**Санкт-Петербургский**
**государственный университет телекоммуникаций**
**им. проф. М. А. Бонч-Бруевича**

**С. В. Козин Н. А. Матиясевич**

# Методические указания к лабораторным работам по дисциплине Программирование на языке высокого уровня (электронная версия)

**Порядок выполнения лабораторных работ**

Все лабораторные работы должны выполняться в следующей последовательности.

1. Получение у преподавателя варианта домашнего задания.
2. Выполнение домашнего задания.
3. Проверка преподавателем домашнего задания.
4. Ввод программы в ЭВМ и ее отладка.
5. Решение задачи на ЭВМ.
6. Защита лабораторной работы.

В процессе выполнения домашнего задания студент должен написать “заготовку “ отчета по лабораторной работе. Отчет по лабораторной работе должен содержать следующие материалы.

- титульный лист с названием лабораторной работы и указанием группы и ФИО студента,
- задание на лабораторную работу (полная формулировка задачи),
- таблицу идентификаторов,
- схему алгоритма,
- таблицу вычислений.

Поясним назначение некоторых элементов отчета. Таблица идентификаторов предназначена для установления связи между обозначениями, принятыми в условии задачи и именами переменных, которые используются в программе. Эта таблица может служить некоторым комментарием к программе. Структура этой таблицы приведена ниже.

**Таблица идентификаторов**

| Номер | Обозначение в задаче | Идентификатор | Назначение |
|---|---|---|---|
| | | | |
| | | | |

Для получения зачета по лабораторной работе студент должен продемонстрировать ее работоспособность. С этой целью им в процессе выполнения домашнего задания должны быть разработаны тесты. Результаты этой разработки должны быть оформлены в виде таблицы вычислений. Формат таблицы вычислений приведен ниже.

Подготовка теста необходима для отладки программы. Тестирование – испытание программы в целях выявления в ней возможных ошибок. Тест состоит из контрольного набора данных и рассчитанных вручную ожидаемых выходных данных.

Совпадение вычисленного вручную результата с результатом, полученным на ЭВМ, дает основание полагать, что программа может считаться работоспособной. Следует особо отметить, что успешное

завершение тестирования не является гарантией того, что в программе нет ошибок.

Целесообразно для теста выбирать такой набор исходных данных, при использовании которого удается упростить вычисления.

Например, для проверки вычислений по формуле

$$y = \frac{a\sin(x)}{\sqrt{a+3}}$$

целесообразно выбрать следующий контрольный набор исходных данных: $a = 13$ и $x = 0{,}5236(\pi/6)$. При этом легко вычисляются значения функций $sin(\pi/6) = 0{,}5$ и $\sqrt{13+3} = 4$. Ожидаемый результат вычислений $y = 1{,}625$.

Замечания:

- не рекомендуется в качестве контрольных исходных данных выбирать величины, приводящие к появлению сомножителей, равных 0 или 1, и слагаемых, равных нулю,
- контрольный расчет должен выполняться с точностью до четырех (пяти) значащих цифр,
- в ряде случаев для выполнения контрольного расчета приходится использовать калькулятор.

Результаты контрольного расчета и вычислений, полученных на ЭВМ, оформляются в виде таблицы (таблицы вычислений), которая имеет следующий вид.

Таблица вычислений

| Назначение набора данных | Набор данных | | | | Результаты вычислений | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | ручных | | машинных | |
| | | | | | | | | |
| Контрольный набор данных (тест) | | | | | | | | |
| Рабочий набор данных | | | | | | | | |

***Лабораторная работа 1***

# Линейные вычислительные процессы

Лабораторная работа должна выполняться в соответствии с указаниями, приведенными в разделе Порядок выполнения лабораторных работ.

### Цель работы

Целью настоящей работы является получение студентами практических навыков в решении на ЭВМ задач, связанных с вычислением значений по заданным формулам.

### Варианты заданий

Необходимо решить задачу вычисления и вывода на экран значений функций $y = f_1(x)$ и $z = f_2(y,a,b)$. Варианты заданий, а также рабочий набор данных приведены в таблице.

| N | Функция $y = f_1(x)$ | Функция $z = f_2(y,a,b)$ | Рабочий набор | | |
|---|---|---|---|---|---|
| | | | x | a | b |
| 1 | $\frac{\sqrt{x^2+16}}{x+2}$ | $\frac{y+\sqrt{\sin a+3}+b}{y^2+\sqrt{\sin a+3}}$ | 3,5 | 1,8 | 3,7 |
| 2 | $\frac{e^{x-2,7}+3}{x+1,3}$ | $\frac{y+0,75\lvert\cos b\rvert+a}{y^2+0,75\lvert\cos b\rvert}$ | 8,2 | 2,2 | 8,2 |
| 3 | $\frac{\sin x+1,5}{2}$ | $\frac{y^3+\sqrt{\sqrt{a}+3,3}}{b+\sqrt{\sqrt{a}+3,3}}$ | 8,1 | 0,8 | 1,2 |
| 4 | $\frac{\ln(x-3)+4}{x^2+12}$ | $\frac{\sqrt[3]{y+7}+a}{\sin b+\sqrt[3]{y+7}}$ | 4,7 | 7,6 | 8,1 |
| 5 | $\frac{\lvert x\rvert+8}{x^3+18}$ | $\frac{\sqrt[4]{y+15}+a}{\cos b+\sqrt[4]{y+15}}$ | 3,4 | 82 | 2,5 |
| 6 | $\frac{\cos^2(x)+2}{3}$ | $\frac{y\sqrt[5]{a}+1}{\sin b+y\sqrt[5]{a}}$ | -8 | 8,7 | 1,3 |
| 7 | $\frac{e^{x+3,1}+2}{x+6,1}$ | $\frac{\sqrt[3]{a}+2y+tgb+3}{tgb+2y+3}$ | 2,5 | 8,7 | 1,8 |
| 8 | $\frac{\sqrt{e^{x-2}+3}}{x}$ | $\frac{\sqrt[4]{a}+\sqrt{5y+20}}{\sqrt{5y+20}+b}$ | 2,7 | 17 | 11 |

**Продолжение таблицы**

| N | Функция $y = f_1(x)$ | Функция $z = f_2(y,a,b)$ | Рабочий набор | | |
|---|---|---|---|---|---|
| | | | x | a | b |
| 9 | $\frac{tgx+3{,}73}{4}$ | $\frac{7y+3\sin a+\sqrt{b^2+19}}{7y+\sqrt{b^2+19}+2}$ | 0,1 | 1,5 | 10 |
| 10 | $\frac{\sin^3(x)+3{,}7}{5}$ | $\frac{\sqrt{14y+2}+6}{\sqrt{14y+2}+\cos b+a}$ | 2,5 | 5 | 6,1 |
| 11 | $\frac{\sqrt{x+12}}{2x^3+1}$ | $\frac{\lvert y^2-a\rvert+6}{2\cos b+\lvert y^2-a\rvert+6}$ | 18 | -3 | 8,1 |
| 12 | $\frac{\sqrt[3]{x+8{,}3}}{x+0{,}3}$ | $\frac{4+y^2+\lvert\sin x\rvert+a}{\lvert\sin x\rvert+y^2+0{,}2b}$ | 3,7 | -2 | 8,1 |
| 13 | $\frac{1+\ln(x+5{,}3)}{x+5{,}3}$ | $\frac{\sqrt{y+15\sin a}}{\sqrt{y+15\sin a}+2b}$ | 2 | 2 | 3 |
| 14 | $\frac{e^{x-1{,}5}+2}{2x+0{,}3}$ | $\frac{\sqrt[4]{27y+54}+a}{\sqrt[4]{27y+54}+\lvert\cos b\rvert+1}$ | 4,1 | 9 | 3,5 |
| 15 | $\frac{\lvert\sin x\rvert+2}{3}$ | $\frac{\sqrt[3]{y+7a}+b}{\sin b+1+\sqrt[3]{y+7a}}$ | 2,5 | 1,3 | 3,3 |
| 16 | $\frac{2}{\sqrt{\cos x+5}}$ | $\frac{\sqrt[3]{y+13a}+5}{\cos b+\sqrt[3]{y+13a}}$ | 6,1 | 2,3 | 2,6 |
| 17 | $\frac{\sqrt{\lvert\cos x+3\rvert}}{3}$ | $\frac{\sin b+\sqrt[4]{y+15a}}{\sqrt[4]{y+15a}}$ | 8 | 1,3 | 2,5 |
| 18 | $\frac{\sqrt[3]{x-3{,}1}}{x-27}$ | $\frac{(y+1)^2+5a}{\sin b+(y+1)^2+5a}$ | 80 | 0,8 | -2 |
| 19 | $\frac{3e^{x-2}}{x+1}$ | $\frac{a\sqrt[3]{y+2b}}{2-\cos b+\sqrt[3]{y+2b}}$ | 6,1 | 8 | 9,2 |
| 20 | $\frac{\sin^2(x)+5}{5}$ | $\frac{\sqrt[3]{ay+57}}{3+\lvert\cos b\rvert+\sqrt[3]{ay+57}}$ | -2 | 7,3 | 5,1 |
| 21 | $\frac{x-7}{\ln(x-2)+2}$ | $\frac{\sqrt[4]{ay^2+3}+2}{\sqrt[4]{ay^2+3}+b}$ | 10 | 23 | 1,1 |
| 22 | $\frac{\sqrt{\cos^2 x+10}}{5}$ | $\frac{\lg(y^2+8)+5\sin a}{\lg(y^2+8)+\lvert\cos b\rvert}$ | 5,2 | 2,5 | 7,2 |
| 23 | $\frac{\lg(17-2x)+2}{x+1}$ | $\frac{\lvert\cos a\rvert(y+3)}{\lvert\cos a\rvert(y+3)-b}$ | 0,6 | 5 | 2,1 |

