# АЛГОРИТМИЗАЦИЯ
# И ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Учебно-методический комплекс

Методические указания
к практическим работам

## Оглавление

## Практическая работа № 1
## Запись простейших алгоритмов обработки числовых данных

**Цель работы** – ознакомиться с ГОСТом 19.701-90 (ИСО 5807-85), введенным к обязательному применению с 01.01.92 г. и регламентирующим правила записи алгоритмов; получить навыки записи базовых алгоритмических структур, исследовать преимущества счетчиков и аккумуляторов при разработке алгоритмов работы с числовыми массивами.

1. Найти в Интернете и изучить ГОСТ 19.701-90 (ИСО 5807-85) в части требований к оформлению схем алгоритмов.

2. Используя примеры алгоритмов из опорного конспекта, разработать и записать в виде блок-схем алгоритмы решения следующих задач:

- ввести два целых числа, минимальное из них удвоить и результат вывести на экран;
- ввести три целых числа, вычислить среднее арифметическое этих чисел.
- ввести три целых числа, вычислить $y = \min(a,b,c) + \max(a,b,c)$, т.е. сумму минимального и максимального из них;
- ввести три целых числа, вычислить $y = \max(a,b,c) - \min(a,b,c)$, т.е. разность максимального и минимального из них;
- ввести одномерный массив целых чисел, определить сумму его элементов;
- ввести одномерный массив целых чисел, определить номер первого отрицательного элемента;
- ввести одномерный массив целых чисел, определить сумму его отрицательных элементов;
- ввести одномерный массив целых чисел, определить его максимальный элемент.

**Результат выполнения практической работы**: Вы ознакомились с утвержденными ГОСТом правилами записи алгоритма решения задачи в виде блок-схемы и научились записывать простейшие алгоритмы обработки числовых данных. Вы исследовали типовые приемы организации

суммирования числовых данных с помощью переменной-счетчика и переменной-аккумулятора.

Форма представления результатов практической работы: отчет на бумажном носителе с блок-схемами решения задач.

## Практическая работа № 2
## Исследование визуальной среды программирования
## Microsoft Visual Basic 6.0 для операционной системы Windows

**Цель работы** – исследовать возможности визуальной инструментальной среды и овладеть навыками работы в ней при создании простейшего приложения для операционной системы Windows.

### Этап 2.1. Исследование рабочего окна системы программирования

1. Запустить `MS Visual Basic 6.0`

2. Выбрать вид нового файла `Standart.exe` (выбирается по умолчанию при нажатии на кнопку `Open`)

3. На рабочем столе появилась пустая форма в виде серого прямоугольника на белом фоне.

4. Увеличить размер формы можно путем протаскивания за диагональный маркер. Форма должна занимать значительную часть экрана, но не выходить за его пределы.

5. В правой нижней части экрана расположено окно свойств, в котором отображается список свойств активного элемента с их значениями. В начальный момент активной является новая форма, ее свойствам `Name` и `Caption` приписаны значения по умолчанию `Form1`. Измените значение свойства `Caption` на произвольное. Новое имя появилось в синей строке заголовка формы.

6. В правой верхней части экрана располагается окно проекта, в котором отображается структура создаваемого приложения. В начальный момент в проект входит только форма `Form1`. Помимо форм в проект могут входить отдельные программные модули. Добавить в проект дополнительную форму или модуль можно через диалог меню `Add...`

7. В левой части экрана располагается панель элементов управления. При наведении на элементы курсора мыши появляется всплывающая подсказка с названием элемента. Для размещения элемента на форме

достаточно кликнуть на элементе, затем очертить курсором область на форме, где будет располагаться элемент. Активный элемент выделен граничными маркерами, а его свойства отображаются в окне свойств.

8. В процессе создания дизайна экранной формы каждому элементу управления присваиваются значения его свойств, основным из которых является свойство `Name`. По значению этого свойства данный элемент связывается с программным кодом обработки события.

9. Окно программного кода можно вызвать через диалог меню `View->Code`. Текстовый редактор позволяет записывать тексты программы на языке `Visual Basic` и контролирует наличие синтаксических ошибок.

10. Для запуска приложения используются кнопка `Start` основного меню или клавиша `F5`. Активное приложение, состоящее из экранной формы и программного кода обработки событий, готово к действиям пользователя. Для возврата в режим редактирования формы следует остановить работу приложения кнопкой `Stop` основного меню.

11. Готовое приложение следует сохранить как проект через диалог меню `Save Project As`... Для сохранения проекта необходимо выделить отдельную папку. Проект будет сохранен как набор файлов, содержащих как минимум файл формы и файл проекта. Двойной клик на файле проекта приведет к активизации системы программирования и открытию формы для редактирования.

**Результат выполнения этапа**: Вы познакомились со структурой рабочего окна визуальной среды программирования и важнейшими пунктами меню, позволяющими создать дизайн экранной формы, записать текст программного кода, сохранить приложение как проект, активизировать приложение и остановить его работу.

**Форма представления результатов работы**: папка с файлами проекта.

**Этап 2.2. Создание проекта «Фигура»**

1. Запустить MS Visual Basic 6.0
2. Выбрать вид нового файла Standart.exe.
3. Разместить на форме элементы управления в соответствии с рис. 1 и определить значения свойств каждого элемента формы в соответствии с табл. 1.

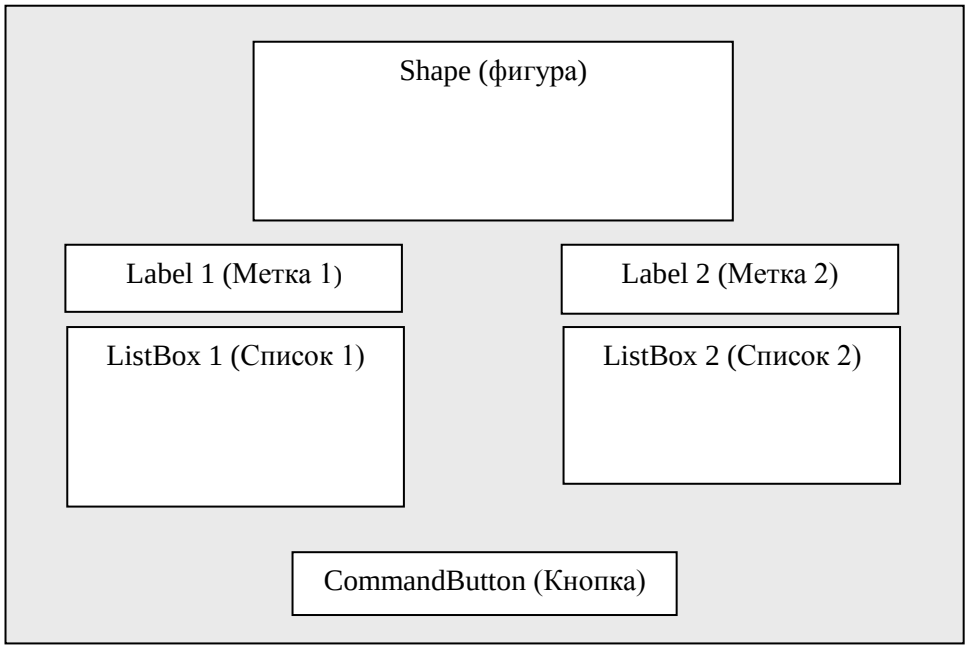


Рис. 1 Дизайн формы для проекта «Фигура»

