Лабораторная работа № 1

Динамические структуры данных.

**Цель работы:** Получить практические навыки решения задач на компьютере с использованием динамических данных, а такжеработы со сложными (пользовательскими) типами данных – динамическими структурами.

## Теоретические сведения

Для программы наС++ существует два способа хранения данных в основной памяти компьютера. Первый состоит в том, что память, предоставляемая под переменные, выделяется им при компиляции и не может быть изменена при выполнении программы. Это статическое распределение памяти. Второй способ заключается в использовании динамического распределения памяти, то есть получение памяти во время выполнения программы.

Динамическая работа с памятью дает программисту широкие возможности - программа может создавать переменные в нужном количестве, в зависимости от ситуации. Кроме того, динамическое выделение памяти для тех или иных целей – важная составляющая почти всех реальных программ.

Есть такие задачи, которые невозможно реализовать без динамического выделения памяти, например требуется изменение размера памяти по мере использования или размер памяти заранее не известен и определяется в ходе работы программы.

**Распределение памяти компьютера.**

|  |  |
| --- | --- |
| Верхние адреса памяти | Стек, локальные переменные и константы. |
|  | Динамическая память (Куча) |
|  | Глобальные переменные и константы |
| Нижние адреса памяти | Программа пользователя |

Операционная система позволяет получать память динамически«из кучи» во время выполнения программы.

**Динамическое выделение памяти для отдельных переменных.**

Язык С++ имеет два оператора, которые выполняют работу по выделению и освобождению памяти :

* Оператор ***new***выделяет необходимую память из «кучи».

Формат оператора :***Переменная-указатель = new тип\_переменной;***

* Оператор ***delete***освобождает ранее выделенную динамическую память (возвращает ее в «кучу»).

Формат оператора**:*deleteпеременная-указатель;***

Язык С++ позволяет инициализировать динамическую память в момент её выделения из «кучи», для этого начальное значение, которое нужно присвоить при инициализации заключается в круглые скобки.

Формат оператора :

***Переменная-указатель = new тип\_переменной ();***

**Пример1.** Статическое распределение памяти под переменные.

Память под переменные x,yи z выделяется при трансляции, к ним можно обращаться по именам, так как адреса переменных фиксированы.

#include <iostream>

#define n 100

using namespace std;

void main()

{ int x,y;

double z;

x=n;

do

{

cout<<"y=";

cin>>y;

}

while(!y);

z=double(x)/y;

cout<<"\nz="<<z<<'\n';

}

В следующем примере память, необходимую для переменных x и y, выделим динамически, то есть в момент работы программы.

Пример2. Под две переменные типаintпамять выделяется динамически (числитель и знаменатель дроби. Результат вычисления записывается в статическую переменную z.

#include<iostream>

usingnamespacestd;

voidmain()

{ int\*p\_x,\*p\_y; // определить указатели p\_xи p\_y

doublez;

p\_x=newint() ; /\* выделить динамическую память под переменную типаint и записать её адрес в указатель p\_x\*/

p\_y=newint(); /\* выделить дин. память (адрес в указателе p\_y)\*/

cout<<"x=";

cin>>\*p\_x; // ввести данные с клавиатуры в динамическую переменную

do// ввести ненулевые данные с кл-ры в динамическую переменную

{

cout<<"y=";

cin>>\*p\_y;

}

while(!\*p\_y);

z=double(\*p\_x)/(\*p\_y);

cout<<"\nz="<<z<<'\n';

deletep\_x; // освободить динамическую память

deletep\_y;

system ("pause"); // задержкаэкрана

}

Память под переменнуюz выделяется при трансляции, поэтому к ней можно обращаться по имени. Память под переменные для числителя и знаменателя дроби, выделяется динамически и, поэтому эти данные не имеют имён , так как их адреса в момент трансляции неизвестны. К этим переменным можно обращаться только через указатели, в нашем примере – это указателиp\_xи p\_y.

**Динамическое выделение памяти для массивов.**

С помощью оператора можно выделять память и для массивов. Вот как выглядит общий формат операции выделения памяти для одномерного массива :***Переменная-указатель = new тип\_переменной [размер];***

Чтобы освободить память, выделенную для динамически созданного массива, используется такой формат оператора ***delete***:

***delete [] переменная\_указатель***

Следует отметить, что инициализировать динамический массив одновре6менно с выделением памяти ***нельзя***.

Пример 3.Выделить память для 10 элементов массива типа double, который затем инициализировать значениями от 1 до 10. Отобразить содержимое массива на экране.

Напомним, что в языке С++ имя массива – это указатель на первый элемент массива, то есть «природа» указателя и имени массива одинакова. Отсюда следует, что указатель можно индексировать, подобно массиву.

#include <iostream>

#define n 10

using namespace std;

void main()

{ double \*p\_m;

int i;

p\_m = newdouble[n]; // выделяем память для 10-ти элементов массива

for(i=0;i<n ;i++) p\_m[i] = 1+i; // инициализируеммассив

for(i=0;i<n;i++) cout<<p\_m[i]<<' '; // выводим массив на экран

delete[] p\_m; /\* удаляем весь массив

(возвращаем память в «кучу») \*/

}

Следующий пример показывает, как ту же самую задачу можно решить без использования индексов. Почему следует избегать использования индексов? Дело в том, что при индексировании массива перед обращением к очередному элементу каждый раз вычисляется его адрес, путем прибавления к начальному адресу массива (имени) смещения (индекса). При использовании указателей используются операции инкремент/декремент, которые выполняются значительно быстрее, что и является решающим фактором при обработке больших объемов данных.