**Окончание таблицы**

| N | Функция $y = f_1(x)$ | Функция $z = f_2(y,a,b)$ | Рабочий набор | | |
|---|---|---|---|---|---|
| | | | x | a | b |
| 24 | $\frac{e^{2x-7,4}+6}{x+4,3}$ | $\frac{\lvert\sin a\rvert(y+7)}{\lvert\sin a\rvert(y+7)+2b}$ | 5 | -2 | 0,7 |
| 25 | $\frac{3\sin x+21}{\cos 2x+25}$ | $\frac{\lg(y^2+99)+a^2}{\lg(y^2+99)+b^2}$ | 3,5 | 14 | 7 |
| 26 | $\frac{\sqrt[3]{\lvert\sin x\rvert+26}}{\sin(2x)+5}$ | $\frac{\lg(17y^2+83)+a^2}{\lg(17y^2+83)+b}$ | 2 | 8 | 4,3 |
| 27 | $\frac{\sqrt[3]{\lvert\cos x\rvert+8}}{\cos 2x+5}$ | $\frac{\cos^2(ay)+5b}{\cos^2(ay)+b}$ | -2 | 8,1 | 13 |
| 28 | $\frac{\lvert\lg x\rvert+5}{x+4}$ | $\frac{\sin^2(a(2y^2+1))+29b}{\sin^2(a(2y^2+1))+b}$ | 0,2 | 2 | 12 |
| 29 | $\frac{14\lvert\lg x\rvert+2}{40+x}$ | $\frac{(\cos a+11y^2)^2}{(\cos a+11y^2)+b}$ | 0,2 | 7,1 | 9 |
| 30 | $\frac{4\sin^2 x+3}{2}$ | $\frac{\sin^2(a(3y^2-\frac{1}{3}))+11,75}{\sin^2(a(3y^2-\frac{1}{3}))+b}$ | 2,2 | 3,2 | 6,8 |

## Методические указания по выполнению лабораторной работы

Приведем ряд общих правил, которые следует учитывать при написании программ на языке Си.

1. Все действующие в программе переменные должны быть определены.
2. Любая программа должна содержать следующие три составные части:
   - ввод исходных данных,
   - обработка,
   - вывод результатов.
3. Недопустимо задавать исходные данные с помощью операторов присваивания. В связи с этим следует предусмотреть ввод всех данных, входящих в рабочий набор.
4. Для правильной компиляции вызовов библиотечных функций следует подключить те заголовочные файлы, в которых эти функции объявлены. Например, для компиляции вызовов функций printf() и scanf() следует подключить заголовочный файл stdio.h, а для компиляции функций clrscr() и getch() необходимо подключить файл

conio.h. Для компиляции вызовов математических функций следует подключить заголовочный файл math.h.

5. Ввод данных с клавиатуры следует предварить выводом наводящего сообщения. Например, пусть переменная x имеет тип float, тогда ее ввод может быть организован следующим образом::
   printf(‘x=’); /* Вывод наводящего сообщения */
   scanf(“%f”, &x) ; /* Ввод значения переменной x */

При вычислении по формулам часто используется прием, который называют вычленением одинаковых подвыражений. Например, для 30 варианта в формуле, определяющей значение величины *z*, дважды входит подвыражение $\sin^2(a(3y^2-2))$. Выполняя вычленение одинаковых подвыражений в задаче варианта 30, исходную расчетную формулу для вычисления величины *z* можно заменить следующими двумя формулами:

$$p = \sin^2(a(3y^2 - \frac{1}{3})),$$

$$z = \frac{p + 11{,}75}{p + b}.$$

Введение дополнительной переменной *p* позволяет уменьшить количество вычислений и упрощает расчетную формулу для вычисления величины *z*.

При записи арифметических выражений на языке Си необходимо учитывать следующее:

1. В языке Си отсутствует оператор возведения в степень. Для возведения некоторой величины “a” в степень “b” необходимо использовать библиотечную функцию pow(). Вызов этой функции для рассматриваемого случая будет иметь следующий вид pow(a, b).

2. Эта же (функция pow()) может использоваться для вычисления корней. Например, вычислить значение корня кубического из величины “a” можно следующим образом: pow(a, 1.0 / 3.0).

**Справочные материалы**

В настоящем разделе приводятся некоторые сведения о библиотечных функциях языка Си. Эти сведения приводятся в таблице

| Имя функции | Прототип | Описание |
|---|---|---|
| `abs` | `int abs(int num);` | Вычисление модуля аргумента `num` |
| `ceil` | `double ceil(double num);` | Возвращает наименьшее целое, которое удовлетворяет условию >= num. Обратите внимание на тип возвращаемого значения (**`double`**). |
| `cos` | `double cos(double num);` | Вычисляет значение косинуса от аргумента num. Значение аргумента должно быть задано в радианах. |
| `fabs` | `double fabs(double num);` | Вычисление значение модуля аргумента num |
| `floor` | `double floor(double num);` | Возвращает наибольшее целое, которое удовлетворяет условию <= `num`. |
| `exp` | `double exp(double num)` | Вычисляет значение экспоненты от аргумента num |
| `log` | `double log(double num);` | Вычисляет значение натурального логарифма от аргумента num. |
| `log10` | `double log10(double num);` | Вычисляет значение логарифма по основанию 10 от аргумента `num`. |
| `pow` | `double pow(double base, double x);` | Вычисляет значение аргумента `base`, возведенное в степень `x`. |
| `sin` | `double sin(double num);` | Вычисляет значение синуса от аргумента `num`. Значение аргумента должно быть задано в радианах. |
| `sqrt` | `double sqrt(double num);` | Вычисляет значение корня квадратного от аргумента `num`. |
| `tan` | `double tan(double num);` | Вычисляет значение тангенса от аргумента `num`. Значение аргумента должно быть задано в радианах. |

Замечание. В стандартной библиотеке языка Си имеются несколько функций для вычисления модуля (abs, fabs). Применение функции abs() для данных вещественных типов может привести к потере точности. Избежать этого можно при применении функции fabs().

## Пример оформления отчета по лабораторной работе (для варианта 30)

### А. Постановка задачи

Требуется составить программу вычисления значений функций

$$y = \frac{4\sin^2 x + 3}{2},$$

$$z = \frac{\sin^2(a(3y^2 - \frac{1}{3})) + 11.75}{\sin^2(a(3y^2 - \frac{1}{3})) + b}$$

по заданным значениям *a*, *x*, *b* и выполнить вычисления на ЭВМ.

