Лабораторная работа №1
Тема: Структуры.

Ход выполнения лабораторной работы должен быть отражен в отчете. Отчет должен содержать титульный лист, номера задания, коды программ, результат выполнения программы.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Языки программирования C/C++ поддерживает определяемые пользователем структуры – структурированный тип данных. Он является *собранием одного или более объектов (переменных, массивов, указателей, других структур и т.д.), которые для удобства работы с ними сгруппированы под одним именем*.

Структуры:

- облегчают написание и понимание программ.
- помогают сгруппировать данные, объединяемые каким-либо общим понятием.
- позволяют группу связанных между собой переменных использовать как множество отдельных элементов, а также как единое целое.

Как и массив, структура представляет собой *совокупность данных*, но *отличается* от него тем, что *к ее элементам (компонентам) необходимо обращаться по имени* и ее элементы могут быть *различного типа*. Структуры целесообразно использовать там, где необходимо объединить данные, относящиеся к одному объекту.

Определение структуры состоит из двух шагов:

- объявление шаблона структуры (задание нового типа данных, определенного пользователем);
- определение переменных типа объявленного шаблона.

Объявление шаблонов структур. Общий синтаксис объявления ***шаблона структуры***:

```
struct имя_шаблона
{
    тип1 имя_переменной1;
    тип1 имя_переменной1;
    //другие члены данных;
};
```

```
struct Person
{
    char surname[20];
    char name[15];
    long phone_number;
    char *address;
    double weight;
};
```

Имена *шаблонов* должны быть *уникальными* в пределах их *области определения* для того, чтобы компилятор мог различать различные типы шаблонов. Задание шаблона осуществляется с помощью ключевого слова **struct**, за которым следует имя шаблона структуры и список элементов, заключенных в фигурные скобки.

Имена элементов *в одном шаблоне* также должны быть *уникальными*. Однако в разных шаблонах можно использовать одинаковые имена элементов.

Задание только шаблона *не влечет* резервирования памяти компилятором.

Шаблон представляет компилятору необходимую информацию об элементах структурной переменной для *резервирования места* в оперативной памяти и организации доступа к ней *при определении* структурной переменой и использовании отдельных элементов структурной переменной.

Среди членов данных структуры могут также присутствовать, кроме стандартных типов данных (**int**, **float**, **char** и т.д.), ранее определенные типы, например:

```
/* объявление шаблона структуры типа Date */
    struct Date
    { int day, month, year; };
/* шаблон структуры Person */
    struct Person
    {
        char surname[20];
        char name[15];
        long phone_number;
        char *address;
        double weight;
        struct Date birthday;
    };
```

Структура *Date* имеет три поля типа **int**. Шаблон структуры *Person* в качестве элемента включает поле *birthday*, которое, в свою очередь, имеет ранее объявленный тип данных: **struct** *Date*. Этот элемент (*birthday*) содержит в себе все компоненты шаблона **struct** *Date*.

Определение структур-переменных. Определение структуры-переменной ничем не отличается от объявления обычной переменной с предопределенным типом. Общий синтаксис:

```
struct имя_шаблона имя_переменной;
```

Пример*:*

```
struct Person stud[10];
```

Компилятор выделит под каждую переменную количество байтов памяти, необходимое для хранения всех ее элементов.

Разрешается совмещать описание шаблона и определение структурной переменной.

Пример:

```
struct Book
{
        char title[20];
        char author[30];
        double cast;
} book1, book2, *ptr_bk=&book1;
```

Пример:

```
struct Date {int day, month, year;} date1[5]; // объявление массива из 15 структур
```

Доступ к компонентам структуры. Доступ к полям осуществляется с помощью оператора «**.**» при непосредственной работе со структурой или «**->**» - при использовании указателей на структуру. Эти операторы называются *селекторами* членов класса. Общий синтаксис для доступа к компонентам структуры следующий:

```
имя_переменной_структуры.член_данных;
имя_указателя->имя_поля;
(*имя_указателя).имя_поля;
```

Пример*:*

Прямой доступ к элементам

```
1) date1[5].day=10;
2) date1[5].year=1991;
3) strcpy(book1.title, “Война и мир”);
```