В примере 4 для обращения к элементам динамического массива данных используются два указателя – p\_beg и p\_tek. Оператор ***new***возвращает адрес выделенной области памяти, который записывается в указатель p\_beg, этот указатель служит в дальнейшем для хранения этого адреса (начало динамического массива). Для обращения к элементам массива служит «текущий» указатель p\_tek.

Пример 4. Работа с динамическим массивом без использования индексов.

#include <iostream>

#define n 10

using namespace std;

void main()

{ double \*p\_beg,\*p\_tek;

int i;

// выделяем память для 10-ти элементов массива

p\_beg = newdouble[10];

// устанавливаем «текущий» указатель на начало массива

p\_tek=p\_beg;

/\* цикл для заполнения массива, «текущий» указатель перемещается на следующий элемент массива при каждом проходе цикла \*/

for(i=1;p\_tek<p\_beg+10;p\_tek++)

\*p\_tek = i++;

/\* снова устанавливаем «текущий» указатель на начало массива, так как после окончания цикла заполнения массива он указывает на конец массива \*/

p\_tek=p\_beg;

while(p\_tek<p\_beg+10) // цикл для отображения массива

cout<< \*p\_tek++<<' ';

delete[] p\_beg; //возвращаем память в «кучу»

}

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

intyear; // 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):

sructdate

{ intday;

charmonth[10]; // статический символьный массив

intyear;

};

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 **sructdate**.

Можно совместить описание структуры и создание переменных (выделение памяти), например:

structdate

{ intday;

charmonth[10];

intyear;

} 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:

…

…

structdated1;

Переменнаяd1имееттот же тип, что и a,b,c (**structdate**).

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:

structtovarbook; // переменная-структура book

structtovarsklad2[1000]; // массив структур sklad2

structtovar\*poin\_f ; // указательpoint\_fна 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 <stdio.h>

void main()

{struct date

{ int day;

char month[15];

int year;

};

struct date e\_day,s\_day,\*p1,\*p2;

cout<<"Введитедату"<<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;

}

Обратите внимание, что перед точкой стоитимя 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** :

structtovar

{

char\* name ;/ / наименование, указатель на строку

double price ;/ / цена

intvol ; / / количество

}food;

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

#include <iostream>

using namespace std;

void main()

// определение структурного типа

{

structtovar

{ char\* name ;

double price ;

int vol ;

};

structtovarmeat; //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;

}

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. Приведите пример статического выделения памяти под переменную.
2. Приведите пример динамического выделения памяти под переменную.
3. В чем основное различие статического и динамического массива.
4. Какой оператор выделяет динамическую память, приведите пример.
5. Какой оператор освобождает динамическую память, приведите пример.
6. Перечислите последовательность действий при использовании структур.
7. Как выделить память под структурный объект?
8. Опишите два способа для обращения к полям структуры?

**Общие требования**

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.

При печати данных о структуре использовать табличную форму, включающую первую строку-комментарий и последующие строки с информацией например:

Автор Название Издательство Тираж

-------------------------------------------------------------------------------------

Булгаков М.А. Мастер и Маргарита Эксмо 5000

Акунин Б.Ш. Азазель Вагриус 1000

1. После ввода данных с клавиатуры предусмотреть многократное выполнение действий с ними. Например, при поиске фамилии, начинающейся на заданную букву дать возможность многократно выполнить поиск с разными значениями. При этом после каждого поиска задавать вопрос «Продолжить выполнение?»
2. Использование пользовательских функций.

## Варианты заданий

|  |  |
| --- | --- |
| Вариант |  |
| 1,9,17,25 | Создать структуру для электронного паспорта:  фамилия, имя отчество, адрес.  Ввести с клавиатуры данные для нескольких паспортов. Распечатать паспорта с фамилиями, начинающимися на заданную букву. Подсчитать количество паспортов с заданным именем. |
| 2,10,18,26 | Создать структуру для электронного каталога книг: Автор, название, издательство, тираж. Ввести с клавиатуры данные о несколькихкнигах. Узнать, есть ли среди них книги заданного автора, если есть, вывести данные на экран. Распечатать книги с тиражом больше заданного. |
| 3,11,19,27 | Создать структуру для электронного ценника :товар, цена, секция магазина (строка). Ввести с клавиатуры данные о нескольких ценниках. Распечатать информацию о ценниках заданного товара. Подсчитать общую сумму товаров заданной секции. |
| 4,12,20 | Создать структуру для базы данных авиабилетов :Компания, пункт назначения, номер рейса, стоимость билета. Ввести с клавиатуры данные о несколькихпроданных билетах. Распечатать данные о проданных билетах заданной компании. Вывести данные о количестве проданных билетов на указанный рейс. |
| 5,13,21 | Создать структуру для электронного каталога книг:Автор, название, год издания. Ввести с клавиатуры данные о нескольких книгах. Узнать кто автор книги с заданным названием. Вывести данные о книгах, изданных после указанного года. |
| 6,14,22 | Создать структуру для электронного ценника (товар, цена, количество, сумма). Ввести с клавиатуры данные о нескольких ценниках (товар может быть с одинаковым названием, но с разной ценой). Вывести информацию о товарах с ценой меньше указанной. Подсчитать общую сумму указанного товара. |
| 7,15,23 | Создать структуру для электронного паспорта:  фамилия, имя отчество, год рождения.  Ввести с клавиатуры данные для нескольких паспортов. Распечатать паспорта с годами рождения меньше заданного.Определить есть ли однофамильцы, если есть, то распечатать по ним полную информацию. |
| 8,16,24 | Создать структуру для базы данных авиабилетов (Компания, пункт назначения, номер рейса, стоимость билета). Ввести с клавиатуры данные о несколькихпроданных билетах. Распечатать данные о проданных билетах до заданного пункта назначения. Подсчитать суммарную стоимость билетов на заданный рейс. |