Таблица 1. Свойства и значения элементов управления

| **Объект** | **Элемент на панели** | **Свойство** | **Значение** |
|---|---|---|---|
| Форма | Form | Name | frmShape |
| | | Caption | Shaping Up |
| Фигура | Shape | Name | shpSample |
| Метка 1 | Label | Name | lblShape |
| | | Caption | shape |
| Метка 2 | Label | Name | lblPattern |
| | | Caption | Pattern |
| Список 1 | ListBox | Name | lstShape |
| Список 2 | ListBox | Name | lstPattern |
| Кнопка | CommandButton | Name | cmdExit |
| | | Caption | E&xit |

4. Для приложения необходимо написать программный код, который инициализирует списки значениями, отображаемыми на экране при запуске программы, и будет реагировать на действия пользователя.

5. Для перехода к окну программного кода выберите пункт меню `View -> Code` . В верхней части окна редактора располагаются два раскрывающихся списка: список объектов приложения (форма, фигура, два списка, две метки и командная кнопка, чьи имена соответствуют именам из таблицы свойств) и список допустимых событий каждого из этих объектов. Необходимо создать программный код только для тех событий, на которые будет реагировать приложение.
6. Выберите в левом раскрывающемся списке элемент `Form`. В правом раскрывающемся списке автоматически выделилось событие `Load`, а в окне редактора появились начальные и конечные строки кода. Введите между ними остальные строки обработчика события из листинга 1. Код представляет собой список действий (методов). Например, метод `AddItem` создает список.

**Листинг 1.** Инициализация списков

```
Private Sub Form_Load()
   ' Инициализация раскрывающегося списка Shape
  LstShape.AddItem  "0 - Rectangle"
  LstShape.AddItem  "1 - Square"
  LstShape.AddItem  "2 - Oval"
  LstShape.AddItem  "3 - Circle"
  LstShape.AddItem  "4 - Rounded Rectangle"
  LstShape.AddItem  "5 - Rounded Square"
   ' Инициализация раскрывающегося списка FiIIStyle
  LstPattern.AddItem  "0 - Solid"
  LstPattern.AddItem  "1 - Transparent"
  LstPattern.AddItem  "2 - Horizontal Line"
  LstPattern.AddItem  "3 - Vertical Line"
  LstPattern.AddItem  "4 - Upward Diagonal"
  LstPattern.AddItem  "5 - Downward Diagonal"
  LstPattern.AddItem  "6 - Cross"
  LstPattern.AddItem  "7 - Diagonal Cross"
   ' Установить начальные значения для каждого списка
  LstShape.ListIndex = 0
  LstPattern.ListIndex = 0
End Sub
```

7. Выберите в левом верхнем раскрывающемся списке редактора объект `LstPattern`. В правом раскрывающемся списке автоматически выделилось событие `Click`. Введите для обработчика первого

раскрывающегося списка программный код, представленный в листинге 2.

**Листинг 2.** Изменение способа заливки фигуры

```
Private Sub LstPattern_Click()
   ' Изменить вид в соответствии с выбором
   shpSample.FillStyle=LstPattern.ListIndex
End Sub
```

8. Выберите в левом верхнем раскрывающемся списке редактора объект `LstShape`. В правом раскрывающемся списке автоматически выделилось событие `Click`. Введите для обработчика второго раскрывающегося списка программный код, представленный в листинге 3.

**Листинг 3.** Изменение фигуры

```
Private Sub LstShape_Click()
     ' Изменить фигуру в соответствии с выбором
     shpSample.Shape = LstShape.ListIndex
End Sub
```

9. Выберите в левом верхнем раскрывающемся списке редактора объект `cmdExit`. В правом раскрывающемся списке автоматически выделилось событие `Click`. Введите для обработчика командной кнопки программный код, представленный в листинге 4. Оператор `End` заставляет `Visual Basic` прекратить выполнение программы по клику на кнопке выхода..

**Листинг 4.** Завершение программы

```
Private Sub cmdExit_Click()
     End
End Sub
```

10. Нажмите на клавишу `F5`. Если программный код набран безошибочно, введены все элементы управления и установлены значения их свойств в соответствии с табл. 1, приложение должно запуститься. Кликните на любом из списков и наблюдайте за изменениями на форме. Приложение позволяет исследовать некоторые геометрические фигуры и способы их заливки. Для завершения работы приложения кликните

на кнопке Exit.

11. Если приложение не заработало или получено сообщение об ошибке, то следует вернуться к окну программного кода, исправить все ошибки и проверить все значения по табл. 1.
12. Сохраните проект в отдельной папке. Проверьте ее содержимое: в ней сохранился файл экранной формы и файл проекта.

**Результат выполнения этапа**: Вы научились создавать дизайн экранной формы, располагая на ней элементы управления и определяя их свойства. С каждым элементом управления Вы связали программный код, который реагирует на действия пользователя. Вы научились сохранять проект как совокупность файлов, запускать приложение и вызывать события с помощью кликов на элементах формы. Таким образом, Вы познакомились с возможностями визуальной системы программирования Microsoft Visual Basic 6.0.

**Форма представления результатов работы**: папка с файлами проекта.

## Практическая работа № 3
## Технология алгоритмического программирования

**Цель работы** – исследовать возможности командной инструментальной среды TurboPascal 7.0 и овладеть навыками разработки простейших программ обработки числовых и символьных данных на языке Pascal.