/* используя прямое обращение к элементу, присваиваем значение выбранной переменной. Текст помещается в переменную, используя функцию копирования – strcpy(); */

```
4) stud[3].birthday.month=1;
5) stud[3].birthday.year=1980;
```

Доступ по указателю

```
1) (date1+5)->day=10;
2) (stud+3)->birthday.month=1;
```

// Используя доступ по указателю на структуру, присваиваем значение соответствующей переменной. Указатель можно использовать и так:

```
3) (*(date1+5)).day=10;
4) (*(stud+3)).birthday.month=1;
```

Инициализация структур. При определении структурных переменных можно инициализировать их поля. Эта возможность подобна инициализации массива и следует тем же правилам:

```
имя_шаблона имя_переменной_структуры = {значение1, значение2, …};
```

Компилятор присваивает *значение1* первой переменной в структуре, *значение2* – второй переменной структуры и т.д., и тут необходимо следовать некоторым правилам:

- присваиваемые значения должны совпадать по типу с соответствующими полями структуры;
- можно объявлять меньшее количество присваиваемых значений, чем количество полей. Компилятор присвоит нули остальными полями структуры;
- список инициализации последовательно присваивает значения полям структуры, вложенных структур и массивов.

Пример:

```
struct Date
{ int day,month,year; }d[5] = { {1,3,1980}, {5,1,1990}, {1,1,1983} };
```

Копирование структур-переменных. Язык С(С++) позволяет оператору *присваивания* копировать значения одной структуры-переменной в другую переменную, при условии, что обе структуры-переменные относятся к одному и тому же типу. Таким образом, единственный оператор может скопировать несколько членов данных, которые включают массивы и вложенные структуры.

Однако следует учитывать, что оператор присваивания выполняет то, что называется *поверхностной* копией в применении к структурам-переменным. Поверхностная копия представляет собой копирование бит за битом значений полей переменной-источника в соответствующие поля переменной-цели. При этом может возникнуть проблема с такими членами данных, как *указатели*, поэтому использовать поверхностное копирование структур надо осторожно.

Пример 1: Создать массив структур о студентах группы. О каждом студенте записать: имя, фамилию, год рождения, оценки по пяти экзаменам. Определить средний балл за сессию и отсортировать список по сумме баллов.

```
#include "stdafx.h"
#include <stdio.h>
#include <conio.h>

struct student
{ char name[20];
  char fam[30];
  int year;
  int mark[5];
  int average;
};

struct student students[30];
struct student buffer;
int records;
int i, j;
int main()
{
      records=0;
      do
      {
            printf("Student №%d\n", records+1);
            puts("Vvedite familiu: ");
            fflush(stdin);
            gets(students[records].fam);
            puts("Vvedite imya: ");
            fflush(stdin);
            gets(students[records].name);
            puts("Vvedite vozrast: ");
            scanf("%d", &students[records].year);
            for(i=0; i<5; i++)
            {     printf("Vvedite ocenku po ekzamenu №%d ", i+1);
                  scanf("%d", &students[records].mark[i]);
            }
            records++;
```

```
            puts("Prekratit' rabotu? (1/0)");
            scanf("%d", &i);
      } while(i);

      for (i=0; i<records; i++)
      { students[i].average=0;
        for (j=0; j<5; j++)
              students[i].average+=students[i].mark[j];
      }

      for (i=0; i<records-1; i++)
            for (j=i; j<records; j++)
                  if (students[i].average>students[j].average)
                  {  buffer=students[i];
                     students[i]=students[j];
                     students[j]=buffer;
                  }

      for (i=0; i<records; i++)
      {
            printf("Student %s %s ", students[i].name, students[i].fam);
            printf(" Vosrast %d ", students[i].year);
            printf(" Sr. ball %d \n", students[i].average);
      }
      getch();
      return 0;
    }
```

