Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

Лабораторная работа 4

По дисциплине «Структуры данных»

На тему «Индексировние записей. Простой/сложный индекс»

Выполнила:

Студентка 1 курса 2 группы

Глухова Д.В.

Преподаватель: Север А.С.

2023, Минск

Операция создания заключается в выделении памяти для структуры данных. Память может выделяться в процессе выполнения программы при первом появлении имени переменной в исходной программе или на этапе компиляции.

Главное заключается в том, что независимо от используемого языка программирования, имеющиеся в программе структуры данных не появляются "из ничего", а явно или неявно объявляются операторами создания структур. В результате этого всем структурам программы выделяется память для их размещения.

Операция уничтожения структур данных противоположна по своему действию операции создания.

Операция выбора используется программистами для доступа к данным внутри самой структуры. Форма операции доступа зависит от типа структуры данных, к которой осуществляется обращение. Метод доступа - один из наиболее важных свойств структур, особенно в связи с тем, что это свойство имеет непосредственное отношение к выбору конкретной структуры данных.

Операция обновления позволяет изменить значения данных в структуре данных. Примером операции обновления является операция присваивания или более сложная форма - передача параметров.

При структурировании больших программных изделий возможно применение подхода, основанного на структуризации алгоритмов и известного как "нисходящее" проектирование или "программирование сверху вниз", или подхода, основанного на структуризации данных и известного как "восходящее" проектирование или "программирование снизу вверх".

В первом случае структурируют прежде всего действия, которые должна выполнять программа. Большую и сложную задачу, стоящую перед проектируемым программным изделием, представляют в виде нескольких подзадач меньшего объема. Таким образом, модуль самого верхнего уровня, отвечающий за решение всей задачи в целом, получается достаточно простым и обеспечивает только последовательность обращений к модулям, реализующим подзадачи.

Другой подход к структуризации основывается на данных. Таким образом, задачей любого программного изделия является преобразование входных данных в выходные. Инструментальные средства программирования предоставляют набор базовых (простых, примитивных) типов данных и операции над ними. Интегрируя базовые типы, создаются более сложные типы данных и определяются новые операции над сложными типами. Можно здесь провести аналогию со строительными работами: базовые типы - "кирпичики", из которых создаются сложные типы - "строительные блоки".

Еще одним чрезвычайно продуктивным технологическим приемом, связанным со структуризацией данных, является инкапсуляция. Смысл ее состоит в том, что сконструированный новый тип данных - "строительный блок" - оформляется таким образом, что его внутренняя структура становится недоступной для программиста-пользователя этого типа. Программист, использующий этот тип данных в своей программе (в модуле более высокого уровня), может оперировать с данными этого типа только через вызовы функций, определенных для этого. Новый тип данных представляется для него в виде "черного ящика", для которого известны входы и выходы, но содержимое - неизвестно и недоступно.

# Массивы структур

Структуры часто образуют массивы. Чтобы объявить массив структур, вначале необходимо определить структуру (то есть определить агрегатный тип данных), а затем объявить переменную массива этого же типа. Например, чтобы объявить 100-элементный массив структур типа addr, который был определен ранее, напишите следующее:

struct addr addr\_list[100];

Вариант 14

Вы являетесь сотрудником коммерческого отдела компании, продающей различные товары через Интернет. Вашей задачей является отслеживание финансовой составляющей работы компании.  
Работа Вашей компании организована следующим образом: на Интернет-сайте компании представлены (выставлены на продажу) некоторые товары. Каждый из них имеет некоторое название, цену и единицу измерения (штуки, килограммы, литры).  
Для проведения исследований и оптимизации работы магазина Вы пытаетесь собирать данные с Ваших клиентов. При этом для Вас определяющее значение имеют стандартные анкетные данные, а также телефон и адрес электронной почты для связи.  
По каждому факту продажи Вы автоматически фиксируете клиента, товары, количество, дату продажи, дату доставки.