**Этап 3.1. Исследование возможностей системы программирования TurboPascal 7.0 для операционной системы MS DOS.**

1. Для начала работы в системе программирования TurboPascal 7.0 необходимо подготовить рабочую папку с коротким (8 символов) именем в латинском алфавите (например, /TP) и в ней создать папку /TP/Prog для хранения собственных программ.
2. Скопировать файлы TurboPascal 7.0 в папку /TP и запустить на выполнение файл Turbo.exe.При этом на экране появляется окно редактора с синим фоном, верхней линейкой меню и нижней линейкой с функциональными клавишами. Система программирования лишена графического интерфейса, реагирует на действия мышью и нажатие клавиш на клавиатуре.
3. Первоначально в окне редактора открыт пустой безымянный файл, который можно сохранить под собственным именем. Для сохранения файла выбрать пункт меню File->Save As… В диалоге установить путь к папке /TP/Prog и ввести имя файла не более чем из восьми латинских символов (например, task_1). Проверить, что файл task_1.pas действительно появился в папке.
4. В процессе редактирования текста программы файл рекомендуется периодически сохранять, выбирая меню File->Save или нажимая функциональную клавишу F2.
5. Для открытия существующего файла использовать команду File->Open (клавиша F3).

6. После записи текста программы необходимо вызвать компилятор для проверки синтаксических ошибок и генерации объектного кода. Вызов компилятора происходит при выборе пункта меню `Compile-> Compile` или одновременном нажатии клавиш `Alt+F9`.
7. Если все синтаксические ошибки в тексте программы исправлены и компиляция завершилась успешно, программу следует запустить на выполнение, выбрав пункт меню `Run->Run` или одновременном нажатии клавиш `Ctrl+F9`.
8. Если в программе предусмотрен ввод/вывод данных, то при запуске появится окно программы с черным фоном, в которое будет производится вывод. По завершении программы для просмотра данного окна необходимо выбрать пункт меню `Debug->User screen` или воспользоваться одновременным нажатием клавиш `Alt+F5`.
9. Для отладки программы (поиска семантических ошибок) предназначены особые режимы запуска программы, доступные по следующим пунктам меню:
   - `Run->Step over` – выполнение программы по шагам,
   - `Run->Go to cursor` – программа выполняется до зафиксированного положения курсора в тексте и останавливается, дальше можно выбрать ее пошаговое выполнение
10. Для просмотра текущих значений переменных при выполнении программы по шагам можно воспользоваться пунктом меню `Debug->Evaluate/modify` или одновременным нажатием клавиш `Ctrl+F4`. В этом режиме следует указать имена переменных, значения которых подлежат проверки, и наблюдать их изменения при пошаговом выполнении программы.

11. Для завершения работы в системе программирования необходимо выбрать пункт меню `File->Exit` или сочетание клавиш `Alt+x`.

**Результат выполнения этапа**: Вы познакомились со структурой рабочего окна командной среды программирования `TurboPascal 7.0` и важнейшими пунктами меню, позволяющими создать текст программы, провести процесс компиляции, запустить на выполнение, просмотреть результаты в различных режимах и сохранить программу в виде файла.

**Этап 3.2. Программы обработки числовых данных .**

1. Вам предстоит, действуя по шаблонам и используя разработанные ранее алгоритмы, создать простейшие программы обработки числовых данных и сохранить их в виде файлов `int_1.pas, int_2.pas` и т.д. для простых чисел, `arr_1.pas, arr_2.pas` и т.д. для массивов.

2. Переписать из листинга № 1 и отладить программу суммирования двух целых чисел.

**Листинг 1.** Ввести два целых числа. Вычислить их сумму и вывести результат на экран.

```
Program int_1;
Uses CRT;
Var   a, b, c: integer;
BEGIN
  ClrScr;
  Write('Введите два целых числа:');
  Readln(a, b);
   c := a + b;
  write('Результат сложения чисел ', a, ' и ', b, ': ', c);
END.
```

3. По аналогии с предыдущим заданием, используя оператор ветвления, написать и отладить программы:

- ввести два целых числа. Вывести их на экран в порядке возрастания;
- ввести два целых числа. Найти их разницу и результат вывести на экран;
- ввести два целых числа. К максимальному из них прибавить 10 и результат вывести на экран;

- ввести два целых числа. Минимальное из них удвоить и результат вывести на экран.

4. Переписать из листинга № 2 и отладить программу обработки массива целых чисел.

**Листинг 2.** Ввести одномерный массив целых чисел. Преобразовать его, заменив положительные элементы 1, отрицательные – 0. Результат вывести на экран.

```
Program arr_1;
Uses CRT;
Const n=10;
Var   A: array[1..n] of integer;
i: byte;
BEGIN
  ClrScr;
  Write ('Введите элементы массива: ');
  for i:=1 to n do
      readln (A[i]);
  for i:=1 to n do
   if A[i] > 0 then
        A[i]:=1
   else
        A[i]:=0;
  write (‘измененный массив = ');
  for i:=1 to n do
         write (A[i],' ');
END.
```

5. По аналогии с предыдущим заданием, используя оператор цикла, написать и отладить программы:

- ввести одномерный массив целых чисел, определить сумму его элементов, результат вывести на экран;
- ввести одномерный массив целых чисел, определить сумму его отрицательных элементов, результат вывести на экран;
- ввести одномерный массив целых чисел, определить его максимальный элемент, результат вывести на экран
- ввести одномерный массив целых чисел, определить сумму его максимального и минимального элементов, результат вывести на экран
- ввести одномерный массив целых чисел, из положительных элементов массива сформировать новый массив и вывести его на

экран.

- ввести одномерный массив целых чисел, определить номер первого отрицательного элемента, результат вывести на экран (использовать команду `Break` для прерывания цикла).

6. Переписать из листинга № 3 и отладить программу обработки двумерного массива целых чисел.

**Листинг 3.** Ввести двумерный массив целых чисел, найти максимальный элемент, результат вывести на экран

```
Program arr_2;
Uses CRT;
Const n = 4, m = 5;
Var A: array[1..n, 1..m] of integer;
    i, j, max: integer;
BEGIN
ClrScr;
Write('Введите элементы двумерного массива размерности ',n, '
x ',m, ': ');
for i:=1 to n do
   for j:=1 to m do
        ReadLn (A[i,j]);
max := A[1,1];
for i:=1 to n do
   for j:=1 to m do
        if A[i,j] > max then
             max := A[i,j];
Write('Максимальный элемент двумерного массива: ', max);
END.
```

7. По аналогии с предыдущим заданием, используя вложенные операторы цикла, написать и отладить программы:

- ввести двумерный массив целых чисел; на его основе создать новый одномерный массив, каждый элемент которого – максимальное отрицательное число в каждой строке исходного массива (см. пример на рис. 2);

