Лабораторная работа №02. Типы данных, определяемые пользователем. Структуры и объединения. Многофайловые проекты.

1 Цель и порядок работы

Цель работы – ознакомиться с типами данных, определяемыми пользователем и их применением в процессе программирования.

Порядок выполнения работы:

- ознакомиться с описанием лабораторной работы;
- получить задание у преподавателя, согласно своему варианту;
- написать программу и отладить ее на ЭВМ.

2 Краткая теория

В реальных задачах, обрабатываемая информация может иметь достаточно сложную структуру. Язык программирования C++ позволяет определять свои типы данных и правила работы с ними. Рассмотрим эти возможности подробней.

2.1 Определение типов (typedef)

Для того чтобы сделать программу более легкой для восприятия, удобно задавать используемым типам новые мнемонические имена, отражающие их суть в разрабатываемой программе. Кроме того, принято употреблять такие имена для сложных составных типов воспринимаемых, в противном случае, с трудом, например при объявлении указателей на функции.

Директива **typedef** позволяет задать синоним для встроенного или пользовательского типа данных. Имена, определенные с помощью директивы **typedef**, можно использовать точно так же, как обычные спецификаторы типов.

Синтаксис директивы **typedef** имеет два варианта:

```
typedef тип новое_имя;

typedef тип новое_имя [ размерность ];
```

Второй вариант используется при определении нового имени типа для массива, а квадратные скобки являются элементом синтаксиса.

Например:

```
typedef unsigned int UINT; //UINT - беззнаковое целое
typedef char msg[100];     //msg - массив из ста символов типа char
```

Объявленные таким образом имена типов используются так же, как и имена стандартных типов:

```
UINT i, j;     //две переменных типа unsigned int
UINT A[10];    //массив переменных типа unsigned int
msg m;         //массив (строка) из ста символов типа char
msg strs[10]; //массив из 10 строк по сто символов каждая
```

Кроме задания типам более коротких псевдонимов **typedef** используется для облегчения переносимости программ. При изменении машинно-зависимых типов, при

переносе необходимо будет изменить только оператор определения типа без изменения всей остальной программы.

2.2 Перечисления (enum)

При написании программ часто возникает потребность определить несколько именованных констант, для которых требуется, чтобы все они имели различные значения (при этом конкретные значения могут быть не важны). Для этого удобно воспользоваться перечисляемым типом данных, все возможные значения которого задаются списком целочисленных констант.

Формат объявления перечисляемого типа данных:

```
enum  [ имя типа ] { список_констант };
```

Здесь квадратные скобки указывают на необязательность использования.

Имя типа задается в том случае, если в программе требуется определять переменные этого типа. Компилятор обеспечивает, чтобы эти переменные принимали значения только из списка констант. Константы должны быть целочисленными и могут инициализироваться обычным образом. При отсутствии инициализатора первая константа будет рана нулю, а каждой следующей присваивается на единицу большее значение, чем предыдущей.

Например:

```
enum Err {ERR_READ, ERR_WRITE, ERR_CONVERT};

Err error;

switch (error)
{
    case ERR_READ:    /* операторы */ break;
    case ERR_WRITE:   /* операторы */ break;
    case ERR_CONVERT: /* операторы */ break;
}



enum {two = 2, three, four, ten = 10, eleven, fifty = ten + 40};
```

Константам `ERR_READ, ERR_WRITE, ERR_CONVERT` присваивается значения 0, 1 и 2 соответственно.

Константам three и four присваиваются значения 3 и 4, константе eleven – 11. Имена перечисляемых констант должны быть уникальными, а значения могут совпадать. Преимущество применения перечисления перед описанием именованных констант и директивой **typedef** состоит в том, что связанные константы нагляднее; кроме того, компилятор при инициализации констант может выполнять проверку типов.

При выполнении арифметических операций перечисления преобразуются в целые. Поскольку перечисления являются типами, определяемыми пользователем, для них можно вводить собственные операции.

Диапазон значений перечисления определяется количеством бит, необходимым для представления всех его значений. Любое значение целочисленного типа можно явно привести к типу перечисления, но при выходе за пределы его диапазона результат не определен.

2.3 Структуры (struct)

Структура – это объединенное в единое целое множество поименованных элементов в общем случае разных типов. В отличие от массива, все элементы которого однотипны, структура может содержать элементы разных типов.

Каждая структура включает в себя один или несколько объектов (переменных, массивов, указателей, структур и т. д.), называемых элементами структуры (компонентами).

Элементы структуры также называются *полями структуры* и могут иметь любой тип, кроме типа этой же структуры, но могут быть указателями на него.

Структуры, так же, как и массивы относятся к структурированным типам данных. Они отличаются от массивов тем, что, во-первых, к элементам структуры необходимо обращаться по имени, во-вторых, все поля структуры необязательно должны принадлежать одному типу.

```
struct [ имя_типа ]
{
   тип_1 элемент_1;
   тип_2 элемент_2;
...
   тип_n элемент_n;
} [ список_описателей ];
```

Такое определение вводит новый производный тип, который называется структурным типом.

Если список описателей отсутствует, описание структуры определяет новый тип, имя которого можно использовать в дальнейшем наряду со стандартными типами

Например:

```
struct Worker{ // описание нового типа Worker
   char fio[30];
   int age, code;
   double salary;
}; // описание заканчивается точкой с запятой

// определение переменной, массива типа Worker и указателя на тип Worker
Worker y, staff[100], *ps;
```

Если отсутствует имя типа, должен быть указан список описателей переменных, указателей или массивов. В этом случае описание структуры служит определением элементов этого списка.

```
// Определение переменной, массива структур и указателя на структуру
struct {
   char fio[30];
   int age, code;
   double salary;
} x, staff[100], *ps;
```

Имя структуры можно использовать сразу после его объявления (определение можно дать позднее) в тех случаях, когда компилятору не требуется знать размер структуры.

```
struct List; // объявление структуры List
struct Link
{
```

```
   List *p;           // указатель на структуру List
   Link *prev, *succ; // указатели на структуру Link
}

struct List { /* определение структуры List */};
```

Это позволяет создавать связные списки структур.

