**Лабораторная работа №6**

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

***Цель работы:***

1. Изучить методы работы c 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.

**Cтруктура** - 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, cназывают 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**

usingnamespace **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.

usingnamespace **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 cклавиатуры в переменную buff

//выделить динамическую память под строку

**meat.name=**newchar[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.

1. Массив структур (базу данных) размещать в динамической памяти.
2. Строки в структурах располагать в динамической памяти.
3. Обязательные функции для всех вариантов:

* добавить новый элемент массива
* распечатка данных в табличном виде
* выход из программы

1. Остальные функции указаны в задании индивидуально.
2. Для выполнения функций, указанных в задании, написать диалоговый интерфейс, позволяющий выполнять функции в произвольном порядке многократно
3. Первичное создание базы – ввод данных с клавиатуры
4. При реализации функций в параметрах и возвращаемых значениях использовать указатели и ссылки
5. **Вы начинаете работу над проектом, которая будет продолжаться до конца семестра (лаб. №6,7)**

**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 Цена Количество Общая сумма**

**Товара (руб) (кг) (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

|  |  |
| --- | --- |
| **Номер компью-тера** | **Задание** |
| 1, 11, 21 | База данных: Пищевое производство (расчеты).  Создать базу данных со следующей структурой :  char\* name ; //Название изделия  int quant; // количество  double cost; //стоимость  Смотрите «Общие требования к выполнению заданий»  Создать обязательные функции, указанные в общих требованиях.  Создать функции для данного варианта:   * Поиск изделий по названию * Найти все изделия больше заданной стоимости * Сортировать базу по возрастанию количества |
| 2, 12, 22 | База данных: Поезд (заказ билетов).  Создать базу данных со следующей структурой :  char\* fam; //Фамилия пассажира  char\* name; // Имя пассажира  char\* arrival; // пункт прибытия  double price; // цена билета  Смотрите «Общие требования к выполнению заданий»  Создать обязательные функции, указанные в общих требованиях. Создать функции для данного варианта:   * Поиск билета по имени и фамилии пассажира * Найти все билеты с ценой более заданного значения * Сортировать базу по фамилии пассажира (по алфавиту) |

|  |  |
| --- | --- |
| 3, 13, 23 | База данных: Почта (ценное письмо).  Создать базу данных со следующей структурой :  char\* adr1; //адрес получателя  char\* fam1; //фамилия получателя  char\* name1; // имя получателя  char\* adr2; // адрес отправителя  char\* fam2; //фамилия отправителя  char\* name2; // имя отправителя  double cost; // стоимость письма  Смотрите «Общие требования к выполнению заданий»  Создать обязательные функции, указанные в общих требованиях. Создать функции для данного варианта:   * Поиск письма по полям fam2, name2 (фамилия и имя отправителя) * Найти все письма со стоимостью более заданного значения * Сортировать базу по убыванию стоимости |
| 4, 14,  24 | База данных: Студент.  Создать базу данных со следующей структурой :  char\* fam; //фамилия студента  char\* name; // имя студента  int grup // номер группы  Смотрите «Общие требования к выполнению заданий»  Создать обязательные функции, указанные в общих требованиях. Создать функции для данного варианта:   * Поиск студента по фамилии и имени * Найти всех студентов заданной группы * Сортировать базу по фамилии студента (по алфавиту) |
| 5, 15,  25 | База данных: Производство (поставщики сырья)  Создать базу данных со следующей структурой :  char\* firma; // название фирмы поставщика  double raw\_ material // сумма поставки сырья  double payment; // оплата сырья  Смотрите «Общие требования к выполнению заданий»  Создать обязательные функции, указанные в общих требованиях.  Создать функции для данного варианта:   * Поиск поставщика по названию фирмы * Найти поставщиков, которые получили не полную оплату за сырье * Сортировать базу по возрастанию суммы поставки |
| 6, 16,  26 | База данных: Поликлиника.  Создать базу данных со следующей структурой :  char\* fam; //фамилия врача  char\* name; // имя врача  enum specialty; // специальность (1-терапевт, 2-кардиолог, 3- стоматолог)  Смотрите «Общие требования к выполнению заданий»  Создать обязательные функции, указанные в общих требованиях. Создать функции для данного варианта:   * Поиск врача по фамилии и имени * Найти всех врачей заданной специальности * Сортировать базу по фамилии врача (по алфавиту) |
| 7, 17,  27 | База данных: Склад (товары).  Создать базу данных со следующей структурой :  char\* name; //Название товара  double price; // цена  int quant; // количество  Смотрите «Общие требования к выполнению заданий»  Создать обязательные функции, указанные в общих требованиях.  Создать функции для данного варианта:   * Поиск товара по названию * найти все товары с количеством меньше заданного * Сортировать базу по убыванию цены |
| 8, 18,  28 | База данных: Банк (депозит).  Создать базу данных со следующей структурой :  char\* fam; // фамилия клиента  char\* name; // имя клиента  double sum; // Сумма на счете (сумма депозита )  Смотрите «Общие требования к выполнению заданий»  Создать обязательные функции, указанные в общих требованиях.  Создать функции для данного варианта:   * Поиск клиента по фамилии и имени * Найти клиентов с суммой на счете больше заданной * Сортировать базу по фамилии клиента (по алфавиту) |

|  |  |
| --- | --- |
| 9,  19,  29 | База данных: Работник (зарплата).  Создать базу данных со следующей структурой :  char\* fam; // фамилия работника  char\* name; // имя работника  int department; // отдел  double salary; // оклад  Смотрите «Общие требования к выполнению заданий»  Создать обязательные функции, указанные в общих требованиях.  Создать функции для данного варианта:   * Поиск работника по фамилии и имени * Найти всех работников заданного отдела * Сортировать базу по возрастанию оклада |
| 10, 20, 30 | База данных: Преподаватель.  Создать базу данных со следующей структурой :  char\* fam; //фамилия преподавателя  char\* name; // имя преподавателя  enum status; //ученое звание ( 1- преподаватель, 2- доцент, 3-доктор наук)  Смотрите «Общие требования к выполнению заданий»  Создать обязательные функции, указанные в общих требованиях.  Создать функции для данного варианта:   * Поиск преподавателя по имени и фамилии * Найти всех преподавателей заданного ученого звания (status) * Сортировать базу по фамилии преподавателя (по алфавиту) |