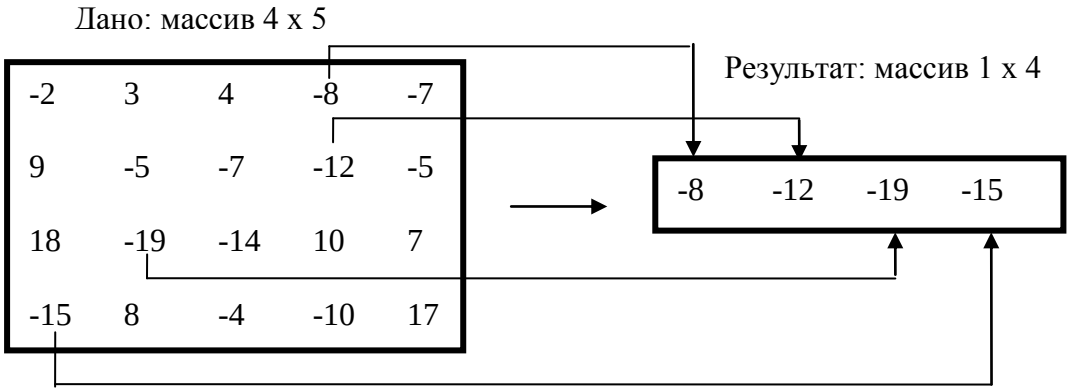


Рис. 2 поиск максимального отрицательного числа в двумерном массиве

- ввести двумерный массив целых чисел; на его основе создать новый одномерный массив, каждый элемент которого − индекс наибольшего числа в каждой строке исходного массива (рис. 3);

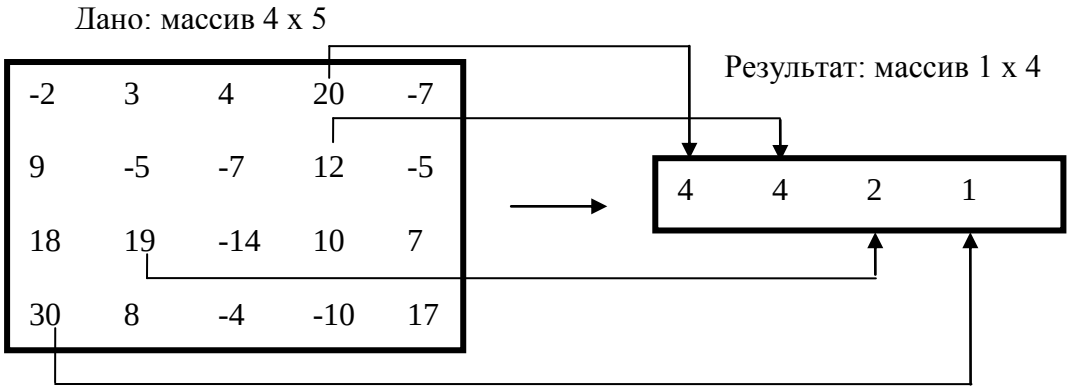


Рис. 3. Поиск координаты максимального элемента в строках двумерного массива

**Результат выполнения этапа**: Вы научились записывать разработанные ранее алгоритмы обработки целых чисел в виде программ на алгоритмическом языке `Pascal`. Вы получили представление о структуре программы и назначении ее разделов. Вы узнали и проверили на практике, какими операторами языка записываются базовые алгоритмические структуры ветвления и цикла и как работает оператор присваивания. Вы научились обрабатывать одномерные и двумерные числовые массивы с помощью вложенных циклов.

**Этап 3.3. Программы обработки символьных данных .**

1. Вам предстоит, действуя по шаблонам, создать простейшие программы обработки символьных данных и сохранить их в виде файлов `char_1.pas, char_2.pas`.

2. Переписать из листинга № 4 и отладить программу вывода на экран букв латинского алфавита, расположив их в строку.

**Листинг 4.** Вывести на экран символы латинского алфавита, расположив их в строку, используя цикл `For … to …`

```
Program char_1;
USES CRT;
VAR  i: char;
BEGIN
     CLRSCR;
     Write('Изучаем алфавит: ')
     for i:= 'A' to 'Z' do
          Write (i);
     Writeln;
END.
```

3. По аналогии с предыдущим заданием написать и отладить программу вывода на экран символов латинского алфавита, расположив их в строку в обратном порядке, используя оператор цикла `For … downto …`

4. Переписать из листинга № 5 и отладить программу вывода на экран букв латинского алфавита, расположив их в строку в особом порядке.

**Листинг 5.** Вывести на экран символы латинского алфавита, расположив их в строку в следующем порядке: `ABBCCCDDDD … ZZZ … Z`

```
Program char_2;
USES CRT;
VAR  i: char;
     k, j: integer;
BEGIN
     CLRSCR;
     k:=1;
     for i:= 'A' to 'Z' do
          BEGIN
               For j:=1 to k do
                    Write (i);
                    k:=k+1;
          END;
END.
```

5. По аналогии с предыдущим заданием, используя оператор цикла, написать и отладить программы:

- вывести в одну строку символы в следующем порядке: `Z Y Y X X X…A A…A`;
- вывести символы в следующем порядке (число строк равно числу символов в алфавите):

```
A B C D … Z
A B C D … Z
A B C D … Z
     …
A B C D … Z
```

– вывести треугольник, состоящий из букв латинского алфавита, используя функцию `SUCC(char)`, которая возвращает следующий символ после `char`:

```
A B C …  Z
B C …  Z
C …  Z
… Z
Z
```

– вывести треугольник другой конфигурации:

```
A B C … Z
    B C … Z
        C …  Z
            …  Z
                Z
```

**Результат выполнения этапа**: Вы ознакомились с приемами обработки символьных данных на примере латинского алфавита, узнали новые возможности организации цикла `For`, применили функцию-генератор следующего символа. Вы получили навыки форматированного вывода символов на экран.

**Этап 3.4. Программы обработки строковых данных**

1. Вам предстоит, действуя по шаблонам, создать простейшие программы обработки строковых (текстовых) данных и сохранить их в виде файлов `str_1.pas, str_2.pas`.

2. Переписать из листинга № 6 и отладить программу ввода и редактирования текста.

**Листинг 6.** Ввести текст с клавиатуры. Вывести его на экран до первого знака препинания.

```
Program str_1;
USES CRT;
VAR  str: string;
     k, i : integer;
     m: set of char;
BEGIN
     CLRSCR;
     m:= ['.', ',', '!', '? ', ':', ';', '-'];
```

```
        writeln ('Введите текст');
        readln (str);
        k:=length (str);
        writeln ('Печатаем текст до первого знака препинания:');
        for i:=1 to k do
             If not (str[i] in m) then
                  write (str[i])
             else
                  break;
END.
```

3. По аналогии с предыдущим заданием написать и отладить программы:
   - ввести текст с клавиатуры, вывести его на экран без знаков препинания;
   - ввести текст с клавиатуры, вывести его на экран без гласных букв, определив их первоначальное количество в тексте.
4. Переписать из листинга № 7 и отладить программу работы с текстовым файлом.