Для инициализации структуры значения ее элементов перечисляют в фигурных скобках в порядке их описания.

```
struct Worker
{
   char fio[30];
   int age, code;
   double salary;
};

Worker ivanov = {"Иванов И.И.", 31, 215, 5800.35};
```

При инициализации массивов структур следует заключать в фигурные скобки каждый элемент массива.

```
struct complex
{
   float re, im;
} compl [3] = { {1.3, 5.2}, {3.0, 1.5}, {1.5, 4.1}};
```

Для переменных одного и того же структурного типа определена операция присваивания, при этом происходит поэлементное копирование. Но присваивание – это и все, что можно делать со структурами целиком. Другие операции, например сравнение на равенство или вывод, не определены. Структуру можно передавать в функцию и возвращать в качестве значения функции.

Размер структуры не обязательно равен сумме размеров ее элементов, поскольку они могут быть выровнены по границам слова.

Доступ к полям структуры выполняется с помощью операций выбора . (точка) при обращении к полю через имя структуры и -> при обращении через указатель.

Ввод/вывод структур, как и массивов, выполняется поэлементно.

```
Worker worker, staff[100], *ps;

worker.fio = "Петров С.С.";
staff[3] = worker;
staff[8].code = 123;
ps->salary = 4350.00;
```

Переменные структурного типа можно размещать и в динамической области памяти, для этого надо описать указатель на структуру и выделить под нее место

```
Worker *ps = new Worker;     //создает переменную структурного типа
Worker *pms = new Worker[5]; //создает массив структурного типа
```

```
//обращение через операцию косвенного доступа
ps->age = 55;
//обращение через разыменовывание указателя
(*ps).code = 253;

//обращение к 0 элементу созданного массива через индекс
pms[0].salary = 5800;

//обращение к 1 элементу созданного массива через указатель
(*(pms + 1)).salary = 4800;

//очистка занимаемой памяти
delete ps;
delete []pms;
```

Если элементом структуры является другая структура, то доступ к ее элементам выполняется через две операции выбора.

```
struct A
{
   int a;
   double x;
};

struct B
{
   A a;
   double x;
};

B x[2];

x[0].a.a = 1;
x[0].a.x = 35.15;
x[1].x = 0.1;
```

Как видно из примера, поля разных структур могут иметь одинаковые имена, поскольку у них разная область видимости.

2.4 Объединения (union)

Объединение (union) представляет собой частный случай структуры, все поля которой располагаются по одному и тому же адресу. Формат описания такой же, как у структуры, только вместо ключевого слова struct используется слово union.

```
union [ имя_типа ]
{
   тип_1 элемент_1;
   тип_2 элемент_2;
...
   тип_n элемент_n;
} [ список_описателей ];
```

Длина объединения равна наибольшей из длин его полей. Когда используется элемент меньшей длины, то переменная типа объединения может содержать неиспользуемую память.

Все элементы объединения хранятся в одной и той же области памяти, начиная с одного адреса.

В каждый момент времени в переменной типа объединение хранится только одно значение, и ответственность за его правильное использование лежит на программисте.

Объединение применяется для следующих целей:

– экономия памяти в тех случаях, когда известно, что больше одного поля одновременно не требуется;

– интерпретация одного и того же содержимого области памяти объединения с точки зрения различных типов данных.

Доступ к элементам объединения осуществляется тем же способом, что и к структурам.

Заносить значения в участок памяти, выделенный для объединения, можно с помощью любого из его элементов. То же самое справедливо и относительно доступа к содержимому участка памяти, выделенного для объединения.

При определении конкретных объединений разрешена их инициализация, причем инициализируется только первый элемент объединения.

Например, пусть в магазине имеется возможность использования различных способов оплаты (кредитная карта, чек, наличные). Тогда в программе учета одновременно нам необходимо будет хранить только одно из значений, тогда как остальные нас не интересуют.

```
#include <iostream>

using namespace std;

int main()
{
   enum paytype {CHECK, CARD, CASH};

   paytype ptype;

   union payment{
       long check;
       char card[25];
       float sum;
   }

   //инициализация возможна только через первый элемент объединения
   payment info = 24557695;
   /* присваивание значений info и ptype */

   ptype = CASH;
   cin >> info.summ;

   switch (ptype)
   {
       case CHECK: cout << "Оплата чеком: " << info.check; break;
       case CARD : cout << "Оплата по карте: " << info.card; break;
       case CASH : cout << "Оплата наличными: " << info.sum; break;
   };

   return 0;
}
```

При определении объединений без явного указания имени объединяющего типа разрешено не вводить даже имени объединения. В этом случае создается анонимное или безымянное объединение.

К элементам анонимного объединения можно обращаться как и к отдельным объектам, но при этом могут изменяться другие элементы объединения.

```
union {
    int INT[2];
    char CH[8];
} = {1, 2}

INT[0] = 25;    // изменятся значения CH[0], CH[1], CH[2], CH[3]
CH[4] = 'a';    // изменится значение INT[1]
```

Объединение часто используют в качестве поля структуры, при этом в структуру удобно включить дополнительное поле, определяющее, какой именно элемент объединения используется в каждый момент. Имя объединения можно не указывать, что позволяет обращаться к его полям непосредственно.

```
#include <iostream>

using namespace std;

int main()
{
    enum paytype {CARD, CHECK, CASH};

    struct payment
    {
        paytype ptype;
        union{
            char card[25];
            long check;
        };
    };

    payment info;
    /* присваивание значений info и ptype */

    ptype = CASH;
    cin >> info.summ;

    switch (info.ptype)
    {
        case CARD : cout << "Оплата по карте: " << info.card; break;
        case CHECK: cout << "Оплата чеком: " << info.check; break;
        case CASH : cout << "Оплата наличными: " << info.sum; break;
    };

    return 0;
}
```

2.5 Битовые поля

Битовые поля – это особый вид полей структуры. Они используются для плотной упаковки данных, например, флажков типа «да/нет». Минимальная адресуемая ячейка памяти – 1 байт, а для хранения флажка достаточно одного бита. При описании битового поля после имени через двоеточие указывается длина поля в битах (целая положительная константа).

```
struct Options
{
   bool centerX:1;
   bool centerY:1;
   unsigned int border_type:2;
   unsigned int color_index:4;
}
```

Битовые поля могут быть любого целого типа. Имя поля может отсутствовать, такие поля служат для выравнивания на аппаратную границу. Доступ к полю осуществляется обычным способом – по имени. Адрес поля получить нельзя, однако в остальном битовые поля можно использовать точно так же, как обычные поля структуры. Следует учитывать, что операции с отдельными битами реализуются гораздо менее эффективно, чем с байтами и словами, так как компилятор должен генерировать специальные коды, и экономия памяти под переменные оборачивается увеличением объема кода программы. Размещение битовых полей в памяти зависит от компилятора и аппаратуры.

