Лабораторная работа №6

Использование пользовательских типов данных (структуры, перечисления). Использование динамических структур, строк. Начало работы над проектом.

Цель работы:

1. Изучить методы работы с динамическими структурами

Теоретические сведения

Структурный тип данных.

В простейших задачах каждый элемент данных представлен в виде переменной, определенной встроенным типом float, int, char,но при программировании реальных задач приходится иметь дело с объектами, организованными более сложно, в состав которых входит не одна переменная. В этом случае используется структурный тип данных.

Структура - это совокупность элементов, каждый из которых может иметь любой тип кроме функции. В отличии от массива, который состоит из элементов одинакового типа, структура может состоять из разнотипных элементов.

Приведем простой пример использования структур. Если в программе необходимо использовать дату, то под каждое поле данных необходимо выделить отдельную переменную, например, так:

```
int day;                    // день
char month[10];             // месяц
int year;                   // год
```

Чтобы эти три переменные рассматривались как единое понятие «дата», их требуетсяобъединить в структуру.

Первым шагом в создании взаимосвязанного множества переменных является определение структурного (пользовательского) типа данных. Определение дает структуре имя и указывает компилятору имя и тип каждого элемента (поля) структуры.

Пример: определение нового структурного типа **date,** состоящего из 3-х полей (элементов):

```
struct date
{       int day;
        char month[10];     // статический символьный массив
        int year;
};
```

Синтаксис определения полей структуры аналогичен синтаксису определения переменной -необходимо указать тип данных каждого поля и размер всех строк и массивов.

Список полей структуры не создает переменных, то есть не выделяет никакой памяти, а лишь задает структуру и, тем самымсообщает компилятору состав полей и размер новой структуры (пользовательского типа данный).

Написав определение структуры, можно пользоваться новым структурным типом данных - создавать объекты в памяти - переменные, указатели, массивы, и тому подобное. Название нового типа в нашем примере будет **sruct date**.
Можно совместить описание структуры и создание переменных (выделение памяти), например:

```
struct date
        {       int     day;
                char    month[10];
                int     year;
        } a, b, c;
```

При этом выделяется соответствующая память под переменные a,b,c (каждая из переменных содержит по три поля данных). После того, как новый тип (structdate) введен, его можно использовать для объявления структурныхпеременных в любом месте программы, например:

```
        …
        …
        struct dated1
```

Переменнаяd1имееттот же тип, что и a,b,c (**struct date**).
Переменные, подобные a, b, сназывают объектами структурного типа или же структурными переменными, но чаще всего - структурами.Из-за этого может возникнуть некоторая терминологическая путаница. Обращаю ваше внимание, на то, что определение структуры – это создание нового типа данных (без выделения памяти), а объявление структуры – это выделение памяти под объект структурного типа.

Примеры использования структурного типа для создания различных объектов:

```
struct tovar book;              // переменная-структура book
struct tovar sklad2[1000];      // массив структур sklad2
struct tovar *poin_f ;          // указатель point_f на структуру tovar.
```

Доступ к элементам структуры.

В языке Си с каждым полем приходиться работать, как с самостоятельной переменной. Для доступа к полям структуры есть два специальных оператора:
. (точка), используется, когда работают с самой структурой (прямое обращение по имени);
->(минус и знак "больше"), когда работают с указателем на структуру (косвенное обращение).

Пример:

```
#include <iostream>
#define n 10
using namespace std;
int _tmain (int argc, _TCHAR* argv[])
{       struct date
{       int      day;
        char month[15];
        int year;
};
struct date e_day,s_day,*p1,*p2;
cout<<"Введите дату"<<endl;
cout<<"День:";       cin>>s_day.day;
cout<<"Месяц:";      cin>>s_day.month;
cout<<"Год:";        cin>>s_day.year;
cout<<"_____"<<endl;
p1=&s_day;           // установка указателя
cout<<"Введенная дата"<<endl;
cout<<p1->day<<"/"<<p1->month<<"/"<<p1->year<<endl;
p2=&e_day;   // установкауказателя
p2->day      = 22;
// копирование из одного объекта в другой
memcpy (p2->month, p1->month,15);
 p2-> year    = p1-> year;

        system ("pause");    // задержка экрана
        return 0;
}
```

Обратите внимание, что перед точкой стоит имя объекта, для которого выделена память.
Поля структур ведут себя, как обычные переменные соответствующего типа, можно ставить поля в выражения, вводить с клавиатуры, передавать их функциям, получать адрес поля и так далее.
При объявлении каждой из переменных e_day и s_day будет создано два поля типа int и массив из 10 символов. Выделять память под строку в самой структуре неэкономично, так как строки вводимые с клавиатуры – разной длины, а память выделяется всегда одна и та же. В следующем примере память под строку выделяется динамически. Структура имеет указатель на строку, память же под строку выделяется в процессе работы динамически после того, как становится известна длина строки.