### Б. Разработка алгоритма

Решаемая задача относится к категории задач формульного счета. В формуле для вычисления величины *z* целесообразно выполнить вычленение одинаковых подвыражений. Для выполнения вычленения введем дополнительную переменную *p*. С учетом этого расчетные формулы принимают следующий вид:

$$y = \frac{4\sin^2 x + 3}{2},$$

$$p = \sin^2(a(3y^2 - \frac{1}{3})),$$

$$z = \frac{p + 11{,}75}{p + b}.$$

В программе должен быть предусмотрен ввод исходных данных, к которым относятся переменные *x*, *a*, *b*; вычисления величин *y*, *p* и *z*; вывод результатов вычислений (вывод значений величин *y* и *z*).

### В. Таблица идентификаторов

| N | Обозначение в задаче | Идентификатор | Назначение |
|---|---|---|---|
| 1 | *X* | *x* | Исходные данные |
| 2 | *A* | *a* | |
| 3 | *B* | *b* | |
| 4 | *Y* | *y* | Результаты вычислений |
| 5 | *z* | *z* | |
| 6 | - | *p* | Промежуточная величина |

*Лабораторная работа 2*

# Разветвляющиеся вычислительные процессы

Лабораторная работа должна выполняться в соответствии с указаниями, приведенными в разделе “Порядок выполнения лабораторных работ”.

## Цель работы

Целью настоящей работы является получение практических навыков в решении задач, в которых выбор расчетной формулы определяется некоторыми условиями.

## Варианты заданий

Необходимо решить на компьютере задачу вычисления значения функции y = f(x). Варианты заданий, а также рабочие наборы исходных данных приведены в таблице

| Номер варианта | Функция y = f(x) | Рабочий набор данных |
|---|---|---|
| | | X |
| 1 | $\begin{cases} \lg(x) & \text{при } 4 <= x < 6, \\ \sin^2(x) & \text{при } x >= 6, \\ e^{0.1x} & \text{при } -3 <= x < 4, \\ \dfrac{\cos(x)}{x+10} & \text{при } x < -3 \end{cases}$ | 7,4 |
| 2 | $\begin{cases} e^{\sqrt{x}} & \text{при } x > 2, \\ \cos^2(x) & \text{при } 2 >= x > 1, \\ \lg(20x) & \text{при } 1 >= x > 0.5, \\ e^{0.2x} & \text{при } x <= 0.5 \end{cases}$ | 4,3 |
| 3 | $\begin{cases} \dfrac{1}{x} & \text{при } x > 4, \\ e^{-x} & \text{при } 2 < x <= 4, \\ x^2 & \text{при } -1 < x <= 2, \\ \dfrac{\lvert x-1 \rvert}{2x} & \text{при } x <= -1 \end{cases}$ | 10,9 |

| Номер варианта | Функция y = f(x) | Рабочий набор данных |
|---|---|---|
| | | X |
| 4 | $\begin{cases} \frac{2}{x} & \text{при } x < -5, \\ x\sin x & \text{при } 2 > x >= -5, \\ \frac{x+10}{2+x^2} & \text{при } 3 > x >= 2, \\ x + \sqrt[3]{x} & \text{при } x >= 3 \end{cases}$ | -10 |
| 5 | $\begin{cases} \sqrt{x} & \text{при } x > 7, \\ e^{-x} & \text{при } 2 < x <= 7, \\ e^{x} & \text{при } -1 < x <= 2, \\ \sin(x) & \text{при } x <= -1 \end{cases}$ | 9,2 |
| 6 | $\begin{cases} 3x^3 & \text{при } x > 20, \\ \ln(x+3) & \text{при } 20 >= x > 10, \\ e^{-x^2} & \text{при } 10 >= x > 3, \\ \lvert\sin(x)\rvert & \text{при } 3 >= x \end{cases}$ | 25 |
| 7 | $\begin{cases} \frac{\sin^2(x)}{2} & \text{при } x > 20, \\ \sqrt[3]{x} & \text{при } 20 >= x >= 6, \\ \sin^2(x) & \text{при } 6 >= x > -4, \\ 0 & \text{при } x <= -4 \end{cases}$ | -19,8 |
| 8 | $\begin{cases} \sin(\sqrt{x}) & \text{при } x > 30, \\ \sqrt{x} & \text{при } 30 >= x > 10, \\ \lg(0.5*x) & \text{при } 10 >= x > 2, \\ \lvert\sin(x)\rvert & \text{при } x <= 2 \end{cases}$ | 3,1415 |
| 9 | $\begin{cases} 2\sqrt{(x^2+15)} & \text{при } x < -6, \\ 4\cos(x) & \text{при } -6 <= x < 2, \\ \frac{\sin(x-3)}{2} & \text{при } 2 <= x < 10, \\ \frac{tgx}{10} & \text{при } x >= 10 \end{cases}$ | 0 |
| 10 | $\begin{cases} 1+x^2 & \text{при } x > 25, \\ 2+x^2 & \text{при } 25 >= x > 8, \\ 3+x^2 & \text{при } 8 >= x > 2, \\ 4+x^2 & \text{при } x <= 2 \end{cases}$ | 100 |

| Номер варианта | Функция y = f(x) | Рабочий набор данных |
|---|---|---|
| | | X |
| 11 | $\begin{cases} x\lvert x+21\rvert & \text{при } x<-14, \\ x^2\ln\lvert x^2+48\rvert & \text{при } -14<=x<-5, \\ \frac{x}{3}+\sqrt{(x^2+16)} & \text{при } -5<=x<0, \\ 2+\frac{x}{3} & \text{при } x>=0 \end{cases}$ | -15,5 |
| 12 | $\begin{cases} x+(\lvert x\rvert-12)^2 & \text{при } x<-8, \\ \frac{\sin(x)+3}{\cos(x)+15} & \text{при } -8<=x<7, \\ x\sqrt{(x^2-36)} & \text{при } 7<=x<10, \\ x+8 & \text{при } x>=10 \end{cases}$ | 7,6 |
| 13 | $\begin{cases} \frac{\cos(x)+14}{\sin(x)+15} & \text{при } x<=4, \\ \sqrt[3]{x+\ln(\lvert x-8\rvert)} & \text{при } 4<x<12, \\ \sqrt{x-13} & \text{при } 12<=x<38, \\ 5x & \text{при } x>=38 \end{cases}$ | 40 |
| 14 | $\begin{cases} \sqrt{\lvert x\rvert}+\frac{\sin(x)}{2} & \textit{при } x<-12, \\ \sin(x-5) & \textit{при } -12<=x<8, \\ \frac{\sqrt{(x+6)}}{x+3} & \textit{при } 8<=x<10, \\ \frac{4}{3+x} & \textit{при } x>=10 \end{cases}$ | 12 |
| 15 | $\begin{cases} \frac{x\sin(x)+5}{2+\sin(x)} & \text{при } x<5, \\ \sqrt{\lvert x-7\rvert}+\frac{x}{2} & \text{при } 5<=x<10, \\ \lg(x) & \text{при } 10<=x<111, \\ 2x & \text{при } x>=111 \end{cases}$ | 1,5 |
| 16 | $\begin{cases} \sqrt{(\lvert x\rvert-1)} & \text{при } x<=-20, \\ \frac{x+\sin(x)}{2} & \text{при } -20<x<6, \\ x^2+\ln(x-2) & \text{при } 6<=x<12, \\ x^2+\ln(10) & \text{при } x>=12 \end{cases}$ | 3 |