Пример 2: Ввести массив структур. Рассортировать массив в алфавитном порядке фамилий, входящих в структуру, перемещая сами структуры.

```
#include "stdafx.h"
#include<stdio.h>
#include<conio.h>
#include<string.h>

struct st
{ char name[80];
  int age;
};

int _tmain(int argc, _TCHAR* argv[])
{
      int i,j,k;
      struct st m[100], t;
      printf("Vvedite kol-vo studentov: ");
      scanf("%d", &k);
      for (i=0; i<k; i++)
      {     puts("Vvedite imya studenta: ");
            fflush(stdin);
            gets(m[i].name);
            puts("Vvedite vozrast: ");
            scanf("%d", &m[i].age);
      }
      for (i=0; i<k-1; i++)
            for (j=i+1; j<k; j++)
                  if (strcmp(m[i].name, m[j].name)>0)
                  {  t=m[i]; m[i]=m[j]; m[j]=t; }
      printf("\n\nRezultat: ");
      for (i=0; i<k; i++)
      {
            printf("\n\n");
            puts(m[i].name);
            printf("%d years ", m[i].age);
```

```
    }
    getch();
    return 0;
}
```

Пример 3: Определить структурированный тип, набор функций (в виде меню) для работы с массивом структур. Структура «Склад»: наименование, единица измерения, цена единицы, количество, дата последнего прихода/расхода, тип накладной (приход или отгрузка).

В перечень обязательных функций входят:

- ввод элементов (полей) структуры с клавиатуры;
- вывод элементов (полей) структуры;
- поиск в массиве структуры с заданным значением поля;
- удаление заданного элемента;
- изменение (редактирование) заданного элемента.

Интерфейс пользователя осуществить в виде командного процессора:

1 - загрузить данные

2 - вывести на экран ……

```
#include <iostream>
#include <stdio.h>
#include <string.h>
#include <windows.h>
using namespace std;

struct sklad  //объявляем шаблон структуры
{char name[30];     //Наименование
 char ed[5];          //Единица измерения
 float cena;          //Цена
 int kol;             //Количество
 int date;            //Дата последнего завоза
 int type;            //тип накладной (приход или отгрузка)
};

struct sklad mas[30]; //объявляем глобальный массив структур
struct sklad tmp; //объявляем временную переменную структурного типа
int sch=0;        //Счетчик полных записей
int er;            //Переключатель

void enter_new();
int menu();
void out();
void del();
void change();
void find();

int main()
{
      setlocale(LC_ALL,"Russian");
      while(1)
      {     switch(menu())
            {  case 1:del();break;
               case 2:enter_new();break;
               case 3:change();break;
               case 4:out();break;
               case 5:find();break;
               case 6: return 0;
                   default: cout<<"Не верный выбор/n";
```

```
            }
      }
}

void enter_new()                              // ф-ция ввода новой структуры
{
      if(sch<30) //вводим новую запись только, если счетчик полных записей меньше
максимального количества записей
      {
            cout<<"Запись номер"<<sch+1; //выводим номер записи
            cout<<"\nВыберите тип накладной (1 - приход, 2 - отгрузка)\n";
            cin>>mas[sch].type;
            cout<<"\nВведите наименование\n";
            cin>>mas[sch].name;
            cout<<"Введите ед.измерения \n";
            cin>>mas[sch].ed;
            cout<<"Введите цену\n";
            cin>>mas[sch].cena;
            cout<<"Введите кол-во\n";
            cin>>mas[sch].kol;
            cout<<"Введите дату последнего поступления\n";
            cin>>mas[sch].date;
            sch++; //увеличиваем счетчик полных записей на единицу
      }
      else cout<<"Введено максимальное кол-во записей";
}


int menu()
{
      int er;
      cout<<"Введите:\n";
      cout<<"1-для удаления записи\n";
      cout<<"2-для ввода новой записи\n";
      cout<<"3-для изменения записи\n";
      cout<<"4-для вывода записи(ей)\n";
      cout<<"5-для поиска \n";
      cout<<"6-для выхода\n";
      cin>>er;
      return er;
}

void out()     //ф-ция вывода записей
{
   int sw;    // переключатель для выбора выводить все записи или одну
   int k;     //номер структуры, кот. надо вывести
   if (sch==0) //если счетчик количества структур равен 0, то выводим, что нет
записей
      cout<<"\nНет запичей: \n";
   else
   {
   cout<<"\nВведите: \n";
   cout<<"1-если хотите вывести какую-либо запсь\n";
   cout<<"2-если хотите вывести все записи\n";
   cin>>sw;
   if(sw==1)
   {
         cout<<"Введите номер записи, которую нужно вывести\n";
       cin>>k;
       cout<<endl;
         if (mas[k-1].type==1)
            cout<<"Приход"<<endl;
         else
            cout<<"Отгрузка"<<endl;
         cout<<"Наименование:"<<mas[k-1].name<<endl;
         cout<<"Ед.измер.:"<<mas[k-1].ed<<endl;
```

```
        cout<<"Цена:"<<mas[k-1].cena<<endl;
          cout<<"Кол-во:"<<mas[k-1].kol<<endl;
          cout<<"Дата:"<<mas[k-1].date<<endl;
          cout<<"_____________"<<endl;

    }
    if(sw==2)
    {  for(int i=0;i<sch;i++) //выводим в цикле все записи
       {if (mas[i].type==1)
             cout<<"Приход"<<endl;
         else
            cout<<"Отгрузка"<<endl;
          cout<<"Наименование:"<<mas[i].name<<endl;//выводим на экран значение
name i-ой структуры из массива структур mas
          cout<<"Ед.измер.:"<<mas[i].ed<<endl;
        cout<<"Цена:"<<mas[i].cena<<endl;
          cout<<"Кол-во:"<<mas[i].kol<<endl;
          cout<<"Дата:"<<mas[i].date<<endl;
          cout<<"_____________"<<endl;
        }
    }
    }
}

void del()         //ф-ция удаления записи
{     int d;         //номер записи, которую нужно удалить
     cout<<"\nВведите номер записи, которую необходимо удалить\n";
     cout<<"Если необходимо удалить все записи,нажмите '99'\n";
     cin>>d;
     if (d!=99)
     { for (int i=(d-1);i<sch;i++) //цикл для удаления заданной записи, начинаем
цикл с удаляемой записи
            mas[i]=mas[i+1]; //замещаем текущую запись следующей за ней
       sch=sch-1; //уменьшаем счетчик полных записей на 1
       }
     if (d==99)
       {    for(int i=0;i<30;i++)//цикл по все записям от первой до 30-ой
            mas[i]=tmp; //замещаем каждую структуру в массиве пустой структурой
             sch=0; //счетчик структур обнуляем, т.к. все записи удалены
       }
}

void change() //функция для изменения записи
{   int c;                                          //номер записи, которую
нужно изменить
     int per;
       cout<<"\nВведите номер записи\n";
       cin>> c;
       do
   {cout<<"Введите: \n";
       cout<<"1-для изменения типа накладной\n";
       cout<<"2-для изменения наименования\n";
       cout<<"3-для изменения ед.измерения\n";
       cout<<"4-для изменения цены\n";
       cout<<"5-для изменения количества\n";
       cout<<"6-для изменения даты\n";
       cout<<"7-для прекращения\n";
       cin>>per;
       switch (per)
       {     case 1: cout<<"\nВведите новый тип накладной (1 - приход, 2 -
отгрузка)\n";
             cin>>mas[c-1].type;
             break;
             case 2:
             cout<<"\nВведите новое наименование\n";
             cin>>mas[c-1].name;
```

```
            break;
            case 3:
            cout<<"Введите новые ед.измерения \n";
            cin>>mas[c-1].ed;
            break;
            case 4:
            cout<<"Введите новую цену\n";
            cin>>mas[c-1].cena;
            break;
            case 5:
            cout<<"Введите новое кол-во\n";
            cin>>mas[c-1].kol;
            break;
            case 6:
            cout<<"Введите новую дату последнего поступления\n";
            cin>>mas[c-1].date;
            break;
            case 7: return;
      }

      }while(1);

}

void find()   //ф-ция поиска записей
{
   int sw;      // переключатель
   if (sch==0)
      cout<<"\nНет запичей: \n";
   else
   {
   cout<<"\nВведите: \n";
   cout<<"1-все накладные прихода\n";
   cout<<"2-все накладные отгрузки\n";
   cin>>sw;
   for(int i=0;i<sch;i++) //в цикле просматриваем все структуры из массива
структур
      if (mas[i].type==sw)
      {
         if (mas[i].type==1)
            cout<<"Приход"<<endl;
         else
            cout<<"Отгрузка"<<endl;
          cout<<"Наименование:"<<mas[i].name<<endl;
          cout<<"Ед.измер.:"<<mas[i].ed<<endl;
        cout<<"Цена:"<<mas[i].cena<<endl;
          cout<<"Кол-во:"<<mas[i].kol<<endl;
          cout<<"Дата:"<<mas[i].date<<endl;
          cout<<"____________"<<endl;
      }
   }
}
```