2.6 Примеры использования

Пример 1. Нахождение расстояния между точками в декартовой системе координат

```
#include "stdafx.h"
#include <iostream>
#include <math.h>

using namespace std;

//объявляем структуру точка
struct point
{
   float X;
   float Y;
};

float dst(point p, point q)
{
   return sqrt((p.X - q.X)*(p.X - q.X) + (p.Y - q.Y)*(p.Y - q.Y));
}

int main()
{
   setlocale(LC_ALL, "Russian");

   point a, b;

   cin >> a.X >> a.Y;

   cin >> b.X >> b.Y;
```

```
    float d = dst(a, b);

    cout << "Расстояние между точками: " << d << endl;

    return 0;
}
```

Пример 2. Работа с каталогом книг

```
#include "stdafx.h"
#include <iostream>
#include <math.h>

using namespace std;

//объявляем структуру книга
struct book
{
    char author[40];
    char title[100];
    int year;
    int pages;
    int quantity;
    float price;
};

int main()
{
    setlocale(LC_ALL, "Russian");

    int N;

    cout << "Введите количество книг в каталоге: " << endl;

    cin >> N;

    //созадим динамически каталог книг
    book *catalog = new book[N];

    //введем данные о книгах с клавиатуры
    for (int i = 0; i < N; i++)
    {
        cout << "Введите имя автора: " << endl;
        cin >> catalog[i].author;
        cout << "Введите название: " << endl;
        cin >> catalog[i].title;
        cout << "Введите год издания: " << endl;
        cin >> catalog[i].year;
        cout << "Введите количество страниц: " << endl;
        cin >> catalog[i].pages;
        cout << "Введите количество книг: " << endl;
        cin >> catalog[i].quantity;
        cout << "Введите цену книги: " << endl;
        cin >> catalog[i].price;
        cout << endl;
    }
```

```
    //выведем на экран книги, цена которых меньше 200 руб
    //и найдем сумму всех книг в каталоге
    float s = 0;
    for (int i = 0; i < N; i++)
    {
        if (catalog[i].price < 200)
        {
            cout << "Автор: " << catalog[i].author << endl;
            cout << "Название: " << catalog[i].title << endl;
            cout << "Год издания: " << catalog[i].year << endl;
            cout << "Количество страниц: " << catalog[i].pages << endl;
            cout << "Количество книг: " << catalog[i].quantity << endl;
            cout << "Цена книги: " << catalog[i].price << endl;
            cout << "Сумма: " << catalog[i].price*catalog[i].quantity <<
endl;
            cout << endl;
        }
        s += catalog[i].price*catalog[i].quantity;
    }

    cout << "Общая сумма: " << s << endl;

    delete []catalog;

    return 0;
}
```

2.7 Особенности работы с многофайловыми проектами

До сих пор мы писали программы, исходный текст которых размещался в одном файле. Однако реальные задачи требуют создания многофайловых проектов с распределением решаемых подзадач по разным модулям (файлам).

Среда разработки Visual Studio 2008, как и подавляющее большинство других, поддерживают создание, компиляцию и сборку многофайловых проектов.

Под модулем в программировании понимается сгруппированные по своему функциональному назначению совокупности констант, типов данных, функций обрабатывающих эти данные, независимые от основной программы, компилируемые отдельно и допускающие повторное использование.

Модуль содержит данные и функции для их обработки. Для того чтобы его использовать достаточно знать его интерфейс, а не все детали реализации. Сокрытие деталей реализации называется инкапсуляцией.

Пользуясь технологией нисходящего проектирования программ, мы разбиваем исходную задачу на подзадачи. Затем при необходимости каждая из них также разбивается на подзадачи, и так далее, пока решение очередной подзадачи не окажется достаточно простым, то есть реализуемым в виде функции обозримого размера (наиболее предпочтительным считается размер не более одного-двух экранов текстового редактора).

Исходные тексты совокупности функций для решения какой-либо подзадачи, как правило, размещаются в отдельном модуле (файле). Такой файл называют исходным (sources). Обычно он имеет расширение .c или .cpp Прототипы всех функций исходного файла выносят в отдельный так называемый заголовочный файл (header file), для него принято использовать расширение .h или .hpp.

Таким образом, заголовочный файл xxx.h содержит интерфейс для некоторого набора функций, а исходный файл xxx.cpp содержит реализацию этого набора. Если некоторая функция из указанного набора вызывается из какого-то другого исходного модуля yyy.cpp, то обязаны включить в этот модуль заголовочный файл xxx.h с помощью директивы #include.

Негласное правило стиля программирования на C++ требует включения этого же заголовочного файла (с помощью #include) и в исходный файл xxx.cpp.

Хотя использование глобальных переменных считается дурным тоном в программировании, иногда возникают ситуации когда невозможно избежать их использования. В многофайловом проекте возможны два «вида глобальности переменных». Если некоторая глобальная переменная glvar1 объявлена в файле xxx.cpp с модификатором static, то она видима от точки определения до конца этого файла, то есть области видимости ограничена файлом. Если же другая глобальная переменная glvar2 объявлена в файле xxx.cpp без модификатора static, то она может быть видимой в пределах всего проекта. Правда, для того, чтобы она оказалась видимой в другом файле, необходимо в этом файле ее объявление с модификатором extern (рекомендуется это объявление поместить в файл xxx.h).

В заголовочном файле принято размещать:

- определения типов, задаваемых пользователем, констант, шаблонов;
- объявления (прототипы) функций;
- объявления внешних глобальных переменных (с модификатором extern);
- пространства имен.

При подобно использовании заголовочных файлов возникает проблема их повторного включения. Проблема может возникнуть при иерархическом проектировании структур данных, когда в некоторый заголовочный файл yyy.h включается при помощи директивы include другой заголовочный файл xxx.h (например, для использования типов, определенных в этом файле) и где-нибудь еще, например в основной программе.