Пример :создать структуру и выделить память под структурные переменные для хранения информации о товаре на складе. Каждая позиция склада содержит разнотипную информацию о товаре, например: название, цену, количество. Создаем новый структурный тип **struct tovar** и выделяем память под переменную с именем **food** :

```
struct tovar
{
char* name ;  //  наименование, указатель на строку
double price ;// цена
int      vol ;   // количество
}food;
```

В следующем примере показано, как работать с полем структуры, которое является текстовой строкой, определенной через указатель.

```
using namespace std;
int _tmain (int argc, _TCHAR* argv[])
{       // определение структурного типа
{
        struct tovar
{       char* name ;
        double price ;
        int vol ;
};
        struct tovar meat;              //определить переменную meat
//_____________________________________________
// Занести информацию в поля переменной meat с клавиатуры
        cout<<"Название:";
        cin>>buff;   // вводстроки склавиатуры в переменную buff
//выделить динамическую память под строку
        meat.name=new char[strlen(buff)+1];
        strcpy(meat.name,buff);  // копировать buff в динамическую память
        cout<<"Цена: "); cin>>meat.price;          // ввод поля «цена»
        cout<<"Количество: "; cin>>meat.vol; // ввод поля «кол-во»
        cout<<"_____"<<endl;
        cout<<"Название:"<<meat.name<<endl;
        cout<<"Цена: "<<meat.price<<endl;
        cout<<"Количество: "<<meat.vol<<endl;
}

        system ("pause");    // задержка экрана
        return 0;
}
```

Перечислим этапы работы с полями-строками, чтобы создать и заполнить с клавиатуры строку для названия товара, необходимо три этапа.
Во-первых информация вводится в промежуточную (статическую) строку **buff**, без этой дополнительной строки невозможно определить какой объем памяти запрашивать из кучи. Затем, с помощью оператора new выделяется необходимая динамическая память под уже известную текстовую строку. Затем информация из **buff** копируется в динамическую память с помощью функции **strcpy().**После того, как информация введена во все поля объекта **meat**, она выводится на экран терминала.

Контрольные вопросы

1. Для чего используется динамическая память в программировании?
2. Как долго хранятся данные в динамической памяти?
3. Какие возможны варианты доступа к динамической памяти?
4. Что возвращает операция выделения динамической памяти в случае успешного выполнения?
5. Что возвращает операция выделения динамической памяти, если память требуемого размера не может быть выделена?
6. Почему тип функций выделения динамической памяти определен как *void?
7. Почему при завершении работы с динамической памятью ее необходимо освободить? Какие могут быть последствия для работы программы, если не освобождать динамическую память?
8. Приведите пример статического выделения памяти под переменную.
9. Приведите пример динамического выделения памяти под переменную.
10. В чем основное различие статического и динамического массива.
11. Какой оператор выделяет динамическую память, приведите пример.
12. Какой оператор освобождает динамическую память, приведите пример.
13. Перечислите последовательность действий при использовании структур.
14. Как выделить память под структурный объект?
15. Опишите два способа для обращения к полям структуры?

Общие требования к выполнению заданий

1. При оформлении ввода-вывода данных информация на экране должна быть отформатирована:
   - на экран выводится тема задания (кратко);
   - ввод данных и результат вычислений выводить с комментариями;
   - выделять области ввода и вывода информации с помощью строк-разделителей.
2. Данные размещаются в динамической памяти.
3. Обязательные функции для всех вариантов:
   - добавить новый
   - распечатка данных в табличном виде
   - выход из программы
4. Остальные функции указаны в задании индивидуально.
5. Для выполнения функций, указанных в задании, написать диалоговый интерфейс, позволяющий выполнять функции в произвольном порядке многократно
6. Первичное создание базы – ввод данных с клавиатуры
7. При реализации функций в параметрах и возвращаемых значениях использовать указатели и ссылки

Пример диалогового интерфейса
(База данных «Склад товаров»)

Добавить новый элемент …… 1
Распечатать базу товаров …… 2
Поиск товара по названию …… 3
Фильтр по цене …….. 4
Выход из программы …….. 5
……..
Введите номер функции