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Варианты заданий

Задача 1 - максимум 4-5 баллов при сдаче на том же занятии, на котором задача выдана. При сдаче на следующих занятиях максимум - 2 балла.

Задача 2 - максимум 5 баллов при сдаче на том же занятии, на котором задача выдана. При сдаче на следующих занятиях максимум - 3-4 балла. Принимается частичное решение, если это позволяет условие задачи.

Задача 1 должна быть сдана обязательно.

Вариант	Задание
1	1. Ввести с клавиатуры массив структур, содержащих сведения об аккаунтах: логин, ФИО, число, месяц и год рождения. Вывести имена людей, родившихся в заданном месяце, указывая при этом его/ее количество лет. 2. Добавить в задание 1 возможность вывести все записи, отсортированные по логину по возрастанию и по убыванию (направление сортировки задается с клавиатуры), вывод как номера, так и названия месяца, используя для формата вывода перечисление (enum). Добавить в структуру поле “пароль”. Сделать возможность вывести аккаунты со слабым паролем (не отвечающие всем требованиям: больше 8 символов, есть один из символов “заглавная буква, цифра, специальный символ”).
2	1. Сформировать массив из n структур, содержащих сведения о лекарствах, хранящихся на аптечном складе. Структура записи: наименование лекарственного препарата, количество, цена, срок хранения (в месяцах). Вычислить, сколько стоят все препараты, хранящиеся на складе, и вывести сведения о препаратах, срок хранения которых более X месяцев. Значение X ввести с клавиатуры. 2. В задание 1 добавить к записи лекарства дату производства и тип применения (внутреннее, наружное, другое). По запросу вывести все лекарства, у которых истек срок годности к текущему моменту, а также у которых срок годности истечет через X месяцев. Добавить второй склад и возможность перемещать лекарства между складами, вывод всех лекарств на каждом складе.

3	1. Даны сведения о телефонах абонентов. Каждая запись имеет поля: фамилия абонента, год установки телефона, номер телефона. Определить количество установленных телефонов, начиная с заданного года; по вводимой фамилии абонента выдать номер телефона. 2. В задание 1 добавить в запись тариф. Для хранения тарифа использовать перечисление. Отдельно добавить массив структур, описывающих тариф и содержащих тип тарифа, его имя, абонентскую плату и скорость соединения. В зависимости от тарифа дополнительно выводить для абонента абонентскую плату и скорость соединения. Добавить возможность изменять данные о тарифах, учитывать изменения, внесенные в массив тарифов, при выводе данных об абонентах.
4	1. Сформировать массив из n структур, содержащих сведения о файлах: имя файла, размер, состояние в виде перечисления (открыт/закрыт), имя создателя. Реализовать возможность закрыть файл, открыть файл (реальные действия с файлами производить не нужно, только изменить значения в структурах), файл ищется по имени. 2. Реализовать возможность удалить файл, учесть, что открытый файл удалить нельзя. Добавить в задание 1 возможность создавать папки, выводить список файлов в папке, добавлять файлы и папки. Выводить список файлов и папок для конкретной папки.
5	1. Сформировать массив из n структур, представляющих блюдо или напиток (тип блюда - enum). У каждого элемента должно быть название, калорийность, содержание жиров, белков и углеводов. Из существующих блюд составить наименее калорийный завтрак, обед и ужин с учетом того, что в каждом приеме пищи должно быть как твердое блюдо, так и напиток, а также то, что блюда не должны повторяться. 2. В задание 1 добавить возможность вручную составлять рацион на завтрак, обед и ужин по названиям блюд (добавление и удаление блюд в каждый из приемов пищи). Добавить возможность задания граничных дневных значений калорий, белков, жиров и углеводов. После составления всех приемов пищи вывести по каждому показателю, превышен он или нет, а также какова разница между граничным и фактическим значением.