Для решения этой проблемы рекомендуется использовать так называемые стражи включения. Данный способ состоит в следующем: чтобы предотвратить заголовочных файлов, содержимое каждого .h файла должно находиться между директивами условной компиляции #ifndef #endif, а внутри устанавливаться признак включения при помощи директивы #define.

```
#ifndef FILENAME_H

#define FILENAME_H

/* содержимое заголовочного файла */

#endif
```

2.8 Добавление новых файлов в проект в среде разработки Visual Studio

Для добавления новых файлов в проект новых файлов в проект необходимо вызвать контекстное меню над именем проекта в «Проводнике решений» (Solution Explorer) (см. рисунок 11.1) и далее выбрать раскрывающийся пункт «Добавить» (Add). В нем нас на текущий момент будут интересовать два подпункта “New Item …” (Новый объект) и “Exiting Item …” (Существующий объект). Первый создает новый файл, одного из предложенных типов, а второй добавляет к проекту уже существующий. Диалоговое окно с вариантами доступных для создания файлов представлено на рисунке 11.2.

Результат добавления файлов в проект представлен на рисунке 11.3. Теперь достаточно воспользоваться директивой #include для использования функций, констант и типов данных, располагающихся в новых файлах.

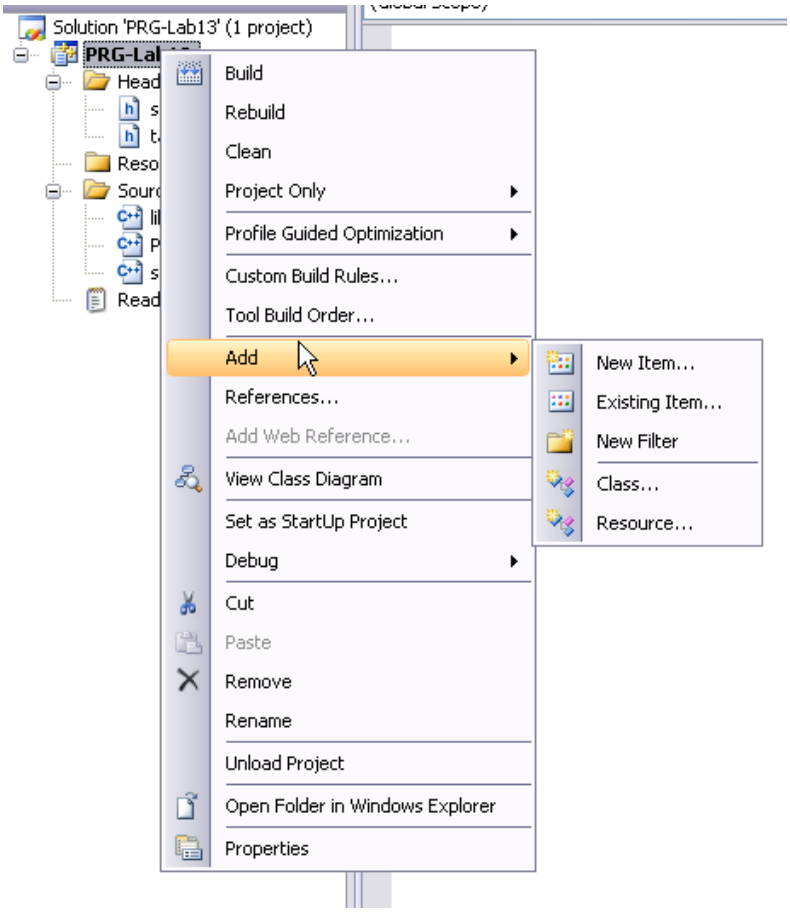

Рисунок 11.1 – Добавление нового файла в проект

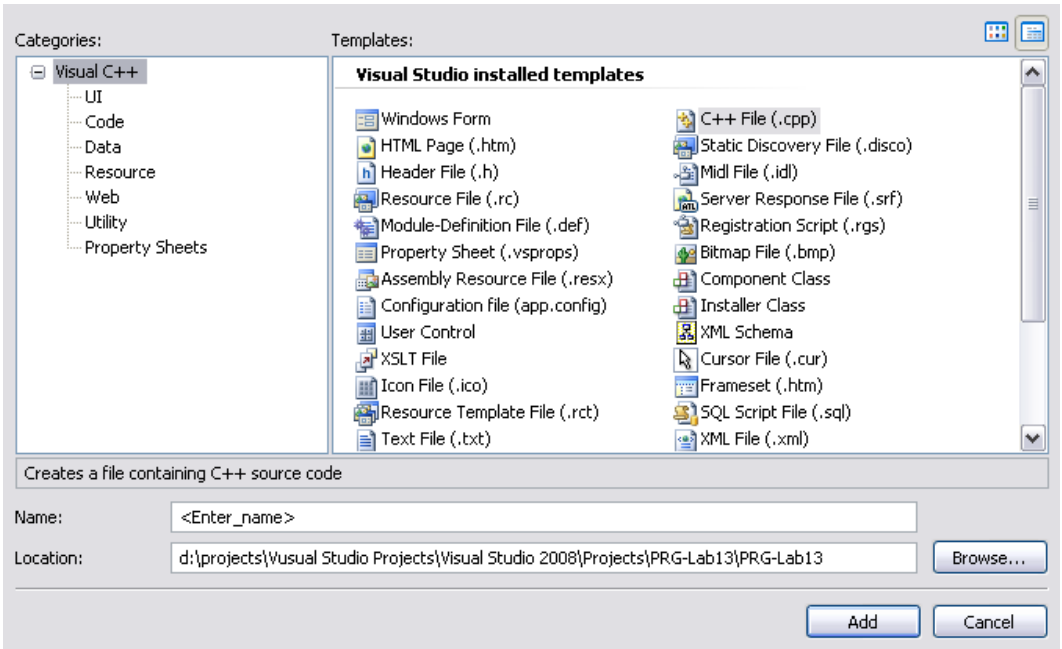

Рисунок 11.2 – Диалоговое окно выбора типа добавляемого файла

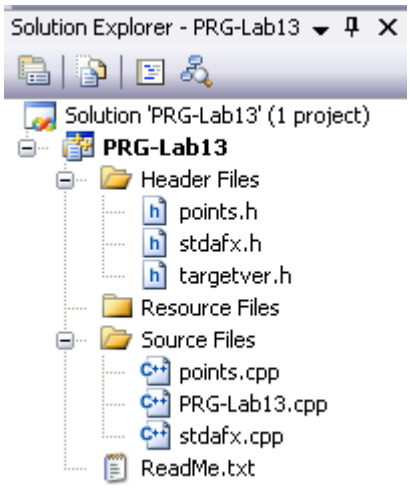

Рисунок 11.3 – Результат добавления новых файлов в проект

3 Контрольные вопросы

1. Как объявить пользовательский тип данных?
2. Что такое перечисления?
3. Как осуществляется описание структуры?
4. Что такое поле структуры?
5. Как обратится к элементу структуры?
6. К каким типам данных относятся объединения и структуры?
7. Как проинициализировать переменную структурного типа?
8. Для чего применяются структуры?
9. Как объявляется объединение?
10. Как используются объединения?
11. Как обратится к элементу объединения?
12. Что такое битовые поля?
13. Как обратится к элементу битового поля?
14. В чем отличие многофайлового проекта?
15. Как добавить новый файл в проект?
16. Что такое модуль?
17. Для чего нужно разбиение на модули?
18. Для чего применяются заголовочные файлы?
19. Что размещается в заголовочных файлах, а что выносится в тело модуля?
20. Что такое стек? Принципы работы со стеком.
21. Что такое очередь? Принципы работы с очередью.

