим на дисплей
  fclose(f1);
```

Еще в программу нужно добавить ввод имени файла с клавиатуры. Вводить имя с полным путем – очень неудобно, гораздо проще сохранять и читать файл из текущей папки, откуда запущена программа.

Еще желательно добавить меню, которое позволяло бы:

- задавать имя файла для записи,
– вводить информацию с клавиатуры и добавлять в файл,
- задавать имя файла для чтения,
– читать информацию из файла в массив структур,
– отображать информацию на дисплее,
– исправлять данные в массиве структур,
– сохранять исправленные значения в тот же или новый файл.

ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

в начале программы ОБЯЗАТЕЛЬНО выводить:
ФИО, группа, номер лаб. работы, номер варианта.

Задания

Написать программу работы со структурой в соответствии с заданием. В программе должна быть возможность создавать динамический массив структур, задавать информацию в поля структур, сохранять информацию в файл, дописывать новую информацию в файл, читать информацию из файла, отображать информацию в виде таблицы, задавать имена файлов для записи и чтения. Программа должна содержать меню и позволять проводить повторные вычисления. Программа должна быть написана с использованием функций. Вариант задания выдает преподаватель.

Для получения максимального балла нужно выполнить следующие требования:

– программа должна проверять корректность вводимой с клавиатуры информации и не позволять вводить заведомо неверные данные;

– программа должна позволять корректировать прочитанную из файла информацию в выбранных элементах массива и сохранять ее заново в файл,

– должна проводиться проверка на корректность действий при чтении из файла: если файл не прочитан, должно выводиться соответствующее сообщение, а программа не должна позволять корректировать и перезаписывать информацию;

Варианты заданий

Вариант 1

Коэффициенты теплопроводности материалов			
Вещество	Тип	Влажность (%)	Коэффициент
Алюминий	М	0-100	209.3
Стекловата	Т	0-100	0.035
Глина	Д	15-20	0.73
Примечание: М - металлы, Т - термоизоляционные материалы, Д - другие материалы			

Вариант 2

Скорость звука в жидкостях			
Вещество	Тип	Температура (град.С)	Скорость (м/сек)
Анилин	Ч	20	1656
Ртуть	Ч	20	1451
Кедровое	М	29	1406
Тип жидкости: Ч - чистое вещество, М - масло			

Вариант 3

Температура перехода веществ в сверхпроводниковое состояние		
Вещество	Тип	Температура
Zn	М	0.8-0.8
Pb-Au	П	2.0-7.3
NbC	С	10.1-10.5
Тип вещества: М - металл, П - сплав, С - соединение		

Вариант 4

Смартфон				
Производитель	Модель	Память	Оперативная память	Диагональ экрана
Samsung	A50	128	6	6,4
Nokia	4.2	32	3	5,71
Xiaomi	7A	16	2	5,45

Вариант 5

Телевизор				
Производитель	Модель	Диагональ	Разрешение	Цена
Samsung				
Sony				
Xiaomi				

Вариант 6

Фотоаппарат				
Производитель	Модель	Тип камеры	Разрешение	Цена
Panasonic				
Sony				
Nikon				
Тип камеры: З – зеркальная, Б – беззеркальная со сменной оптикой, К – компактная				

Вариант 7

Холодильник				
Производитель	Модель	Объем	Высота	Цена
Bosch				
Samsung				
Indesit				

Вариант 8

Сплавы с высоким сопротивлением			
Сплав	Сопротивление	Темп.коэфф.сопр	Макс.температура
Константан	0.44	0.00001	500
Никелин	0.39	0.39	150
Фехраль	1.1	0.0001	900
Единицы измерения: сопротивление - ом*кв.мм/м. Коэффициент сопротивления - 1/град. Температура - град.С			

Вариант 9

Элементарные частицы			
Частица	Группа	Заряд	Масса покоя
Нейтрон	Н	0	940
Ка-плюс	М	+1	494
Электрон	Л	-1	0.511
Группы частиц: Г - гипероны, Н - нуклоны, М - мезоны, Л - лептоны			

Вариант 10

Вязкость металлов в жидком состоянии			
Вещество	Атомный номер	Температура (град.С)	Вязкость (кг/м*сек)
Алюминий	13	700	2.90
Висмут	83	304	1.65
Свинец	82	441	2.11

Вариант 11

Ведомость комплектующих			
Обозначение	Тип	Номинал	Количество
RT-11-24	R	100000	12
RT-11-24	R	50000	10
CGU-12K	C	17.5	3
Примечание: R - резистор; C – конденсатор; L – индуктивность			

Вариант 12

Датчик				
Измеряемая физическая величина	Единицы измерения	Тип датчика	Мин. предел	Макс. предел
Температура	град. С	R	-50	50
Расход жидкости	куб.м/час	F	0	5
Давление	мм.рт.ст.	I	500	600
Примечание: R – резистивный, F – частотный, I - токовый				

Вариант 13

Ведомость деталей			
Наименование	Тип	Количество	Вес 1 детали (г)
Фланец	З	3	450
Переходник	П	8	74
Станина	О	1	117050
Примечание: принято такое кодирование типов: О - оригинальная, П - покупная, З - заимствованная			

Вариант 14

Ведомость общественного транспорта			
Вид транспорта	N маршрута	Протяженность маршрута (км)	Время в дороге (мин)
Тр	12	27.55	75
Т-с	17	13.6	57
А	12а	57.3	117
Примечание: Тр - трамвай, Тс - троллейбус, А - автобус			

Вариант 15

Прайс-лист			
Наименование товара	Тип товара	Цена за 1 шт (грн)	Минимальное количество в партии
Папка	К	4.75	4
Бумага	К	13.90	10
Калькулятор	О	411.00	1
Примечание: К - канцтовары, О - оргтехника			

Вариант 16

Каталог видеопроката			
Режиссер фильма	Название фильма	Год выпуска	Жанр
Рязанов	Ирония судьбы	1974	К
Кэмерон	Титаник	1997	Д
Мускьетти	Оно	2017	У
Жанр: К – комедия, Д – драма, У – ужасы, М - мелодрама			

Вариант 17

Отдел кадров			
Фамилия	Инициалы	Год рожд	Оклад
Иванов	И.И.	1975	517.50
Петренко	П.П.	1956	219.10
Паниковский	М.С.	1967	300.00

Вариант 18

Каталог библиотеки			
Автор книги	Название	Год выпуска	Группа
Сенкевич	Потоп	1978	Х
Ландау	Механика	1989	У
Дойль	Сумчатые	1990	С
Примечание: Х - художественная литература; У - учебная литература; С - справочная литература			

Вариант 19

Ведомость спортивных состязаний			
Фамилия участника	Код команды	Количество балов	Место в итоге
Баландин	С	123.7	2
Шишков	Ш	79.98	3
Кравченко	Д	134.8	1
Примечание: Д - "Динамо", С - "Спартак", Ш - "Шахтер"			

Вариант 20

Сельскохозяйственные предприятия			
Название	Вид собственности	Площадь земли (га)	Кол. работников
Заря	Г	300	120
Росинка	К	174	27
Петренко	Ч	56	6
Вид собственности: Г - государственная, Ч - частная, К - кооперативная			