**Листинг 7.** Создать текстовый файл `File1.txt`, заполнить его текстом. Прочитать текст из файла и вывести его на экран.

```
Program file_1
USES CRT;
VAR  F1: text;
     str: string;
BEGIN
     CLRSCR;
     Assign (F1, 'File1.txt');
     Reset (f1);
     Read (f1, str);
     Writeln('В файле записан текст: ');
     Writeln (str);
     Close(f1);
END.
```

5. По аналогии с предыдущим заданием написать и отладить программы:

- создать текстовый файл `File1.txt`, записать в него текст «`Hello, friend!`». Прочитать текст из файла. Преобразовать его в вид: «`Hello, my friend!`». Использовать процедуру `INSERT(Word:string, String:string, pos:integer)`, где `Word` – вставляемый текст, `String` – исходный текст, `pos` – позиция для вставки) и функцию `POS(word:string, str:string):integer`, где `Word` – символ, позицию которого в строке надо найти, `str` –

строка, в которой ведется поиск.

- создать текстовый файл `File2.txt`, записать в нем текст. Прочитать текст из файла. Ввести с клавиатуры строку текста. Вывести на экран объединение текста и введенной строки. Использовать функцию `CONCAT(str1:string, str2:string):string` – возвращает объединение строк.
- создать текстовый файл `File3.txt`, записать в нем текст. Удалить из текста второе слово. Использовать процедуру `DELETE(str:string, i:integer, j:integer)`, где `str` – строка, в которой надо произвести удаление, `i` – позиция, с которой начать удаление, `j` – количество удаляемых символов.
- создать текстовый файл `File4.txt`, записать в нем текст. Прочитать текст из файла, вывести его на экран, разбив на слова по 5 символов.
- создать текстовый файл `File5.txt`, записать в нем текст. Изменить текст в файле, разбив его на слова по 5 символов.

**Результат выполнения этапа**: Вы ознакомились с приемами обработки текстовых данных, вводимых с клавиатуры и из файла, поработали со специфическим типом данных «множество», освоили последовательность работы с файлом, применили функции и процедуры модификации текстовых данных.

**Форма представления результатов работы**: файлы.

## Практическая работа № 4
## Технология структурного программирования

**Цель работы** – исследовать механизм вызова подпрограмм из основной программы и овладеть навыками описания и использования процедур и функций как разновидностей подпрограмм.

1. Вам предстоит, действуя по шаблону, создать программы вызова подпрограмм, передачи им фактических параметров и сохранить их в виде файлов `CallPr_1.pas, CallPr_2.pas`.
2. Переписать из листинга № 1 и отладить программу ввода, модификации и вывода числового массива.

**Листинг 1.** Программа ввода, модификации и вывода числового массива.

```
Program CallPr_1;
Uses CRT;
Const n = 2; m = 4; k = 5;
var
    A: array[1..n] of Integer;
    B: array[1..m] of Integer;
    C: array[1..k] of Integer;
{Раздел описания процедур. Процедура ввода массива}
Procedure Inp (Var Mas: array of Integer; size: Byte);
     Var i: byte;
 begin
      WriteLn (‘Введите ', size, ' целых чисел');
      for i:=1 to size do
          Read (Mas[i]);
 end;
{Процедура модификации массива}
Procedure Replace(Var Mas: array of Integer; size: Byte);
     Var i: byte;
 begin
      for i:=1 to size do
          if Mas[i]>0 then
             Mas[i]:=1
          else
             Mas[i]:=0;
 end;
{Процедура вывода массива}
Procedure Outp (Var Mas: array of Integer; size: Byte);
     Var i: byte;
 begin
      Write('Результат преобразования массива: ');
      for i:=1 to size do
          Write(Mas[i]:2:0);
      WriteLn(' ');
 end;
{Главная программа}
begin
```

```
     ClrScr;
     Inp (A, n);
     Replace (A, n);
     Outp (A, n);
     Inp (B, m);
     Replace (B, m);
     Outp (B, m);
     Inp (C, k);
     Replace (C, k);
     Outp (C, k);
end.
```

3. Переписать из листинга № 2 и отладить программу нахождения максимального элемента в числовом массиве.

**Листинг 2.** Программа поиска максимального элемента в числовом массиве.

```
Program CallPr_2;
Uses CRT;
Const n = 2; m = 4; k = 5;
var
    A: array[1..n] of Integer;
    B: array[1..m] of Integer;
    C: array[1..k] of Integer;
    Max:Integer;
{Раздел описания процедур. Процедура ввода массива}
Procedure Inp (Var Mas: array of Integer; size: Byte);
     Var i: byte;
 begin
      WriteLn ('Введите ', size, ' целых чисел');
      for i:=1 to size do
          Read (Mas[i]);
 end;
{Процедура поиска максимального элемента массива}
Function Findmax(Var Mas: array of Integer; size: Byte):Integer;
     Var  i: byte;
          maxin: Integer;
 begin
          maxin := Mas[1];
          for i:=2 to size do
               if Mas[i]>maxin then
                    maxin := Mas[i];
          Findmax := maxin;
 end;
{Главная программа}
begin
     ClrScr;
     Inp (A, n);
     Max:= Findmax (A, n);
     Writeln ('Максимальный элемент массива: ', max);
     Inp (B, m);
     Max:= Findmax (B, m);
     Writeln ('Максимальный элемент массива: ', max);
     Inp (C, k);
     Max:= Findmax (C, k);
     Writeln ('Максимальный элемент массива: ', max);
end.
```

4. По аналогии с предыдущим заданием написать и отладить программы:
   - вычисления суммы отрицательных элементов в каждом из трех массивов;
   - вычисления разницы максимального и минимального элементов в каждом из трех массивов,
   - вычисления среднего арифметического элементов в каждом из трех массивов.

**Результат выполнения работы**: Вы ознакомились с приемами структурного программирования как разработкой небольших логически законченных подпрограмм, работой которых управляет главная программа. Такой подход позволяет создавать библиотеки подпрограмм для их последующего многократного использования в различных приложениях.

**Форма представления результатов работы**: папка с файлами.

## Практическая работа № 5
## Технология визуального программирования

**Цель работы** – овладеть навыками работы в визуальной среде программирования при создании приложения для операционной системы Windows, освоить приемы обработки некорректных действий пользователя при работе в приложением.

1. Запустить `MS Visual Basic 6.0`
2. Выбрать вид нового файла `Standart.exe`.
3. Разместить на форме элементы управления и определить значения свойства `Name` каждого элемента формы в соответствии рис. 4.