4 Задание

1. Написать программу в соответствии с вариантом задания из пункта 5.1.
2. Отладить и протестировать программу.
3. Написать программу в соответствии с вариантом задания из пункта 5.2.
4. Отладить и протестировать программу.
5. Написать программу в соответствии с вариантом задания из пункта 5.3. Стандартные функции работы со стеком и очередью **не использовать.** Все функции, используемые в проекте должны содержаться в отдельном модуле.
6. Отладить и протестировать программу.

5 Варианты заданий

5.1 Структуры (Часть 1)

Вариант определяется по последней цифре в списке студентов группы.

Вариант 1.

Описать структуру с именем STUDENT, содержащую поля:

- фамилия и инициалы;
- номер группы;
- успеваемость (массив из пяти элементов).

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из 5 структур типа STUDENT;
- вывод на дисплей фамилий и номеров групп для всех студентов, включенных в массив, если средний балл студента больше 4.0;
- если таких студентов нет, вывести соответствующее сообщение.

Вариант 2.

Описать структуру с именем STUDENT, содержащую поля:

- фамилия и инициалы;
- номер группы;
- успеваемость (массив из пяти элементов).

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из 5 структур типа STUDENT;
- вывод на дисплей фамилий и номеров групп для всех студентов, включенных в массив, имеющих хотя бы одну оценку 2;
- если таких студентов нет, вывести соответствующее сообщение.

Вариант 3.

Описать структуру с именем AEROFLOT, содержащую поля:

- название пункта назначения рейса;
- номер рейса;
- тип самолета.

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из 5 элементов типа AEROFLOT;
- вывод на экран информации о рейсе, номер которого введен с клавиатуры;
- если таких рейсов нет, вывести соответствующее сообщение.

Вариант 4.

Описать структуру с именем AEROFLOT, содержащую поля:

- название пункта назначения рейса;
- номер рейса;
- тип самолета.

Написать программу, выполняющую следующие действия:

- ввод с клавиатуры данных в массив, состоящий из 5 элементов типа AEROFLOT;
- вывод на экран пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры;
- если таких рейсов нет, вывести соответствующее сообщение.

Вариант 5.

Описать структуру с именем WORKER, содержащую поля:

- фамилия и инициалы работника;

– название занимаемой должности;
– зарплату;
– год поступления на работу.

Написать программу, выполняющую следующие действия:
– ввод с клавиатуры данных в массив, состоящий из 5 структур типа WORKER;
– вывод на дисплей фамилий работников, чей стаж работы в организации превышает значение, введенное с клавиатуры;
– если таких работников нет, вывести соответствующее сообщение.

Вариант 6.

Описать структуру с именем TRAIN, содержащую поля:
– название пункта назначения;
– номер поезда;
– время отправления.

Написать программу, выполняющую следующие действия:
– ввод с клавиатуры данных в массив, состоящий из 5 структур типа TRAIN;
– вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени;
– если таких поездов нет, вывести соответствующее сообщение.

Вариант 7.

Описать структуру с именем TRAIN, содержащую поля:
– название пункта назначения;
– номер поезда;
– время отправления.

Написать программу, выполняющую следующие действия:
– ввод с клавиатуры данных в массив, состоящий из 5 структур типа TRAIN;
– вывод на экран информации о пункте назначения, в который отправляется поезд, номер которого введен с клавиатуры;
– если таких поездов нет, вывести соответствующее сообщение.

Вариант 8.

Описать структуру с именем MARSH, содержащую поля:
– название начального пункта маршрута;
– название конечного пункта маршрута;
– номер маршрута.

Написать программу, выполняющую следующие действия:
– ввод с клавиатуры данных в массив, состоящий из 5 структур типа MARSH;
– вывод на экран информации о маршруте, номер которого введен с клавиатуры;
– если таких маршрутов нет, вывести соответствующее сообщение.

Вариант 9.

Описать структуру с именем NOTE, содержащую поля:
– фамилия и имя;
– номер телефона;
– дата рождения (массив из трех чисел).

Написать программу, выполняющую следующие действия:
– ввод с клавиатуры данных в массив, состоящий из 5 структур типа NOTE;
– вывод на экран информации о человеке, номер телефона которого введен с клавиатуры;

– если таких людей нет, вывести соответствующее сообщение.

Вариант 0.
Описать структуру с именем ORDER, содержащую поля:
– расчетный счет плательщика;
– расчетный счет получателя;
– перечисляемая сумма в руб.
Написать программу, выполняющую следующие действия:
– ввод с клавиатуры данных в массив, состоящий из 5 структур типа ORDER;
– вывод на экран информации о сумме, снятой с расчетного счета плательщика, введенного с клавиатуры;
– если таких счетов нет, вывести соответствующее сообщение.

5.2 Структуры (Часть 2)

Варианты заданий выбирать согласно номера в списке группы.