6	1. Реализовать эмулятор банковских переводов. Сформировать массив из n счетов банка, содержащих имя владельца счета, его уникальный идентификатор, текущую сумму, тип валюты (BYN/ EUR/USD). Сделать возможность осуществления транзакции (добавления суммы на счете, уменьшения суммы на счете). Если валюта отличается от валюты счета, автоматически конвертировать сумму в валюту счета. 2. В задание 1 добавить один или несколько банков. Добавить в эти банки счета. Реализовать механизм транзакций между банками: сумму можно переводить как на свой счет в другом банке, так и на чужой. При переводе на счет другого банка взымается комиссия. Добавить возможность закрыть счет.
7	1. Сформировать массив из n структур, представляющих автомобили: марка, модель, объем двигателя, привод (enum - передний, задний, полный), потребление топлива, цена. Для ожидаемого пробега, введенного с клавиатуры, выбрать наиболее бюджетные варианты для покупки, от самых дешевых к самым дорогим (учитывая цену авто и потраченного в будущем топлива). 2. Добавить структуру, представляющую объявление о продаже б/ у автомобиля. Эта структура содержит автомобиль, цену, телефон. Добавить поиск объявления по строке, содержащей частичное имя марки и модели автомобиля. Предусмотреть, что при изменении данных в описании автомобиля, эти изменения отражаются во всех объявлениях, содержащих этот автомобиль. Соответственно, добавить возможность изменения данных в структуре автомобиля.
8	1. Реализовать структуру String, которая представляет строку. Структура хранит массив символов и длину строки. Добавить возможность добавления одной строки к другой, получения подстроки, состоящей из первых N символов строки. Эти функции должны работать со структурами String, должна быть возможность проверить их, задавая строки и действия с клавиатуры. 2. К структуре из задания 1 добавить поле “локаль” (перечисление: английская, русская и т.д.). Добавить структуру LocalizedString, хранящую идентификатор и массив значений типа String - по одной для каждой локали. Сделать возможность ввести несколько локализованных строк, содержащих по строке для каждой локали. Сделать возможность сменить локаль приложения и вывести строку по идентификатору для текущей локали.

9	1. Реализовать структуру, которая представляет собой объявление об аренде квартиры. Структура содержит адрес, этаж, площадь, количество комнат, тип стен (enum - панельный, кирпичный, блочный) и цену аренды. Сделать возможность поиска квартиры с ценой не выше N и кол-вом комнат не меньше M. Добавить возможность вывода средней цены квартиры для каждого кол-ва комнат, а также средней цены аренды квадратного метра. 2. Добавить структуру, которая представляет собой арендатора. Эта структура содержит фамилию, максимальную цену, которую он готов заплатить за аренду, минимальное кол-во комнат. По запросу с клавиатуры организовать сделки - каждому арендатору найти квартиру, подходящую его паметрам. Сделать возможность вывода текущих сделок (квартира и арендатор, снимающий эту квартиру), неарендованных квартир, бездомных арендаторов. Добавить возможность менять данные квартир и арендаторов.
10	1. Реализовать структуру “карта”, которая содержит перечисления “ранг” и “масть”. С клавиатуры вводить код карты вида “JC”, “10D”, “AS”, “2H”. Из введенных данных создать структуру “карта” и вывести ее полное название: Jack of Clubs, 10 of Diamonds, Ace of Spades, Two of Hearts соответственно. С клавиатуры задать произвольное количество карт, проверить, корректная ли это комбинация для одной колоды (повторяются ли карты). 2. Сделать возможность добавить 5 карт и вывести для них наивысшую комбинацию из покера. Комбинации в порядке возрастания: старшая карта (карта с наибольшим рангом), пара (две карты с одинаковым рангом), две пары, тройка (три карты с одинаковым рангом), стрит (пять последовательных по рангу карт), флеш (пять карт одной масти), фулл-хауз (тройка и пара), каре (4 карты одного ранга), стрит флеш (5 последовательных одномастных карт), флеш рояль (10, валет, дама, король, туз одной масти).
11	1. Реализовать структуру “игрок”, которая содержит кол-во жизней, броню, пол (enum - мужчина, женщина, другой) уровень сытости и насыщенности водой. Сделать несколько типов оружия с разной мощью, аптечку, еду и воду. Добавить возможность с клавиатуры использовать все эти предметы по отношению к игроку. Выводить его статус. 2. Добавить оружию дальность. Добавить игроку координату, скорость перемещения и оружие. Сделать возможность сталкивать двух игроков, которые с определенным интервалом времени выполняют действия друг против друга, отображать их статус.