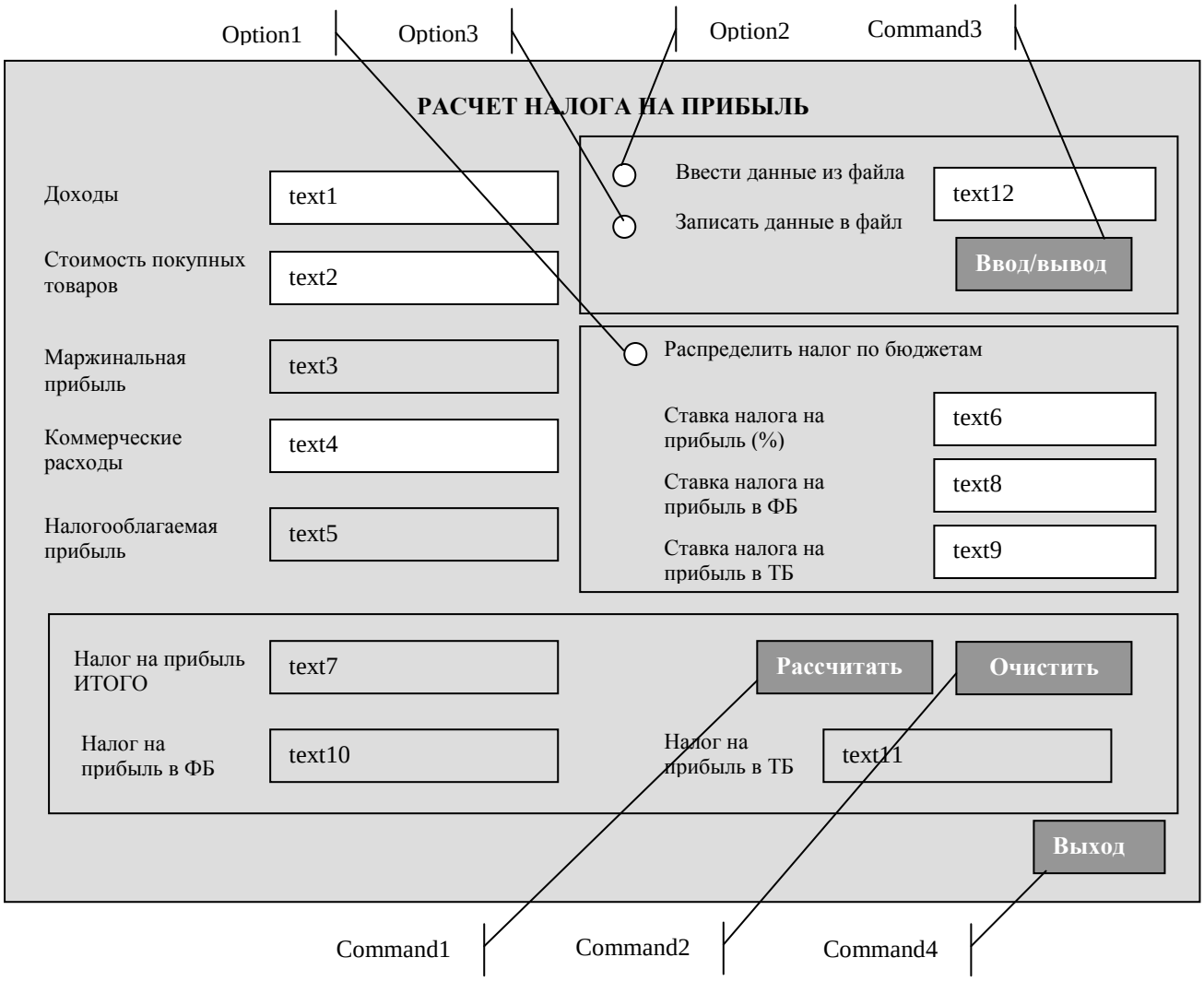


Рис. 4 Дизайн формы для проекта «АРМ экономиста»

4. Добавить к форме программный код обработки событий в соответствии

с листингом 1.

**Листинг 1.** Коды приложения «АРМ экономиста»

```
Private Sub Command1_Click()
If Text1 = "" Then
MsgBox ("Введите сумму доходов")
Text1.SetFocus
GoTo m1
End If
If Text2 = "" Then
MsgBox ("Введите стоимость покупных товаров")
Text2.SetFocus
GoTo m1
End If
If Text4 = "" Then
MsgBox ("Введите коммерческие расходы")
Text4.SetFocus
GoTo m1
End If
If Text6 = "" Then
MsgBox ("Введите ставку налога на прибыль")
Text6.SetFocus
GoTo m1
End If
Text3 = Text1 - Text2
Text5 = Text3 - Text4
Text7 = Text5 * Text6 / 100
If Option1.Value = True Then
Text10 = Text5 * Text8 / 100
Text11 = Text5 * Text9 / 100
End If
m1: End Sub

Private Sub Command2_Click()
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
Text8.Text = ""
Text9.Text = ""
Text10.Text = ""
Text11.Text = ""
Option1.Value = False
End Sub

Private Sub Command3_Click()
Dim w As fil
If Option2.Value = True And Text12 <> "" Then
Open Text12.Text + ".txt" For Random As #1 Len = 2048
Get #1, 1, w
Text1.Text = w.d
Text2.Text = w.t
Text4.Text = w.k
Text6.Text = w.s
Text8.Text = w.sf
Close 1
End If
If Option3.Value = True And Text12 <> "" Then
w.d = Text1.Text
w.t = Text2.Text
```

```
w.k = Text4.Text
w.s = Text6.Text
w.sf = Text8.Text
Open Text12.Text + ".txt" For Random As #1 Len = 2048
Put #1, 1, w
Close 1
End If
End Sub
```

```
Private Sub Command4_Click()
End
End Sub
```

```
Private Sub Option1_Click()
If Text6 = "" Then
MsgBox ("Введите ставку налога на прибыль")
Text8.Enabled = True
Text6.SetFocus
GoTo m2
End If
If Text8 = "" Then
MsgBox ("Введите ставку налога на прибыль в ФБ")
Text8.Enabled = True
Text8.SetFocus
GoTo m2
End If
m2: End Sub
```

```
Private Sub Option2_Click()
If Option3.Value = True Then
Option3.Value = False
End If
If Text12 = "" Then
MsgBox ("Введите имя файла")
Text12.SetFocus
End If
End Sub
```

```
Private Sub Option3_Click()
If Option2.Value = True Then
Option2.Value = False
End If
If Text12 = "" Then
MsgBox ("Введите имя файла")
Text12.SetFocus
End If
End Sub
```

```
Private Sub Text8_Change()
If Text9 = "" Then
Text9 = Text6 - Text8
End If
End Sub
```

**Коды для Modulel**

```
(General) (Declaration)
Type fil
d As String
t As String
k As String
s As String
sf As String
```

```
End Type
Dim w As fil
```

5. Протестировать приложение, моделируя различные действия пользователя, особенно обратить внимание на обработку некорректных ситуаций.