1. Описать структуру «дата» (год, месяц, день). Определить функцию «дней до конца месяца» вычисляющую количество дней до конца месяца.
2. Описать структуры для декартовых (x, y) и полярных (r, pi) координат. Определить функцию для перевода из декартовых в полярные координаты.
3. Описать структуру «время» (часы, минуты, секунды). Определить функцию «предыдущая минута» уменьшающая передаваемое в нее время на 1 минуту (учесть, что в сутках 24 часа).
4. Описать структуру «комплексное число» (действительная часть (re), мнимая часть (im)). Определить функции, выполняющие сложение, вычитание и умножение на действительную константу.
5. Описать структуру «дата» (год, месяц, день). Определить функцию «недель с начала года» вычисляющую количество недель с начала года.
6. Описать структуру «время» (часы, минуты, секунды). Определить функцию «прошедшее время» определяющую интервал времени между t1 и t2 в минутах (округление производить в меньшую сторону).
7. Описать структуру «комплексное число» (действительная часть (re), мнимая часть (im)). Определить функцию, выполняющую произведение двух комплексных чисел.
8. Описать структуры для декартовых (x, y) и полярных (r, pi) координат. Определить функцию для перевода из полярных в декартовы координаты.
9. Описать структуру «дата» (год, месяц, день). Определить функцию «дней с начала года» вычисляющую количество дней с начала года.
10. Описать структуру «время» (часы, минуты, секунды). Определить функцию «следующая минута» увеличивающую передаваемое в нее время на 1 секунду (учесть, что в сутках 24 часа).
11. Описать структуру «комплексное число» (действительная часть (re), мнимая часть (im)). Определить функцию, вычисляющую значение квадратного трехчлена a*x*x+b*x+c в комплексной точке x .
12. Описать структуру «дата» (год, месяц, день). Определить функцию «верна ли дата» проверяющую корректность введенной даты.
13. Описать структуру «время» (часы, минуты, секунды). Определить функцию «раньше» для проверки, предшествует ли время t1 времени t2 (в рамках суток).

14. Описать структуру «комплексное число» (действительная часть (re), мнимая часть (im)). Определить функцию, выполняющую деление двух комплексных чисел.
15. Описать структуру «дата» (год, месяц, день). Определить функцию «дней до конца года» вычисляющую количество дней до конца года.
16. Описать структуру «время» (часы, минуты, секунды). Определить функцию «прошедшее время» определяющую интервал времени между t1 и t2 в секундах.
17. Описать структуру «дата» (год, месяц, день). Определить функцию «дней с начала месяца» вычисляющую количество дней с начала месяца.
18. Описать структуру «комплексное число» (действительная часть (re), мнимая часть (im)). Определить функцию, вычисляющую значение кубического многочлена a*x*x*x +b*x*x +c*x+d в комплексной точке x .
19. Описать структуру «время» (часы, минуты, секунды). Определить функцию «следующая секунда» увеличивающую передаваемое в нее время на 1 секунду (учесть, что в сутках 24 часа).
20. Описать структуру «дата» (год, месяц, день). Определить функцию «день недели» для определения дня недели, на который приходится дата (учитывая, что 1 января 1-го года нашей эры было понедельником).
21. Описать структуру «время» (часы, минуты, секунды). Определить функцию «предыдущая секунда» уменьшающая передаваемое в нее время на 1 секунду (учесть, что в сутках 24 часа).
22. Описать структуру «дата» (год, месяц, день). Определить функцию «дней до конца месяца» вычисляющую количество дней до конца месяца.
23. Описать структуру «время» (часы, минуты, секунды). Определить функцию «позже» для проверки, предшествует ли время t2 времени t1 (в рамках суток).
24. Описать структуру для декартовых координат (x, y). Определить функцию для нахождения расстояния между двумя точками.
25. Описать структуру «дата» (год, месяц, день). Определить функцию «недель до конца года» вычисляющую количество недель до конца года.

Замечание.

Полярная система координат — система координат, ставящая в соответствие каждой точке на плоскости пару чисел $(\rho;\varphi)$.

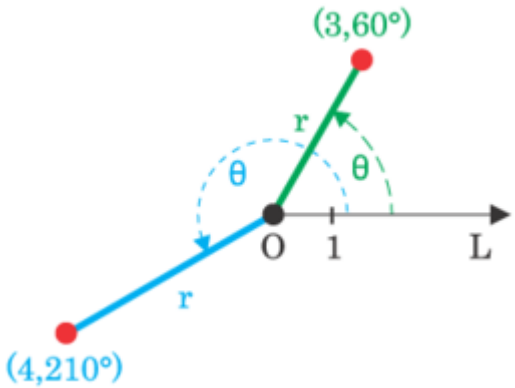

Рис. 10.1 Полярная система координат

Основными понятиями этой системы являются точка отсчёта (полюс) и луч, начинающийся в этой точке (полярная ось).

Координата ρ определяет расстояние от точки до полюса, координата φ — угол между полярной осью и отрезком, соединяющим полюс и рассматриваемую точку.

Координата φ берётся со знаком «+», если угол от оси до отрезка вычисляется против часовой стрелки, и со знаком «-» в противоположном случае. Любая точка в этой системе имеет бесконечное число координат вида $(\rho;\varphi+2\pi\eta)$, которым соответствует одна и та же точка при любых натуральных η. Для полюса $\rho = 0$, а угол φ произвольный.

Формулы перехода:
- от полярной системы координат к декартовой:

$$\begin{cases} x = \rho\cos\varphi \\ y = \rho\sin\varphi \end{cases}$$

- от декартовой системы координат к полярной:

$$\begin{cases} \rho = \sqrt{x^2+y^2} \\ \cos\varphi = \frac{x}{\sqrt{x^2+y^2}};\ \ \sin\varphi = \frac{y}{\sqrt{x^2+y^2}} \\ \operatorname{tg}\varphi = \frac{y}{x}\ (x\neq 0); \varphi = \begin{cases} \frac{\pi}{2}, & y>0 \\ -\frac{\pi}{2}, & y<0 \end{cases} (x=0) \end{cases}$$

5.3 Динамические структуры данных

Варианты заданий выбирать согласно номера в списке группы.

Вариант 1.

1. Создать запись для хранения следующей информации:
   - код груза,
   - номер вагона,
   - стоимость перевозки,
   - дата отгрузки,
   - дата возврата вагона.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать стек.
3. Получить список номеров вагонов, использовавшихся в первом квартале прошлого года.
4. Найти среднюю стоимость перевозки по каждому из встречающихся кодов грузов.
5. Найти сроки использования каждого из вагонов в январе текущего года.

Вариант 2.

1. Создать запись для хранения следующей информации:
   - номер посылки,
   - вес посылки,
   - цена,
   - дата отправки,
   - пункт назначения.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать очередь.

3. Получить список пунктов назначения и номеров посылок, отправленных во втором квартале прошлого года.
4. Найти общую стоимость посылок, отправленных по каждому из встречающихся пунктов назначения.
5. Найти количество отправленных посылок за каждый день в феврале текущего года.

Вариант 3.

1. Создать запись для хранения следующей информации:
   - код товара,
   - название фирмы-производителя,
   - стоимость,
   - дата поступления на склад,
   - дата отгрузки.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать стек.
3. Получить список фирм, поставивших товары в первом квартале прошлого года.
4. Найти среднюю стоимость товаров по каждой из встречающихся фирм.
5. Найти сроки нахождения каждого из товаров на складе в марте текущего года.