|  |
| --- |
| #include <iostream>  # define str\_len 40  # define size 40  using namespace std;  void del();  void enter\_new();  void change();  void out();  struct Client  {  char fio[str\_len];  char telephone[str\_len];  char email[str\_len];  char product[str\_len];  int numProduct;  char dateSale[str\_len];  char dateDelivery[str\_len];  };  struct Client list\_of\_clients[size];  struct Client bad;  int current\_size = 0; int choice;  int main()  {  setlocale(LC\_CTYPE, "Russian");  cout << "Введите:" << endl;  cout << "1-для удаления записи" << endl;  cout << "2-для ввода новой записи" << endl;  cout << "3-для изменения записи" << endl;  cout << "4-для вывода записи(ей)" << endl;  cout << "5-для выхода" << endl;  cin >> choice;  do  {  switch (choice)  {  case 1: del(); break;  case 2: enter\_new(); break;  case 3: change(); break;  case 4: out(); break;  }  } while (choice != 5);  }  void enter\_new()  {  cout << "Ввод информации" << endl;  if (current\_size < size)  {  cout << "Строка номер ";  cout << current\_size + 1;  cout << endl << "Фамилия клиента: " << endl;  cin >> list\_of\_clients[current\_size].fio;  cout << "Номер телефона: " << endl;  cin >> list\_of\_clients[current\_size].telephone;  cout << "Электронная почта: " << endl;  cin >> list\_of\_clients[current\_size].email;  cout << "Товар: " << endl;  cin >> list\_of\_clients[current\_size].product;  cout << "Количество товаров: " << endl;  cin >> list\_of\_clients[current\_size].numProduct;  cout << "Дата продажи: " << endl;  cin >> list\_of\_clients[current\_size].dateSale;  cout << "Дата доставки: " << endl;  cin >> list\_of\_clients[current\_size].dateDelivery;  current\_size++;  }  else  cout << "Введено максимальное кол-во строк";  cout << endl << "Введите:" << endl;  cout << "1-для удаления записи" << endl;  cout << "2-для ввода новой записи" << endl;  cout << "3-для изменения записи" << endl;  cout << "4-для вывода записи(ей)" << endl;  cout << "5-для выхода" << endl;  cin >> choice;  }  void del()  {  int d;  cout << "\nНомер строки, которую надо удалить (для удаления всех строк нажать 99)" << endl; cin >> d;  if (d != 99)  {  for (int de1 = (d - 1); de1 < current\_size; de1++)  list\_of\_clients[de1] = list\_of\_clients[de1 + 1];  current\_size = current\_size - 1;  }  if (d == 99)  for (int i = 0; i < size; i++)  list\_of\_clients[i] = bad;  cout << endl << "Введите:" << endl;  cout << "1-для удаления записи" << endl;  cout << "2-для ввода новой записи" << endl;  cout << "3-для изменения записи" << endl;  cout << "4-для вывода записи(ей)" << endl;  cout << "5-для выхода" << endl;  cin >> choice;  }  void change()  {  int n, per;  cout << "\nВведите номер строки" << endl; cin >> n;  do  {  cout << "Введите: " << endl;  cout << "1-для изменения фамилии" << endl;  cout << "2-для изменения номера телефона" << endl;  cout << "3-для изменения номера почты" << endl;  cout << "4-для изменения номера товара" << endl;  cout << "5-для изменения количества товаров" << endl;  cout << "6-для изменения даты продажи" << endl;  cout << "7-для изменения даты доставки" << endl;  cout << "0-конец\n";  cin >> per;  switch (per)  {  case 1: cout << "Новая фамилия";  cin >> list\_of\_clients[n - 1].fio; break;  case 2: cout << "Новый номер телефона";  cin >> list\_of\_clients[n - 1].telephone; break;  case 3: cout << "Новая почта";  cin >> list\_of\_clients[n - 1].email; break;  case 4: cout << "Новый товар";  cin >> list\_of\_clients[n - 1].product; break;  case 5: cout << "Новое количество товаров";  cin >> list\_of\_clients[n - 1].numProduct; break;  case 6: cout << "Новая дата продажи";  cin >> list\_of\_clients[n - 1].dateSale; break;  case 7: cout << "Новая дата доставки";  cin >> list\_of\_clients[n - 1].dateDelivery; break;  }  } while (per != 0);  cout << endl << "Введите:" << endl;  cout << "1-для удаления записи" << endl;  cout << "2-для ввода новой записи" << endl;  cout << "3-для изменения записи" << endl;  cout << "4-для вывода записи(ей)" << endl;  cout << "5-для выхода" << endl;  cin >> choice;  }  void out()  {  int sw, n;  cout << "1-вывод 1 строки" << endl;  cout << "2-вывод всех строк" << endl;  cin >> sw;  if (sw == 1)  {  cout << "Номер выводимой строки " << endl; cin >> n; cout << endl;  cout << "Фамилия ";  cout << list\_of\_clients[n - 1].fio << endl;  cout << "Телефон ";  cout << list\_of\_clients[n - 1].telephone << endl;  cout << "Почта ";  cout << list\_of\_clients[n - 1].email << endl;  cout << "Товар ";  cout << list\_of\_clients[n - 1].product << endl;  cout << "Количество товаров ";  cout << list\_of\_clients[n - 1].numProduct << endl;  cout << "Дата продажи ";  cout << list\_of\_clients[n - 1].dateSale << endl;  cout << "Дата доставки ";  cout << list\_of\_clients[n - 1].dateDelivery << endl;  }  if (sw == 2)  {  for (int i = 0; i < current\_size; i++)  {  cout << "Фамилия ";  cout << list\_of\_clients[i].fio << endl;  cout << "Телефон ";  cout << list\_of\_clients[i].telephone << endl;  cout << "Почта ";  cout << list\_of\_clients[i].email << endl;  cout << "Товар ";  cout << list\_of\_clients[i].product << endl;  cout << "Количество товаров ";  cout << list\_of\_clients[i].numProduct << endl;  cout << "Дата продажи ";  cout << list\_of\_clients[i].dateSale << endl;  cout << "Дата доставки ";  cout << list\_of\_clients[i].dateDelivery << endl;  }  }  cout << endl << "Введите:" << endl;  cout << "1-для удаления записи" << endl;  cout << "2-для ввода новой записи" << endl;  cout << "3-для изменения записи" << endl;  cout << "4-для вывода записи(ей)" << endl;  cout << "5-для выхода" << endl;  cin >> choice;  } |
| Данные из консоли |
|  |