**Результат выполнения работы:** Вы ознакомились с технологией визуального программирования при создании приложений для графической операционной системы. Научились создавать дизайн экранной формы приложения, определять значения свойств объектов формы, связывать с ними программные коды обработки событий, инициируемых пользователем. Вы получили навыки тестирования приложений с акцентом на обработку возможных некорректных действий пользователя.

**Форма представления результатов работы**: папка с файлами проекта.

## Практическая работа № 6
## Технология объектно-ориентированного программирования

**Цель работы** – познакомиться на практике с основными концепциями объектно-ориентированного программирования и исследовать типовые операторы языка C++ (арифметические, логические, операторы управления ходом программы, операторы ввода-вывода), получить навыки работы со специфическими типами данных C++ (классы, объекты). Проверить области действия внутренних и внешних свойств и методов объектов.

### Этап 6.1. Исследование типовых операторов языка C++

1. Вам предстоит, действуя по шаблонам и используя разработанные ранее алгоритмы, создать простейшие программы обработки числовых данных и сохранить их в виде файлов `int_1.cpp`, `int_2.cpp` и т.д. для простых чисел, `arr_1.cpp`, `arr_2.cpp` и т.д. для массивов.
2. Переписать из листинга № 1 и отладить программу суммирования двух целых чисел.

   **Листинг 1.** Ввести два целых числа. Вычислить их сумму и вывести результат на экран.

```
#include <iostream.h>
#include<fstream.h>
void main (void)
{
int a,b,c;

   cout<<"Введите два целых числа" <<endl;
   cin>>a>>b;
   c=a+b;
   cout<<"Результат сложения: "<<c<<endl;
}
```

3. По аналогии с предыдущим заданием написать и отладить программы:
   - ввести два целых числа. Найти их разницу и результат вывести на экран;
   - ввести два целых числа. Минимальное из них удвоить и результат вывести на экран.

4. Придумать 2 аналогичные задачи, записать решение их на языке C++ и отладить программу.

5. Переписать из листинга № 2 и отладить программу обработки массива целых чисел.

**Листинг 2.** Ввести одномерный массив целых чисел. Преобразовать его, заменив положительные элементы 1, отрицательные – 0. Результат вывести на экран.

```
#include <iostream.h>;
void main()
{
      int A[4];
      int i;
      cout<<"Введите массив целых чисел" <<endl;
      for (i=0; i<=3; i++)             //ввод массива
            cin >> A[i];
      for (i=0; i<=3; i++)
            if (A[i]>0) A[i]=1;
            else
                  A[i]=0;
      cout<<"Результат преобразования массива" <<endl;
      for (i=0; i<=3; i++)             //вывод результата
            cout << A[i] << "\n";
}
```

6. Придумать 2 аналогичные задачи, записать алгоритм их решения на языке С++ и отладить программу.

### Этап 6.2. Исследование механизма работы с подпрограммами

1. Переписать из листинга № 3 текст программы, исследовать ее работу, объяснить результат.

**Листинг 3.**

```
#include <iostream.h>
void show(void)
{
cout<<"Функция show"<<endl;
}
void main(void)
{
cout<<"Вызов функции show"<<endl;
show();
cout<<"Возврат в основную программу"<<endl;
}
```

2. Переписать из листинга № 4 текст программы, исследовать ее работу, объяснить результат.

**Листинг 4.**

```
#include <iostream.h>
void max (float x, float y) //x, y - формальные параметры
{
```

```
if (x>y) cout<<x<<">"<<y<<endl;
else cout<<y<<">"<<x<<endl;
}
void main(void)
{
float a,b;
cin>>a>>b;
max(a,b); //a, b — фактические параметры
max (20,10); //20, 10 — фактические параметры-значения
max (2.71,3.14); //2.71, 3.14 - фактические параметры-значения
}
```

3. Переписать из листинга № 5 текст программы, исследовать ее работу, объяснить результат.

**Листинг 5.**

```
#include <iostream.h>
float max(float x,float y)
{
float result;
if (x>y) result=x;
else result=y;
return (result);
}
void main(void)
{
float a,b;
cin>>a>>b;
cout<<"Наибольшее из двух чисел" <<a<<" и
"<<b<<":"<<max(a,b)<<endl;
cout<<"Наибольшее из двух чисел 2.71 и 3.14: "<< max(2.71,
3.14)<<endl;
}
```

### Этап 6.3. Исследование сложных типов данных

1. Тип данных структура (`struct`) состоит из фиксированного числа элементов разных типов. Переписать из листинга № 6 текст программы, иллюстрирующий механизм доступа к элементам структуры, исследовать ее работу, объяснить результат. Придумать пример по аналогии.

**Листинг 6.** Работа с элементами структуры

```
#include<iostream.h>
#include<string.h>
struct date                 //описание структуры
{
int day;
char month[9] ;
int year;
}d;                         //объявление переменной d
void show_date(date d)      //функция вывода даты
{
cout<<"Число: "<<d.day<<endl;
```

```
cout<<"Месяц: "<<d.month<<endl;
cout<<"Год: "<<d.year<<endl;
}
void main(void)
{
d.day=2;
strcpy(d.month,"август");  //копировать название месяца в строку
d.year=1953;
show date(d);              //обращение к функции вывода даты
}
```

### Этап 6.4. Исследование операторов работы с файлом

1. Переписать из листинга № 7 программу, иллюстрирующую механизм создания текстового файла и записи в него текста, исследовать ее работу, объяснить результат. Придумать пример по аналогии.

**Листинг 7.** Создать файл и записать в него текст

```
#include<fstream.h>
void main(void)
{
ofstream out_file("filel.txt") ;           //открытие файла
out_file<<"Запись в файл. "<<endl;
out_file<<"Режим создания."<<endl;
out_file.close();                          //закрытие файла
};
```

2. Переписать из листинга № 8 программу, иллюстрирующую механизм добавления текста в существующий файл, исследовать ее работу, объяснить результат. Придумать пример по аналогии.

**Листинг 8.** Добавить текст в файл

```
#include<fstream.h>
void main(void)
{
ofstream out_file ("filel.txt",ios::app);
out_file<<"Режим добавления "<<endl;
out_file.close();
}
```

3. Переписать из листинга № 9 программу, иллюстрирующую механизм ввода трех первых слов из строки текста, записанного в файле, исследовать ее работу, объяснить результат. Придумать пример по аналогии.