Вариант 4.

1. Создать запись для хранения следующей информации:
   - наименование товара,
   - место покупки,
   - цена,
   - дата покупки.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать очередь.
3. Получить список мест покупки и наименований товаров, приобретенных во втором квартале позапрошлого года.
4. Найти среднюю стоимость покупок, сделанных по каждому из встречающихся мест покупки.
5. Найти количество покупок за каждый месяц прошлого года.

Вариант 5.

1. Создать запись для хранения следующей информации:
   - номер заказа,
   - дата заказа,
   - стоимость,
   - код исполнителя,
   - дата выполнения.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать стек.
3. Получить список номеров заказов, выполненных меньше, чем за 15 дней летом прошлого года.

4. Найти среднюю стоимость заказов по каждому из встречающихся исполнителей.
5. Найти суммарные сроки выполнения заказов по каждому из кварталов прошлого и текущего года.

Вариант 6.

1. Создать запись для хранения следующей информации:
   - Ф.И.О.,
   - должность,
   - оклад,
   - дата поступления на работу,
   - стаж к моменту поступления.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать очередь.
3. Получить список лиц и занимаемых ими должностей, принятых на работу в третьем квартале прошлого года.
4. Найти количество человек, имеющих оклад выше среднего, среди принятых на работу без стажа.
5. Найти количество принятых на работу по каждой из имеющихся должностей за последние три года.

Вариант 7.

1. Создать запись для хранения следующей информации:
   - код владельца,
   - номер автомобиля,
   - марка автомобиля,
   - дата выпуска,
   - дата регистрации.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать стек.
3. Получить список номеров и марок автомобилей, зарегистрированных в ноябре и декабре прошлого года.
4. Найти средний возраст по каждой из встречающихся марок автомобилей.
5. Найти "возраст" с точностью до года каждого из автомобилей, зарегистрированных в феврале и марте текущего года.

Вариант 8.

1. Создать запись для хранения следующей информации:
   - Ф.И.О.,
   - вес,
   - рост,
   - дата рождения,
   - пол,
   - место рождения.

2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать очередь.
3. Получить список лиц, возраст которых на данный момент больше среднего.
4. Найти средний рост и средний вес по каждому из встречающихся мест рождения.
5. Найти количество лиц, имеющих вес выше среднего по каждому из месяцев рождения.

Вариант 9.

1. Создать запись для хранения следующей информации:
   - шифр книги,
   - название,
   - автор,
   - дата последней выдачи,
   - год издания.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать стек.
3. Получить список шифров и названий книг, выдававшихся в последний раз в первом полугодии позапрошлого года.
4. Найти средний "возраст" книг по каждому из встречающихся авторов.
5. Найти средние сроки, прошедшие после последней выдачи книг по всем авторам.

Вариант 10.

1. Создать запись для хранения следующей информации:
   - номер билета,
   - номер рейса,
   - цена,
   - дата продажи,
   - фамилия кассира.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать очередь.
3. Получить список кассиров, продавших билеты по ценам выше средней во втором квартале прошлого года.
4. Найти общую стоимость билетов, проданных по каждому из встречающихся номеров рейсов.
5. Найти количество проданных билетов за каждый месяц прошлого года.

Вариант 11.

1. Создать запись для хранения следующей информации:
   - пункт назначения,
   - номер рейса,
   - дата, начиная с которой выполняется данный рейс,
   - дата, до которой выполняется данный рейс,
   - стоимость билета,
   - название авиакомпании.

2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать стек.
3. Получить список рейсов, которые будут выполняться в первом полугодии текущего года.
4. Найти среднюю стоимость билетов по каждой из встречающихся авиакомпаний.
5. Найти количество рейсов, максимальную и минимальную стоимость билета по каждому из встречающихся пунктов назначения.

Вариант 12.

1. Создать запись для хранения следующей информации:
   - наименование оборудования,
   - дата покупки,
   - дата истечения гарантии,
   - стоимость,
   - фирма-производитель.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать очередь.
3. Получить список оборудования, на которое истечет гарантия во втором полугодии следующего года.
4. Найти общую стоимость оборудования, выпущенного каждой из встречающихся фирм-производителей.
5. Найти количество оборудования, приобретенного за каждый месяц прошлого года.

Вариант 13.

1. Создать запись для хранения следующей информации:
   - Ф.И.О. студента,
   - факультет,
   - курс,
   - дата рождения,
   - место рождения.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать стек.
3. Получить список студентов, возраст которых меньше среднего.
4. Найти максимальный и минимальный возраст студентов по каждому из встречающихся факультетов.
5. Найти количество студентов для каждого из встречающихся мест рождения.

Вариант 14.

1. Создать запись для хранения следующей информации:
   - Ф.И.О. пациента,
   - дата рождения,
   - дата посещения врача,
   - диагноз,
   - пол.

2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать очередь.
3. Получить список мужчин, обращавшихся к врачу во втором полугодии прошлого года.
4. Найти средний возраст пациентов по каждому из встречающихся диагнозов.
5. Найти количество обращений за каждый месяц прошлого года.

Вариант 15.

1. Создать запись для хранения следующей информации:
   - название предмета, выставленного на аукцион,
   - код аукциониста,
   - стартовая цена,
   - цена продажи,
   - дата продажи.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать стек.
3. Получить список предметов, проданных в первом квартале прошлого года по цене, превосходящей стартовую в 2 раза.
4. Найти среднюю относительную (к стартовой цене) разницу между стартовой ценой и ценой продажи по каждому из встречающихся кодов аукционистов.
5. Найти количество участий в аукционах каждого из аукционистов в феврале текущего года.

Вариант 16.

1. Создать запись для хранения следующей информации:
   - название валюты,
   - цена покупки,
   - цена продажи,
   - дата,
   - название банка.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать очередь.
3. Получить список названий банков, продававших валюту в четвертом квартале прошлого года.
4. Найти среднюю стоимость покупки за прошлый год по каждой из встречающихся валют.
5. Найти количество банков, продававших валюту за каждый месяц прошлого года.

Вариант 17.

1. Создать запись для хранения следующей информации:
   - название товара, предлагаемого к обмену,
   - требующийся товар,
   - дата поступления предложения,
   - дата заключения сделки,
   - код посредника,
   - гонорар посредника.