| Номер варианта | Функция y = f(x) | Рабочий набор данных |
|---|---|---|
| | | X |
| 17 | $\begin{cases} \lg(1.5x) & \text{при } 4 <= x < 7, \\ \sin^4(x) & \text{при } x >= 7, \\ 0.5e^{0.1x} & \text{при } -3.5 <= x < 4, \\ \dfrac{\cos(x)}{\lvert x \rvert + 10} & \text{при } x < -3.5 \end{cases}$ | 2 |
| 18 | $\begin{cases} \dfrac{\sin(x)+2}{\cos(x)+10} & \text{при } x <= 5, \\ \sqrt[3]{(2+\ln(\lvert x-8 \rvert+2))} & \text{при } 5 < x < 15, \\ \sqrt{x-13} & \text{при } 15 <= x < 38, \\ 5+x & \text{при } x >= 38 \end{cases}$ | 3,1415 |
| 19 | $\begin{cases} \dfrac{x}{2} & \text{при } x > 5, \\ 10e^{-x} & \text{при } 2 < x <= 5, \\ \dfrac{3+x^2}{2x} & \text{при } -2 < x <= 2, \\ \dfrac{\lvert x-100 \rvert}{2x} & \text{при } x <= -2 \end{cases}$ | 23 |
| 20 | $\begin{cases} \sqrt{x} & \text{при } x > 9, \\ 5e^{-x} & \text{при } 2 < x <= 9, \\ e^{x} & \text{при } 1 < x <= 2, \\ \cos(x) & \text{при } x <= 1 \end{cases}$ | 2 |
| 21 | $\begin{cases} x^3 & \text{при } x > 10, \\ 2x & \text{при } 10 >= x > 3, \\ 0,5x & \text{при } 3 >= x > -5, \\ \sqrt[3]{\lvert x+2 \rvert} & \text{при } x <= -5 \end{cases}$ | 5,7 |
| 22 | $\begin{cases} x^2 & \text{при } x > 10, \\ \lg(x) & \text{при } 10 >= x > 1, \\ \sqrt{\lvert x-15 \rvert} & \text{при } 1 >= x >= -1, \\ 4 & \text{при } x < -1 \end{cases}$ | 20,5 |
| 23 | $\begin{cases} e^{-x} & \text{при } x <= 0, \\ \sqrt[3]{x+1} & \text{при } 0 < x <= 7, \\ \dfrac{x+13}{10} & \text{при } 7 < x <= 87, \\ 10(x-86) & \text{при } x > 87 \end{cases}$ | -2,5 |

| Номер варианта | Функция y = f(x) | Рабочий набор данных |
|---|---|---|
| | | X |
| 24 | $\begin{cases} \frac{x+75}{\sqrt{x}+5} & \text{при } x >= 25, \\ 0{,}4x & \text{при } 25 > x >= 5, \\ \sqrt{x-1} & \text{при } 5 > x >= 2, \\ e^{x-2} & \text{при } x < 2 \end{cases}$ | -3 |
| 25 | $\begin{cases} \sqrt{x-1} & \text{при } x >= 5, \\ \frac{x+15}{x+5} & \text{при } 5 > x >= 0, \\ 3\cos x & \text{при } 0 > x >= -2\pi, \\ 3 & \text{при } x < -2\pi \end{cases}$ | 2,4 |
| 26 | $\begin{cases} \sqrt{x^2+16} & \text{при } x >= 3, \\ x & \text{при } 3 > x >= 1, \\ x^2+1 & \text{при } 1 > x >= 0, \\ e^{-x} & \text{при } x < 0 \end{cases}$ | 11,5 |
| 27 | $\begin{cases} 11 & \text{при } x > 5, \\ 2x+1 & \text{при } 5 >= x > 2, \\ x^2+1 & \text{при } 2 >= x > 0, \\ e^{-x} & \text{при } x <= 0 \end{cases}$ | 3,4 |
| | | X |
| 28 | $\begin{cases} 0 & \text{при } x >= \pi \\ \sin x & \text{при } \pi > x >= 0, \\ 0 & \text{при } 0 > x >= -2, \\ (x+2)^2 & \text{при } x < -2 \end{cases}$ | 6,9 |
| 29 | $\begin{cases} 4+(x-2)^2 & \text{при } x > 2, \\ 2^x & \text{при } 2 >= x > 1, \\ x+1 & \text{при } 1 >= x > 0, \\ e^x & \text{при } x <= 0 \end{cases}$ | 4,5 |
| 30 | $\begin{cases} 27+(x-3)^3 & \text{при } x > 3, \\ x^3 & \text{при } 3 >= x > 1, \\ x & \text{при } 1 >= x > 0, \\ \frac{\sin^2 x}{2} & \text{при } x <= 0 \end{cases}$ | 13 |

### Методические указания по выполнению работы

В данной лабораторной работе необходимо вычислить значение функции, заданной различными формулами на разных участках ее определения. Возможны два стандартных подхода к решению таких задач. В первом из этих подходов используются вложенные инструкции **if else**, а во втором инструкции **if** (сокращенный **if**).

Рассмотрим возможные способы организации разветвления для варианта 30.

### Метод 1. Использование вложенных инструкций if else

Этот метод может быть назван методом проверки точек ветвления. Запись алгоритма определения значения функции $y = f(x)$ для рассматриваемого варианта с использованием метода проверки точек ветвления будет иметь следующий вид.

```
/* Объявления переменных x и y и  ввод исходных данных  */
if( x > 3)
                    y = 27 + pow(x -3, 3);
else if( x > 1)
                    y = pow(x, 3);
     else if( x > 0)
                    y = x;
          else
                    y = pow(sin(x), 2) / 2;

/* Вывод значения переменной “y” */
```

Отметим, что при использовании вложенных инструкций **if else** целесообразно придерживаться следующих правил:

- новая инструкция **if then else** должна располагаться в false – ветви; запись вложенного **if** в true – ветви ухудшает читабельность программы;
- внутренние инструкции следует записывать с новой строки (в рассматриваемом примере это инструкции присваивания);
- ключевое слово **else** следует располагать под тем зарезервированным словом **if**, к которому оно относится.

Кроме того, при использовании рассматриваемого метода следует придерживаться следующего порядка:

- проверку точек ветвления целесообразно начать с одной из крайних точек (слева или справа),
- следует учитывать, что каждая из точек ветвления задается двумя отношениями; из этих отношений в инструкции **if else** следует выбирать то, для которого по направлению, соответствующему значению **true** проверяемого условия можно выделить расчетную формулу.

Например, в варианте 30 начать проверки можно либо с точки $x = 3$, либо с точки $x = 0$(но не с точки $x = 1$). Пусть в качестве первой точки для проверки выбрана точка $x = 3$. Эта точка задается следующими двумя отношениями: $x > 3$ и $x <= 3$. Для проверки в операторе *if else* следует выбрать отношение $x > 3$. При выборе этого отношения в ветви then можно выполнить вычисления по формуле $y = 27 + (x - 3)^3$.

Общее количество инструкций **if** при использовании рассматриваемого метода не превосходит количества точек ветвления. В данном примере таких точек три ($x = 3$, $x = 1$ и $x = 0$).

**Метод 2. Использование сокращенной формы инструкции *if***

Этот метод может быть назван методом проверки ветвей. Существо метода сводится к следующему. Для каждой области задания функции записывается логическое выражение, принимающее значение *true* в том случае, когда аргумент *x* попадает в эту область и значение *false* – в противном случае.

Запись алгоритма определения значения функции $y = f(x)$ для рассматриваемого варианта с использованием метода проверки ветвей будет иметь следующий вид.

```
/* Объявления переменных “x” и “y” и  ввод исходных данных */

if(x > 3)
                    y = 27 + pow(x -3, 3);
if(x <= 3 && x > 1)
                    y = pow(x, 3)         ;
if(x <= 1 && x > 0)
                    y = x                 ;
if(x >= 0)
                    y = pow(sin(x),2) / 2 ;

/* Вывод значения переменной “y” */
```

Здесь так же, как и в методе проверки точек ветвления, рекомендуется внутренние инструкции записывать в отдельной строке. Общее количество инструкций **if** определяется количеством ветвей разветвления. В нашем примере таких ветвей четыре.

При выборе метода организации разветвления следует учитывать следующее. Достоинством первого метода является меньшее среднее время выполнения. Действительно, после выполнения выбранного оператора присваивания дальнейшие проверки во вложенных инструкциях **if** не производятся. Во втором способе всегда выполняются все проверки в инструкциях **if**. Достоинством второго способа является более высокая степень читабельности.

Студент должен выбрать один из возможных способов реализации разветвления, обосновать сделанный выбор.

Следует отдельно остановиться на подготовке теста для данной лабораторной работы. Прежде всего, следует определить количество необходимых тестов. В связи с тем, что основным назначением теста в данной работе является проверка организации разветвления, общее количество тестов должно определяться количеством расчетных формул. При этом следует учитывать, что с помощью одного теста можно проверить только одну ветвь разветвления.

## Пример отчета по лабораторной работе

**(для варианта 30)**

А. Постановка задачи

Требуется составить программу вычисления значения следующей функции

$$y = \begin{cases} 27 + (x-3)^3 & \text{при} \quad x > 3, \\ x^3 & \text{при} \quad 3 >= x > 1, \\ x & \text{при} \quad 1 >= x > 0, \\ \dfrac{\sin^2(x)}{2} & \text{при} \quad x <= 0. \end{cases}$$

для заданного значения аргумента “x” и выполнить вычисления на компьютере.