**Листинг 9.** Ввод трех первых слов текста из файла

```
#include<iostream.h>
#include<fstream.h>
void main(void)
```

```
{
char a[64],b[64],c[64];
ifstream in_file("filel.txt");
in_file>>a>>b>>c;
cout«"Первая строка файла filel.txt: "<<a<<" "<<b<<" "<<c<<endl;
in_file.close();
}
```

4. Переписать из листинга № 10 программу, иллюстрирующую механизм ввода трех первых строк текста, записанного в файле, исследовать ее работу, объяснить результат. Придумать пример по аналогии.

**Листинг 10.** Ввод первых трех строк текста из файла

```
#include<iostream.h>
#include<fstream.h>
void main(void)
{
char a[64],b[64],c[64];
ifstream in_file("filel.txt");
in_file.getline(a,sizeof(a));
in_file,getline(b,sizeof(b));
in_file.getline(c,sizeof(c));
cout<<a<<endl;
cout<<b<<endl;
cout<<c<<endl;
in_file.close();
}
```

5. Переписать из листинга № 11 программу, иллюстрирующую механизм ввода текста, записанного в файле, исследовать ее работу, объяснить результат. Придумать пример по аналогии.

**Листинг 11.** Ввод текста из файла

```
#include<iostream.h>
#include<fstream.h>
void main(void)
{
char a[64];
if stream in_file("filel.txt");
while(in_file.eof())
   {
   in_file.getline(a,sizeof(a));
   cout<<a<<endl;
   }
in_file.close();
}
```

6. Переписать из листинга № 12 и 13 программы, иллюстрирующие механизмы записи массива в файл и чтение его из файла, исследовать их работу, объяснить результат. Придумать пример по аналогии.

**Листинг 12.** Запись массива в файл

```
#include<fstream.h>
void main(void)
{
int i,x[3]={10,20,30};
ofstream out_file("file2.dat");
for(i=0;i<3;i++)
out_file.write((char *) &x[i],sizeof(x [i]));
out_file.close();
}
```

**Листинг 13.** Чтение массива из файла с выводом на экран

```
#include<iostream.h>
#include<fstream.h>
void main(void)
{
int i, x[3];
ifstream in_file("file2.dat");
for(i=0;i<3;i++)
{
in_file.read((char *) &x[i] ,sizeof(x[i]));
cout<<x[i]<<" ";
}
in_file.close();
}
```

7. Переписать из листинга № 14 и 15 программы, иллюстрирующие механизмы записи структуры в файл и чтение ее из файла, исследовать их работу, объяснить результат. Придумать пример по аналогии.

**Листинг 14.** Запись структуры в файл

```
#include<fstream.h>
void main(void)
{
struct date
{
int day;
char month[9];
int year;
}d={25,"ноября", 1958};
ofstream out_file("file3.dat") ;
out_file.write((char *) &d,sizeof(date));
out_file.close();
}
```

**Листинг 15.** Чтение структуры из файла с выводом на экран

```
#include<iostream.h>
#include<fstream.h>
void main(void)
{
struct date
{
int day;
char month[9];
int year;
}d;
ifstream in_file("file3.dat");
```

```
in_file.read((char *) &d,sizeof(date));
cout<<d.day<<" "<<d.month<<" "<<d. year<<endl;
in_file.close();
}
```

### Этап 6.5. Исследование операций с объектами и классами

1. Переписать из листинга № 16 программу, иллюстрирующую создание и инициализацию объектов класса, исследовать ее работу, объяснить результат. Придумать пример по аналогии.

**Листинг 16.** Создать объекты класса, присвоить начальные значения их свойствам. Обеспечить к ним доступ извне.

```
#include<iostream.h>
#include<string.h>
class date
{
public: char name[64];
int day;
char month[9];
int year;
void show_birthday(void)
{
cout<<"Имя: "<<name<<endl;
cout<<"День рождения :"<<day<<" "<<month<<" "<<year<<endl;
};
};

void main(void)
{
date father,mother,daughte;      //объявление объектов
strcpy(father.name,"Юрий");
father.day=25;
strcpy(father.month,"ноября");
father.year=1958;
strcpy(mother.name,"Лидия");
mother.day=14;
strcpy(mother.month,"ноября");
mother.year=1956;
strcpy(daughte.name,"Юлия");
daughte.day=4;
strcpy(daughte.month,"мая");
daughte.year=1982;
father.show_birthday();          //вывод информации
mother.show_birthday();
daughte.show birthday();
}
```

2. Переписать из листинга № 17 программу, иллюстрирующую концепцию инкапсуляции данных как сокрытие части свойств путем объявления элементов класса частными (`private`), исследовать ее работу, объяснить результат. Придумать пример по аналогии.

**Листинг 17. Пример инкапсуляции**

```
#include<iostream.h>
#include<string.h>
class zp
{
private:                          //объявление частных элементов
char name[64];
float sal;
public:                           //объявление общих элементов
void in_dat(char *,float);        //прототипы функций
void show_sal(void);
float change_sal(float);

void zp::in_dat(char *new_name,float old_sal) //определение функции
{
strcpy(name,new name);
sal=old_sal;
}

void zp::show_sal(void)                //определение функции
{
cout<<"Оклад "<<name<<": "<<sal<<" рублей, "<<endl;
}

float zp::change_sal(float new_sal)   //определение функции
{
if(new_sal<10000)
      {
      cout<<"Новый оклад "<<name<<" недопустимо мал! "<<endl;
      return (0);
      }
sal=new_sal;
return(1);
}

void main(void)
{
float new_sal;
zp father;                                //объявление объекта father
father.in_dat("Иванов И.А.",20000.0) ;
cout<<"Старый ";
fatner.show_sal() ;
cout<<"Введите новый оклад:";
cin>>new_sal;
if(father.change_sal(new_sal)!=0)
      {
      Cout<<"Новый ";
      father.show sal();
      }
  }
```

Главной программе (main) известны лишь методы (in_dat, show_sal, change_sal), позволяющие работать с объектом (father). Свойства (name, sal) и способы их обработки скрыты от

главной программы внутри объектов.

**Результат выполнения работы**: Вы ознакомились с основами технологии объектно-ориентированного программирования и языка C++: типовыми действиями с простыми и сложными типами данных, операторами ввода-вывода. Вы получили представление об особенностях особого типа данных – класс как совокупность свойств и методов (функций), научились описывать объекты данного класса, инициализировать их свойства и использовать методы объектов для обеспечения доступа к свойствам объектов.

**Форма представления результатов работы**: папка с файлами.