12	1. Реализовать структуру “параграф”, которая содержит кол-во строк, длину строки, размер шрифта, междустрочный интервал, интервал до параграфа, интервал после параграфа (размер шрифта и интервалы - значения высоты в пикселях). Сделать возможность добавить произвольное число параграфов с клавиатуры, задавая при этом значения всех полей структуры. Сделать подсчет высоты всего текста в пикселях. 2. Добавить структуру “страница”, которая имеет определенную высоту в пикселях. В зависимости от числа параграфов вычислять кол-во страниц текста. В структуру “параграф” добавить поле “цвет” (enum - черный, красный, синий и т.д.). Высчитать стоимость печать текста в зависимости от кол-ва страниц и кол-ва цветного текста.
13	1. Реализовать структуру “пассажир”, которая содержит ФИО, номер карточки, класс желаемого такси (enum - эконом, кофморт, бизнес), кол-во денег на счету, координату (x,y). Реализовать структуру “такси”, которая содержит координату (x, y), класс, номер машины. Сделать возможность задать несколько пассажиров и несколько машин такси. Для пассажира выбрать ближайшую машину, снять с него деньги за поездку. Позволить просмотреть все данные. 2. Сделать возможность для нескольких пассажиров добавить точку назначения, по кнопке рассчитать оптимальную подачу машин для всех, обновить данные, выдать всем пассажирам чеки с суммой, расстоянием, номером машины. Для каждого пассажира можно просмотреть все чеки.
14	1. Реализовать структуру “лицо”, которая содержит перечисления для полей “нос” (например, прямой, горбинкой и т.д.), “рот”, “уши”, “форма глаз”, “цвет глаз”, "особые приметы" (шрам, родинка и т.д.). Задать базу преступников (структура содержит лицо и ФИО). Сделать возможность с клавиатуры ввести параметры лица. Как результат выдать для каждого преступника процент совпадения внешности с заданной (в зависимости от количества совпавших черт лица). 2. Для изменения каждой части лица задать цену. Сделать возможность высчитать стоимость трансформацию из одного лица в другое. В структуру “преступник” добавить количество денег. Отдельно задать лица добропорядочных людей. По запросу выдать преступников, у которых хватает денег на то, чтобы трансформироваться в добропорядочных людей.

15	1. Реализовать структуру “сообщение”, которая содержит текст сообщения (обязательно), id отправителя (строка, обязательно), id получателя (строка, обязательно), опционально ссылку на вложение и тип вложения (enum - изображение, видео, бинарный файл). Сделать возможность эмулировать в приложении наличие подключения к сети. При отправке сообщения проверяется наличие сети. Если сети нет, то сообщение должно отправиться при следующем появлении сети. Сделать возможность вывода всех неотправленных сообщений. Добавить поиск среди отправленных сообщений по id получателя. 2. Добавить список существующих идентификаторов пользователей. При отправке несуществующему пользователю выдавать ошибку. Добавить возможность удаления конкретного сообщения. Добавить возможность ответа на конкретное сообщение (при просмотре сообщения должна быть возможность просмотра сообщения, на которое сделан ответ).