Б. Таблица идентификаторов

| Обозначение в задаче | Идентификатор | Назначение |
|---|---|---|
| *X* | *X* | Аргумент функции |
| *Y* | *Y* | *Значение функции* |

В. Разработка алгоритма

Здесь студент должен сравнить два возможных способа решения поставленной задачи (см. п.2.3 настоящих указаний). Мы выберем способ, основанный на использовании вложенных инструкций **if else**. Его достоинством является меньшее среднее время выполнения по сравнению со способом, в котором используется сокращенная форма инструкции **if.**

## Г. Схема алгоритма

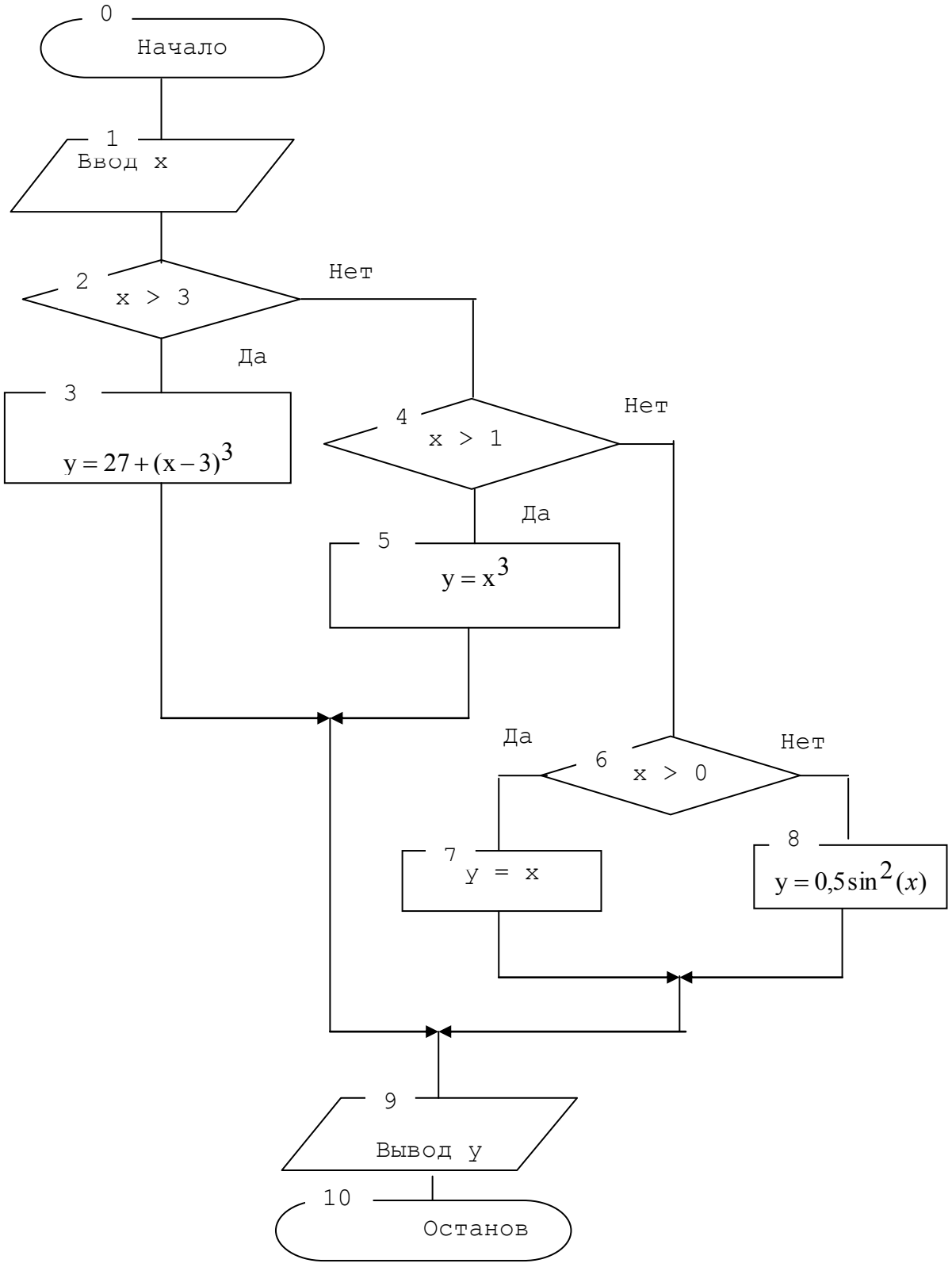



Рис.1.2.1 Схема алгоритма для варианта 30

## Д. Контрольный расчет

Для тестирования необходимо подготовить четыре теста. Количество тестов определяется числом расчетных формул, с помощью которых задана функция в варианте 30.

Для проверки первой формулы (ветви) функции $y = f(x)$ выбираем контрольный набор данных: $x = 5$, а для проверки второй, третьей и четвертой ветвей выберем соответственно $x = 2$, $x = 0{,}8$ и $x = -3.1415$.

Результаты вычислений соответствующих значений функции $y = f(x)$ приведены ниже в таблице вычислений.

| Назначение набора данных | Набор данных | Результаты вычислений | |
|---|---|---|---|
| | | ручных | Машинных |
| | *x* | *y* | *Y* |
| Контрольный | 5 | 31,00 | |
| | 2 | 8,000 | |
| | 0.8 | 0,800 | |
| | -3,1415 | 0,500 | |
| **Рабочий** | 13 | - | |

## Е. Программа на языке Си

```
/*                                              Файл  Lab2.c

                        Лабораторная работа 2

              ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ ПРОЦЕССОВ
                   Студент гр. СП – 91 Петров П. П.
*/

#include<stdio.h>
#include<conio.h>
int main(void)
{
    float x, y;
    printf("x=");
    scanf("%f", &x);

    if(x > 3)
                    y = 27 * pow(x, 3);
    else if(x > 1)
                    y = pow(x, 3);
         else if(x > 0)
                    y = x;
              else
                    y = pow(sin(x), 2) / 2;

    printf("y=%6.2f\n", y);
    getch();
    return 0;
}
```

### Контрольные вопросы

1. Какие стандартные управляющие структуры используются в структурном программировании для реализации разветвляющихся алгоритмов?
2. Какие управляющие структуры используются в языке Си для организации разветвляющихся алгоритмов?
3. Поясните порядок выполнения инструкции **if else** и ее сокращенной формы?
4. Поясните существо первого метода решения рассматриваемых в настоящей лабораторной работе задач?
5. Поясните существо второго метода решения рассматриваемых в настоящей лабораторной работе задач?
6. Сравните возможные методы решения рассматриваемых в настоящей лабораторной работе задач?
7. Сколько внутренних инструкций можно написать в каждой из ветвей инструкции **if else**?
8. Как разрешается неоднозначность, которая может возникнуть при использовании вложенных инструкций **if else**?

## *Лабораторная работа 3*

# Циклические вычислительные процессы. Задача табулирования

Лабораторная работа должна выполняться в соответствии с указаниями, приведенными в разделе “Порядок выполнения лабораторных работ”.

### Цель работы

Целью настоящей работы является получение практических навыков решения на компьютере задач по вычислению значений функции при различных значениях аргумента (табулирование функции).

### Постановка задачи

Необходимо решить на компьютере задачу вычисления $N$ значений функции $y = f(x)$ для ряда равноотстоящих значений аргумента $x$, начиная от значения $x = x_{нач}$ вплоть до значения $x = x_{кон}$. Функция $y = f(x)$ зависит от параметра $a$. Результаты вычислений следует оформить в виде таблицы, снабженной заголовком.