Пример распечатки данных в табличном виде
(База данных «Склад товаров»)

Название Товара	**Цена (руб)**	**Количество (кг)**	**Общая сумма (руб)**
Сыр «Российский»	560	26.5	14840.00
Масло сливочное	380.5	100.25	38145.12
Рис длинный	68	25.0	1700.00
Рис круглый	62	56.75	3518.50
Всего товаров на сумму			58203.62

Количество записей в базе 4

Вы начинаете работу над проектом, которая будет продолжаться до конца семестра (лаб. №6,7)

<table>
<tr><th>Номер компьютера</th><th>Задание для групп</th></tr>
<tr><td>1,13,25</td><td>
Проект: ВУЗ (Студент)

Создайте структуру student со следующими полями:
<table>
<tr><td>Тип данных</td><td>Назначение поля</td></tr>
<tr><td>char*</td><td>Фамилия студента</td></tr>
<tr><td>char*</td><td>Имя студента</td></tr>
<tr><td>char*</td><td>Направление подготовки (ИВТ,УТС и т.п.)</td></tr>
<tr><td>int</td><td>Номер группы</td></tr>
</table>
Создать массив из n студентов (ввод с клавиатуры).
Минимальный массив – 8 студентов, должно быть не менее:
– 2-х направлений подготовки;
– по 2 группы в каждом направлении;
– в каждой группе по 2 студента.
Реализовать функции :
– добавить информацию о новом студенте;
– распечатать информацию о студенте в табличном виде;
– определить всех студентов по заданному направлению подготовки, результат вывести на экран;
– определить всех студентов заданной группы (группа это направление подготовки+ номер группы), результат отсортировать по алфавиту, запомнить в массиве и вывести на экран;
</td></tr>
<tr><td>2,14,26</td><td>
Проект: Склад (товары)

Создайте структуру tovar со следующими полями:
<table>
<tr><td>Тип данных</td><td>Назначение поля</td></tr>
<tr><td>int</td><td>номер секции</td></tr>
<tr><td>char*</td><td>Название товара</td></tr>
<tr><td>double</td><td>цена</td></tr>
<tr><td>int</td><td>количество</td></tr>
</table>
Создать массив из n позиций товаров (ввод с клавиатуры)
Реализовать функции :
– добавить товар (товар – это название + цена).
Если товар уже есть, то добавляем количество, иначе создаем новую позицию ;
– распечатать информацию о товаре в табличном виде;
– найти все товары заданной секции, результат отсортировать по алфавиту, запомнить в массиве и вывести на экран;
– определить товары с количеством меньше заданного, результат отсортировать по алфавиту, запомнить в массиве и вывести на экран;
</td></tr>
</table>

3,15,27

Проект: Транспорт (пассажир самолета)

Создайте структуру **plane** со следующими полями:

Тип данных	Назначение поля
char*	компания
char*	фамилия пассажира
char*	имя пассажира
int	рейс
double	Стоимость билета

Создать массив из n пассажиров (ввод с клавиатуры)
Реализовать функции :

- добавить нового пассажира;
- распечатать информацию о пассажире в табличном виде;
- Найти все рейсы заданной компании, результат вывести на экран;
- Найти всех пассажиров заданного рейса, результат сортировать по алфавиту, запомнить в массиве и вывести на экран;

4,16,28

Проект: Банк (депозит)

Создайте структуру **deposid** со следующими полями:

Тип данных	Назначение поля
int	номер счета
char*	Фамилия клиента
char*	Имя клиента
double	Сумма на счете

Создать массив из n клиентов банка (ввод с клавиатуры)
Реализовать функции :

- добавить новый счет;
- распечатать информацию о счете в табличном виде;
- определить все счета с суммой больше заданной, результат вывести на экран;
- Найти все счета заданного клиента (имя + фамилия), результат запомнить в массиве и вывести на экран;

5,17,29

Проект: Библиотека (статья в журнале)

Создайте структуру **artical** со следующими полями:

Тип данных	Назначение поля
char*	название журнала
int	номер
int	год
char*	Фамилия автора
char*	Название статьи

Создать массив из n статей (ввод с клавиатуры)

Реализовать функции :

- добавить новую статью;
- распечатать информацию о статье в табличном виде;
- найти все статьи заданного автора, результат вывести на экран;
- Найти всех авторов заданного журнала (журнал – это название + номер + год), результат отсортировать по алфавиту, запомнить в массиве и вывести на экран;