2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать стек.
3. Получить список товаров, предложенных к обмену в первом полугодии прошлого года.
4. Найти общий размер гонорара по каждому из встречающихся кодов посредников.
5. Найти средние сроки заключения сделок по каждому из требующихся товаров в январе и феврале текущего года.

Вариант 18.

1. Создать запись для хранения следующей информации:
   - порода собаки,
   - год рождения,
   - кличка,
   - дата регистрации,
   - Ф.И.О. владельца.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать очередь.
3. Получить список владельцев и клички овчарок, зарегистрированных во втором квартале позапрошлого года.
4. Найти средний возраст собак по каждой из встречающихся пород.
5. Найти количество регистраций за каждый день в июле прошлого года.

Вариант 19.

1. Создать запись для хранения следующей информации:
   - код задания,
   - Ф.И.О. исполнителя,
   - контрольный срок выполнения (в днях),
   - дата выдачи задания,
   - дата выполнения.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать стек.
3. Получить список кодов заданий, выполненных с соблюдением контрольных сроков в третьем квартале прошлого года.
4. Найти среднюю продолжительность выполнения заданий по каждому из встречающихся кодов заданий.
5. Найти общую продолжительность выполнения заданий каждым из исполнителей в июне позапрошлого года.

Вариант 20.

1. Создать запись для хранения следующей информации:
   - название продукции,
   - стоимость за единицу,
   - количество,
   - дата выпуска,
   - изготовитель.

2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать очередь.
3. Получить список названий продукции, выпущенной в четвертом квартале прошлого года.
4. Найти общую стоимость продукции по каждому из встречающихся изготовителей.
5. Найти количество продукции, выпущенной в каждом из месяцев прошлого года.

Вариант 21.

1. Создать запись для хранения следующей информации:
   - название прибора,
   - причина первого отказа,
   - причина последнего отказа,
   - дата первого отказа,
   - дата последнего отказа,
   - количество отказов с начала эксплуатации.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать стек.
3. Получить список названий приборов и причин отказа для приборов, отказавших впервые в феврале прошлого года.
4. Найти среднее количество отказов по каждому из встречающихся названий приборов.
5. Найти средние сроки, прошедшие со времени последнего отказа по каждой из причин последнего отказа.

Вариант 22.

1. Создать запись для хранения следующей информации:
   - Ф.И.О.,
   - дата заключения контракта,
   - срок действия контракта,
   - должность,
   - отдел,
   - оклад.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать очередь.
3. Получить список лиц, у которых срок действия контракта истекает во втором полугодии следующего года.
4. Найти средний размер оклада по каждому из встречающихся отделов.
5. Найти количество сотрудников и размер максимального и минимального оклада по каждой из должностей.

Вариант 23.

1. Создать запись для хранения следующей информации:
   - название фирмы,
   - количество акций,
   - стартовая цена акции,
   - цена продажи,

- дата продажи.

2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать стек.
3. Найти общую стоимость проданных акций по каждому из кварталов прошлого года.
4. Найти общее количество акций, проданных по цене, превышающей стартовую цену в 1,5 раза, в феврале текущего года.
5. Определить количество акций, проданных указанной фирмой за каждый день в июне текущего года.

Вариант 24.

1. Создать запись для хранения следующей информации:
   - шифр абитуриента,
   - название специальности,
   - название предмета,
   - оценка,
   - дата сдачи экзамена.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать очередь.
3. Получить список шифров абитуриентов, сдававших экзамены во второй половине июля прошлого года.
4. Найти средние баллы по каждому из встречающихся предметов.
5. Найти количество абитуриентов, сдававших экзамены, за каждый день в июле прошлого года.

Вариант 25.

1. Создать запись для хранения следующей информации:
   - название фирмы,
   - количество сотрудников,
   - дата регистрации,
   - район регистрации,
   - Ф.И.О. руководителя,
   - сфера деятельности.

2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать стек.
3. Получить список руководителей фирм, зарегистрированных в мае прошлого года.
4. Найти количество лиц, занятых в каждой из встречающихся сфер деятельности.
5. Найти количество фирм, зарегистрированных в каждом из районов в первом квартале текущего года.

Вариант 26.

1. Создать запись для хранения следующей информации:
   - наименование оборудования,
   - дата покупки,
   - дата истечения гарантии,

- стоимость,
- фирма-производитель.

2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать очередь.
3. Получить список оборудования, на которое истечет гарантия в первом квартале следующего года.
4. Найти среднюю стоимость оборудования, выпущенного каждой из встречающихся фирм-производителей.
5. Найти количество оборудования, приобретенного за каждый квартал прошлого года.

Вариант 27.

1. Создать запись для хранения следующей информации:
   - Ф.И.О. студента,
   - факультет,
   - курс,
   - дата рождения,
   - место рождения.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать стек.
3. Получить список студентов, возраст которых больше среднего.
4. Найти максимальный и минимальный возраст студентов по каждому из встречающихся факультетов.
5. Найти количество студентов для каждого из встречающихся мест рождения.

Вариант 28.

1. Создать запись для хранения следующей информации:
   - Ф.И.О. пациента,
   - дата рождения,
   - дата посещения врача,
   - диагноз,
   - пол.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать очередь.
3. Получить список женщин, обращавшихся к врачу в третьем полугодии прошлого года.
4. Найти средний возраст пациентов по каждому из встречающихся диагнозов.
5. Найти количество обращений за каждый квартал прошлого года.

Вариант 29.

1. Создать запись для хранения следующей информации:
   - название продукции,
   - стоимость за единицу,
   - количество,
   - дата выпуска,
   - изготовитель.

2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать стек.
3. Получить список названий продукции, выпущенной в третьем квартале прошлого года.
4. Найти общую стоимость и количество продукции по каждому из встречающихся изготовителей.
5. Найти количество продукции, выпущенной в каждом из кварталов прошлого года.

Вариант 30.

1. Создать запись для хранения следующей информации:
   - код товара,
   - название фирмы-производителя,
   - стоимость,
   - дата поступления на склад,
   - дата отгрузки.
2. Предусмотреть возможность добавления, изменения и удаление записи и отображения данных на экран. Для хранения данных использовать очередь.
3. Получить список фирм, поставивших товары в четвертом квартале прошлого года.
4. Найти максимальную стоимость товаров по каждой из встречающихся фирм.
5. Найти сроки нахождения каждого из товаров на складе в апреле текущего года.