**Варианты заданий**

Вид функции $y = f(x)$ и рабочий набор исходных данных приведены в таблице

| $N$ | Вид функции $y = f(x)$ | Рабочий набор исходных данных | | | |
|---|---|---|---|---|---|
| | | $N$ | $a$ | $x_{нач}$ | $x_{кон}$ |
| 1 | $\frac{e^{-x} + e^{\sqrt{a}}}{e^{x-a}}$ | 15 | 1 | 0,2 | 0,5 |
| 2 | $\frac{\cos^4 x + \cos^2 a}{\cos ax + 1,5}$ | 10 | 0,5 | -1,3 | 1 |
| 3 | $\frac{tgax - x^2}{2 + a^2}$ | 12 | 2 | 0,3 | 0,35 |
| 4 | $\frac{\sqrt{a+2,7}}{\sqrt[3]{a+x^3}}$ | 16 | 1 | 2 | 3 |
| 5 | $\frac{\ln\lvert a - x\rvert}{\ln(a+2) + \ln x}$ | 12 | 10 | 2 | 6 |
| 6 | $\frac{e^{-a^2} + e^{-x^2}}{2,8ax}$ | 15 | 0,5 | 1 | 2 |
| 7 | $\frac{\sin\sqrt{a^3 + x}}{14 + ax}$ | 12 | 1 | 2 | 3 |
| 8 | $\frac{\cos^2 a + \cos ax}{ax + 2}$ | 15 | 1,5 | 1 | 2,5 |
| 9 | $\sqrt{\frac{1,6ax + \sqrt{x}}{2,9a + 1,2}}$ | 10 | 0,1 | 4,2 | 6 |
| 10 | $\frac{\ln\lvert a + x\rvert}{\ln\lvert a\rvert + 1,5\ln\lvert x\rvert}$ | 8 | -2,5 | -1,9 | -0,9 |
| 11 | $e^{x} + a^{1,7}$ | 10 | 1,1 | 1 | 2 |
| 12 | $\cos\frac{\sqrt{x} + \sqrt{a} + 1}{\sqrt{ax}}$ | 12 | 3 | 2 | 3 |
| 13 | $\frac{\sin ax + \sin^2 a}{4 + \sin^2 x}$ | 15 | 2 | 1,5 | 2,9 |
| 14 | $\frac{\sqrt[3]{a+x} + \sqrt[3]{a+2}}{a+x}$ | 10 | 3 | 1,5 | 3,5 |
| 15 | $\ln\frac{\lvert 2ax\rvert}{\lvert a - x\rvert}$ | 12 | 3 | 2 | 3,5 |

| | | | | | |
|---|---|---|---|---|---|
| 16 | $\frac{1}{e^{ax}+2e^{a}}$ | 15 | 1,5 | 1 | 2,5 |
| 17 | $\frac{\sin^5 x+ax}{x+\cos^5 x}$ | 10 | 2 | 2,5 | 3,5 |
| 18 | $\frac{\sqrt{a+3x}}{\sqrt{a}+3+x}$ | 12 | 2 | 0 | 5 |
| 19 | $\sqrt[5]{\frac{a+30}{a+\sqrt{ax}}}$ | 20 | 4 | 1 | 10 |
| 20 | $\frac{\ln(a^2+x^2)}{\lvert a+x\rvert}$ | 15 | 2 | 1 | 5 |
| 21 | $1{,}5\cos^4\frac{a+x}{2a^2+x^2}$ | 10 | 1,5 | 1 | 4 |
| 22 | $\frac{\sqrt{a+\sin x}}{\sqrt[3]{4+\cos x}}$ | 15 | 2 | 0 | 1 |
| 23 | $\frac{\ln(x^4-a^2)}{a^4+28}$ | 18 | 1,5 | 2 | 3,5 |
| 24 | $\frac{tgax+tgx}{tg(a+1)}$ | 15 | 1,2 | 0,1 | 0,25 |
| 25 | $\frac{\sin^4 a+\sin^4 x}{a+x}$ | 12 | 0,5 | -π | +π |
| 26 | $\frac{\sqrt{ax+0{,}2x}}{2a^2+x}$ | 15 | 1,5 | 2 | 4 |
| 27 | $\sqrt{\frac{3x+4ax}{10a}}$ | 20 | 2 | 1 | 2,5 |
| 28 | $\frac{e^{-x}+e^{a}}{a+x}$ | 12 | 2,5 | -1 | 1 |
| 29 | $5\sqrt{a+\ln a+\ln x}$ | 15 | 6 | 2 | 5 |
| 30 | $tgax+tg^2(a+2{,}5)$ | 16 | 2 | 0 | 1 |
| 31 | $\frac{\cos\sqrt{x}+\cos x}{\sqrt{a+1}+x}$ | 20 | 3 | 4 | 8 |

### Методические указания по выполнению работы

Прежде всего, следует отметить, что особенностью задач, решаемых в настоящей лабораторной работе, является необходимость организации арифметического цикла. Напомним, что арифметическим циклом называется цикл с известным количеством повторением тела. Среди существующих в языке **Си** циклических инструкций (циклы – **for, while** и **do while),** отсутствует циклическая инструкция, специально предназначенная для реализации арифметического цикла. В рассматриваемом случае можно использовать циклические инструкции **for**

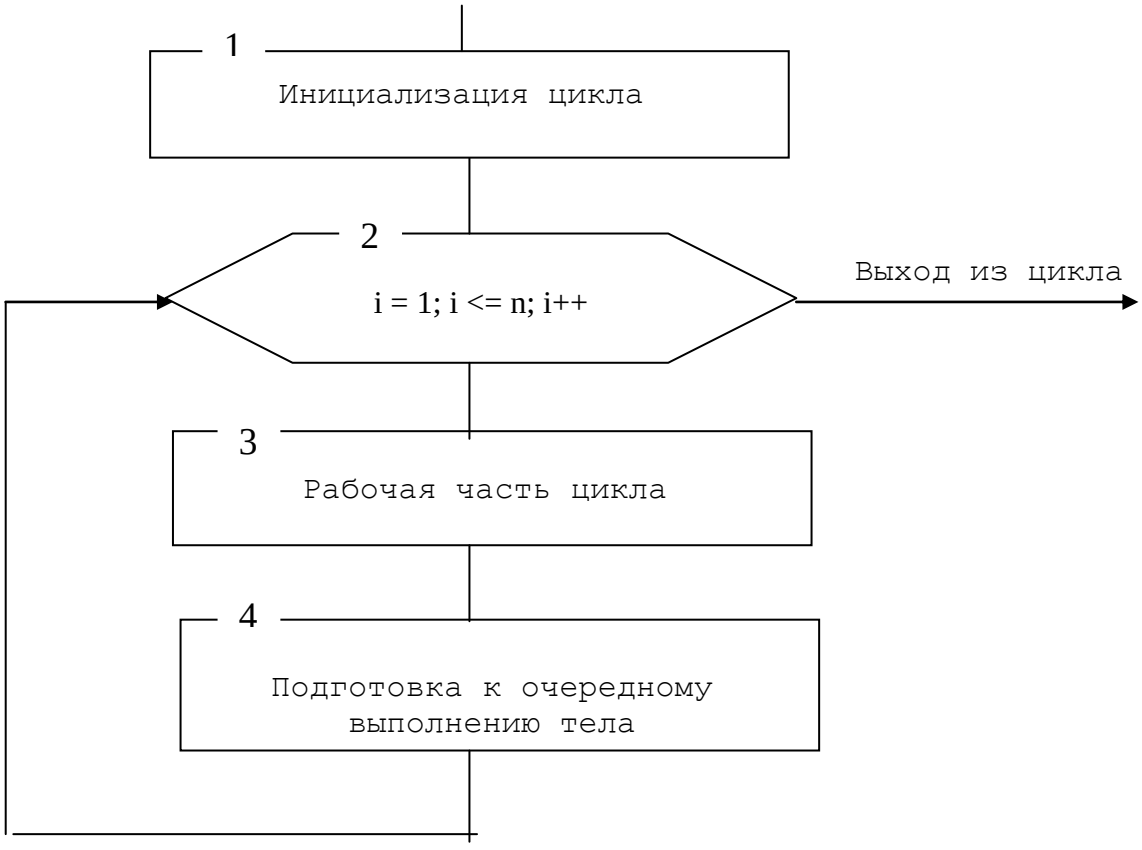
и **while**. Цикл **do while** менее удобен. Это обусловлено тем обстоятельством, что его тело должно выполниться хотя бы один раз. Оставим в качестве задания на самостоятельную работу рассмотрение тех нежелательных последствий, которые повлечет применение этого вида цикла при решении задач настоящей лабораторной работы. Затем необходимо выбрать вид переменной, управляющей работой цикла. Для управления циклом можно использовать следующие переменные:

- аргумент функции x,
- специальную переменную целого типа, которую обычно называют счетчиком.