6,18,30

Проект: Почта (ценное письмо)

Создайте структуру **letter** со следующими полями:

Тип данных	Назначение поля
double	Оценка письма
int	Индекс отправителя
char*	Фамилия отправителя
char*	Имя отправителя
int	Индекс получателя
char*	Фамилия получателя
char*	Имя получателя

Реализовать функции :

- добавить новое письмо;
- распечатать информацию о письме в табличном виде;
- Найти все письма заданного отправителя (отправитель – это фамилия + имя) результат вывести на экран;
- найти все письма с оценкой большей заданного, результат сортировать по алфавиту (по получателю), запомнить в массиве и вывести на экран;

7,19

Проект: Библиотека (книга)

Создайте структуру **book** со следующими полями:

Тип данных	Назначение поля
char*	фамилия автора
char*	имя автора
char*	название книги
char*	издательство
int	год издания

Создать массив из n книг (ввод с клавиатуры)
Реализовать функции :

- добавить новую книгу;
- распечатать информацию о книге в табличном виде;
- найти все книги заданного автора, результат вывести на экран;
- найти всех авторов заданного издательства, результат отсортировать по алфавиту, запомнить в массиве и вывести на экран;

8,20

Проект: Банк (кредит)

Создайте структуру **credit** со следующими полями:

Тип данных	Назначение поля
int	номер счета
char*	фамилия клиента
char*	имя клиента
double	сумма кредита
int	процент по кредиту

Создать массив из n клиентов банка (ввод с клавиатуры)
Реализовать функции :

- добавить новый счет;
- распечатать информацию о счете в табличном виде;
- Найти все счета и общую сумму кредитов у заданного клиента (клиент – это фамилия + имя), результат вывести на экран;
- Определить всех клиентов с заданным процентом по кредиту, результат запомнить в массиве и вывести на экран;

9,21

Проект: Транспорт (машина)

Создайте структуру **car** со следующими полями:

Тип данных	Назначение поля
char*	марка машины
char*	фамилия владелеца
char*	имя владелеца
int	мощность двигателя
int	пробег

Создать массив из n машин (ввод с клавиатуры)

Реализовать функции :

- добавить новую машину;
- распечатать информацию о машине в табличном виде;
- Найти все машины заданной марки, результат вывести на экран;
- Найти всех владельцев с пробегом машины больше заданного, результат сортировать по алфавиту, запомнить в массиве и вывести на экран;

10,22

Проект: ВУЗ (преподаватель)

Создайте структуру **prepod** со следующими полями:

Тип данных	Назначение поля
char*	Фамилия преподавателя
char*	Имя преподавателя
char*	Институт/кафедра (СПИНТех, БМС, ВМ1 и т.п.)
char*	Дисциплина (ОП, ООП, физика и т.п.)

Создать массив из n преподавателей (ввод с клавиатуры)

Реализовать функции :

- добавить нового преподавателя;
- распечатать информацию о преподавателе в табличном виде;
- Найти все предметы, которые ведет заданный преподаватель (преподаватель – это фамилия + имя) результат вывести на экран;
- найти всех преподавателей заданного института/кафедры, результат сортировать по алфавиту, запомнить в массиве и вывести на экран;

11,23

Проект: Склад (поставщики)

Создайте структуру **supplier** со следующими полями:

Тип данных	Назначение поля
char*	Название фирмы
char*	Название товара
double	цена
int	количество

Создать массив из n позиций товаров (ввод с клавиатуры)
Реализовать функции :

- добавить товар (товар – это название + цена).

Если товар уже есть, то добавляем количество, иначе создаем новую позицию ;

- распечатать информацию о товаре в табличном виде;
- определить товары с количеством меньше заданного, результат вывести на экран;
- найти все товары заданной фирмы, результат отсортировать по алфавиту, запомнить в массиве и вывести на экран;

12,24

Проект: Почта (посылка)

Создайте структуру **parcel** со следующими полями:

Тип данных	Назначение поля
double	Вес посылки
char*	Фамилия отправителя
char*	Имя отправителя
char*	Фамилия получателя
char*	Имя получателя
char*	Адрес получателя

Реализовать функции :

- добавить новую посылку;
- распечатать информацию о посылке в табличном виде;
- Найти все посылки заданного отправителя (отправитель – это фамилия + имя) результат вывести на экран;
- найти все посылки с весом больше заданного, результат сортировать по алфавиту (по получателю), запомнить в массиве и вывести на экран;