Использование счетчика является более предпочтительным. Дело заключается в следующем. В общем случае аргументом табулируемой функции может быть переменная вещественного типа. В связи с этим может возникнуть проблема обеспечения заданного количества повторений цикла. Это обусловлено тем положением, что данные вещественного типа имеют приближенное представление в памяти компьютера и все арифметические операции с ними выполняются приближенно. От этих недостатков свободны данные целого типа.

Общий вид циклической части алгоритма решения задач в настоящей лабораторной работе имеет вид, приведенный на рис. 1.3.1. Символ 2 соответствует оператору цикла **for**

.




Обобщенная схема алгоритма решения задачи.

В качестве примера рассмотрим задачу варианта 31. Схема алгоритма для этой задачи приведена на рис.1.3.2. В соответствии с условием задачи необходимо предусмотреть ввод исходных данных: значений переменных n, $x_{нач}$ и $x_{кон}$.

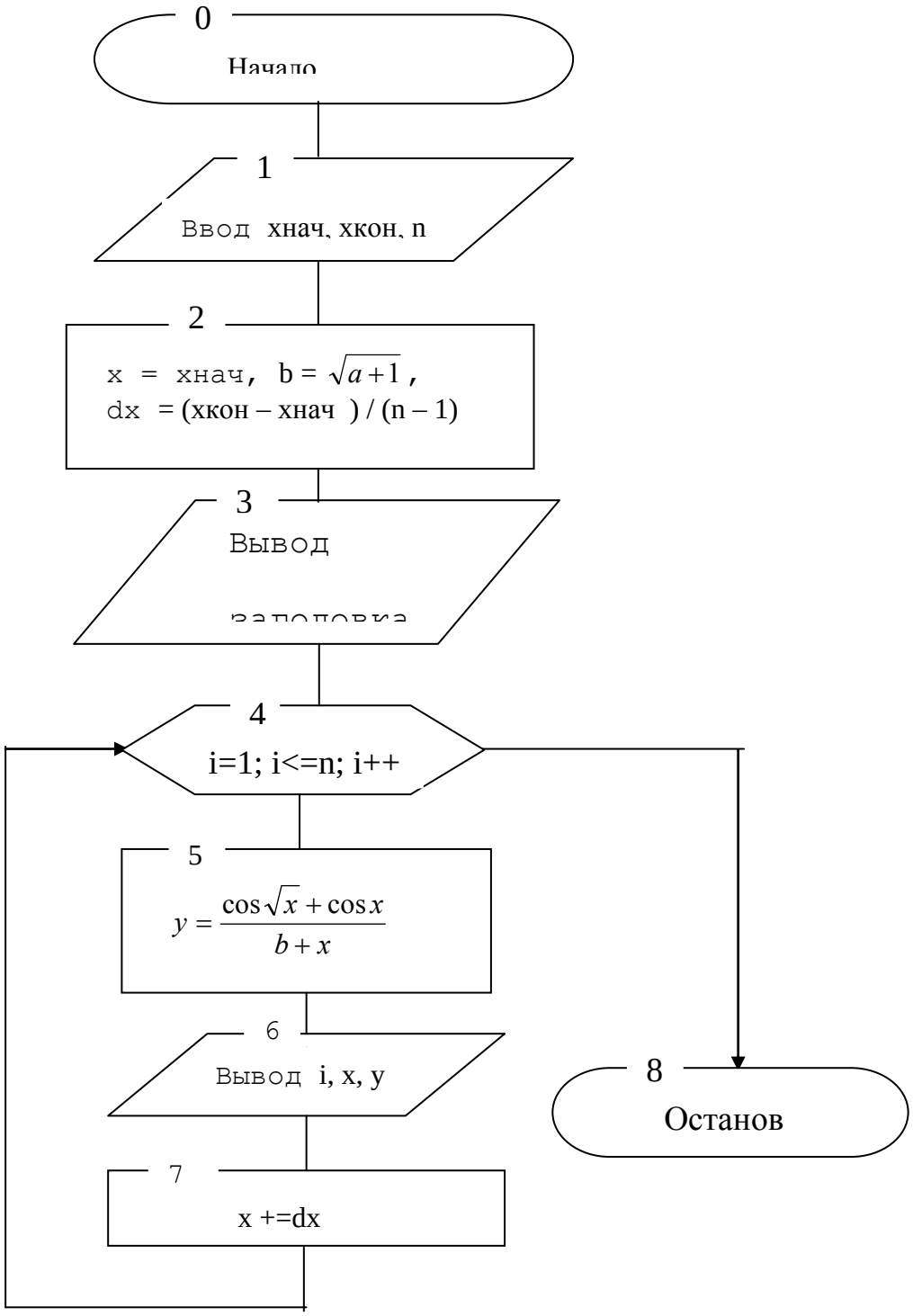



Схема алгоритма решения задачи.

Подготовка к первому выполнению включает в себя присвоение независимой переменной *x* начального значения (символ 2 на рис. 1.3.2), вычисление величины шага изменения аргумента – *dx* (символ 2 на рис. 1.3.2) и вывода заголовка таблицы (символ 3 на рис.1.3. 2).

Анализ расчетной формулы для вычисления величины *y* показывает, что в нее входит выражение, независящее от *x:* $\sqrt{a+1}$. Введем для его обозначения вспомогательную переменную *b*:

$$b = \sqrt{a+1}.$$

Значение вспомогательной переменной *b* целесообразно вычислять заранее, при подготовке к первому вычислению цикла, что позволит избежать многократного вычисления этой величины в цикле (символ 2 на рис.1.3.2). Процедуру, связанную с вынесением из цикла действий, результат выполнения которых в цикле не изменяется, называют "чисткой цикла".

В рабочей части цикла необходимо вычислять значение *y* и выводить на экран результат решения – значения *i, x* и *y* (символы 5 и 6 на рис. 1.3.2).

Подготовка к новому выполнению цикла состоит в изменении аргумента *x* на заданный шаг *dx* (символ 7 на рис. 1.3.2).

В таблице приведены идентификаторы переменных для варианта 31.

**Таблица идентификаторов**

| Обозначение в задаче | Идентификатор | Назначение |
|---|---|---|
| *N* | *N* | Количество расчетных точек |
| *a* | *a* | Параметр функции |
| $x_{нач}$ | *xn* | Начальное значение аргумента |
| $x_{кон}$ | *xk* | Конечное значение аргумента |
| - | *dx* | Шаг изменения аргумента |
| *x* | *x* | Текущее значение аргумента |
| *y* | *y* | Вычисленное значение аргумента |
| - | *I* | Счетчик цикла |
| | *b* | Промежуточная переменная |

Отметим, что при организации цикла очень важным является определение основной операции, применение которой позволяет получить нужный результат. Такую операцию будем называть **опорной**. Такой операцией при решении задачи табулирования является операция, задаваемая оператором присваивания *x += dx*. Эта операция позволяет повторно использовать для вычислений расчетную формулу, стоящую в рабочей части цикла.

По условию задачи результаты вычислений должны быть оформлены в виде таблицы, снабженной заголовком. Это легко реализуется при использовании форматированного вывода. При этом следует согласовывать элементы

форматирования, используемые при выводе заголовка с элементами форматирований, которые используются при выводе строк таблицы.
Например, заголовок таблицы можно выводить с помощью следующего вызова функции printf()

```
printf("%5s%s10%10s\n","Номер","Аргумент",
          "Функция");
```

В этом случае вывод очередной строки таблицы может быть выполнен с помощью следующего вызова функции printf():

```
printf("%5d%10.2f%10.2f\n", i, x, y );
```

### Методические указания по выполнению контрольного расчета

Для выполнения контрольного расчета в данной лабораторной работе необходимо выбрать численные значения величин *n*, *a*, *xn*, *xk* и *a*.
Для сокращения количества ручных вычислений, выполняемых в контрольном расчете, значение величины *n* можно взять равной 3. Заметим, что выбор в контрольном расчете *n* = 2 является нежелательным. Дело заключается в том, что при организации цикла табулирования встречается ошибка, которую при *n* = 2 выявить не удается. Такая ошибка возникает в том случае, когда оператор, осуществляющий подготовку к новому выполнению в цикле (символ 7), неправильно записывают в следующем виде: *x* = *xn* + *dx*.
При расчете на компьютере прохождение цикла выполняется трижды, что позволит проверить правильность организации цикла. Значения величин *xn*, *xk* и *a* целесообразно выбирать таким образом, чтобы упростить вычисления, выполняемые вручную.
Например, для варианта 31 можно выбрать для контрольного расчета *xn* = 0.5, *xk* = 1.5 и *a* = 3.
Результаты вычислений контрольного расчета для рассматриваемого варианта приведены в таблице.

**Таблица вычислений для варианта 31**

| Назначение набора данных | Набор данных | | | | Результаты вычислений | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | ручных | | машинных | |
| | *N* | *a* | *xn* | *xk* | *X* | *y* | *x* | *y* |
| Контрольный | 3 | 3 | 0,5 | 1,5 | 0,5 | 0,65513 | | |
| | | | | | 1,0 | 0,36020 | | |
| | | | | | 1,5 | 0,11712 | | |
| Рабочий | 20 | 3 | 4 | 8 | | | | |

# Лабораторная работа №4

## Циклические вычислительные процессы. Вычисления по рекуррентным формулам.

Определить значение выражения с использованием 3 циклов. Для вывода значений выражения использовать оператор выбора swith()

Задание: ввести с клавиатуры символ.

Если этот символ - «f» или «F» - то вычислить значение выражения с помощью оператора FOR;

если этот символ - «w» или «W» - с помощью оператора WHILE;

если этот символ - «d» или «D» - с помощью оператора DO WHILE;

если введен какой-либо другой символ — вывести сообщение «НЕВЕРНЫЙ СИМВОЛ»

Вместо символов f, w или d можно вводить цифры 1,2 или 3.

В программе должна быть осуществлена многократная проверка вычисления значения выражения всеми циклами без повторного запуска программы (зациклить switch() )

| № | Вид функции | Рабочий набор исх. данных | |
|---|---|---|---|
| | | X | N |
| 1 | $y=2{,}8\,x\cdot\sum_{i=1}^{N}(\cos ix+\frac{\cos x}{\cos i})$ | 2,8 | 10 |
| 2 | $z=\sqrt{\sum_{i=2}^{N}(i^2+\sqrt{i}+\sqrt{2{,}3x})}$ | 6,3 | 11 |
| 3 | $y=\cos(\prod_{i=4}^{N}(0{,}1\,i+\cos x))$ | 4,64 | 12 |
| 4 | $y=29\,x^{4{,}2}+\sum_{i=1}^{N}(\sqrt{i}+\sqrt{x}+2{,}3)$ | 5,61 | 8 |
| 5 | $y=441\cdot\cos x+\prod_{i=2}^{N}(4{,}1\cos x+\sqrt[3]{i})$ | 0,5 | 10 |
| 6 | $y=(\prod_{i=1}^{N}(2{,}8\,x^{4{,}5}+\sqrt[3]{i}))^{1{,}2x}$ | 1,8 | 9 |
| 7 | $y=\sin^2 x+\sin^2\sum_{i=4}^{N}(\sin i+\sin^2 x)$ | 0,2 | 12 |
| 8 | $z=\tan^2 x-\prod_{i=2}^{N}(\tan^2 x\cdot\sin^2 i+1)$ | 0,5 | 11 |
| 9 | $y=\sqrt[3]{\sum_{i=1}^{N}(\sqrt{ix}+\sqrt{i}+1{,}2\sqrt{x})}$ | 2,81 | 9 |

| | | | |
|---|---|---|---|
| 10 | $z=\ln 4{,}8\cdot\prod_{i=2}^{N}(\cos^2 x+x-2{,}8\sqrt{i})$ | 5,3 | 10 |
| 11 | $y=e^{-x^2}+\sum_{i=2}^{N}(e^{-x^2}+e^{-i^2})$ | 1,83 | 11 |
| 12 | $z=\cos^2 x+\cos(\prod_{i=1}^{N}(\tan\lvert x\rvert\cdot\sqrt{i}))$ | 0,01 | 10 |
| 13 | $y=\tan\lvert x\rvert\cdot\sum_{i=1}^{N}(\tan\lvert x\rvert\cdot\sqrt{i})$ | 0,5 | 8 |
| 14 | $z=\sqrt{\prod_{i=1}^{N}(\sqrt{i}+\sqrt{x}+0{,}64)}$ | 2,68 | 9 |
| 15 | $y=(\sum_{i=3}^{N}(\cos^2 x+2{,}8\cos^2 i))^4$ | 1,64 | 12 |
| 16 | $z=2{,}3\prod_{i=1}^{N}(e^{-x}+2{,}8\cos\ i)$ | -2,2 | 8 |
| 17 | $y=3\sin x+\prod_{i=2}^{N}(2{,}8\cdot x^3+2{,}8\sin ix)$ | 0,38 | 9 |
| 18 | $z=\sqrt{\lvert\cos\rvert x}+\cos\sum_{i=1}^{N}(\cos^3 x+\tan 0{,}01\,ix)$ | 0,53 | 10 |
| 19 | $y=\sin\prod_{i=3}^{N}(e^{-x^2}+e^{0{,}1}i)$ | 1,87 | 11 |
| 20 | $z=\cos x+\cos\sum_{i=1}^{N}(8{,}2\frac{x}{\cos}x+\cos i)$ | 14,64 | 9 |
| 21 | $y=e^{x^2}+\prod_{i=4}^{N}(\sqrt{i}+0{,}1\sqrt{x})$ | 6,43 | 12 |
| 22 | $z=15{,}9\,x\cdot\sum_{i=1}^{N}(\tan^4 x+\tan^4 0{,}01\,i)$ | 0,35 | 8 |
| 23 | $y=480\cdot\cos x+\prod_{i=1}^{N}(\sqrt{x+2{,}8}+\sqrt{i+2{,}8})$ | -0,86 | 9 |
| 24 | $y=\dfrac{\sum_{i=1}^{N}(\cos^2 x+\cos^4 x+\cos^4 i)}{2{,}2+\cos^4 x}$ | 0,8 | 8 |
| 25 | $z=\frac{2{,}3}{x}+2{,}3\cdot\prod_{i=4}^{N}(\tan^2(x+1)-\tan(x+0{,}01\,i))$ | -0,1 | 11 |
| 26 | $y=\ln\lvert x-2{,}8\rvert+\ln\left\lvert\sum_{i=1}^{N}(2{,}6\ln\lvert x-2{,}8\rvert+i)\right\rvert$ | 1,6 | 10 |
| 27 | $z=1{,}6\,x^4+\prod_{i=4}^{N}(1{,}6\,x^4+0{,}06\sqrt{i})$ | 1,03 | 13 |

| | | | |
|---|---|---|---|
| 28 | $y=\sin^4 x-\sum_{i=1}^{n}(4\cos x^2+2{,}8i\cdot x)$ | 2 | 12 |
| 29 | $z=49x+4{,}9\prod_{i=3}^{N}(e^{-x^{1,2}}+\sqrt[3]{i}+x)$ | 1,86 | 10 |
| 30 | $y=11{,}6\cos x+\cos\sum_{i=1}^{N}(\cos^2 x+\sqrt{x}+\sqrt{i})$ | 0,01 | 12 |
| 31 | $y=6{,}3x-4\sum_{k=3}^{N}(2x^3k+\cos k\sqrt{x+1}-\frac{2{,}3}{k})$ | 4,75 | 20 |

## Лабораторная работа 5

# Программирование вложенных циклов

### *Постановка задачи*

В настоящей лабораторной работе необходимо выполнить вычисления, для организации которых следует использовать несколько циклов, причем некоторые из них должны быть вложенными.

### *Варианты заданий*

В работе необходимо вычислять значение (я) функции $y = f(x)$. Варианты заданий отличаются видом функции (см. таблицу, приведенную ниже). В нечетных вариантах заданий необходимо вычислять значение функции для одного значения аргумента $x$, а в четных следует решать задачу табулирования. При вычислении значения функции оказывается необходимым вычислять несколько сумм (произведений). Вычисление некоторых сумм (произведений) может потребовать организации вложенных циклов.

| Номер | Функция | Рабочий набор | | | | |
|---|---|---|---|---|---|---|
| | | $x$ | $m$ | $x_{нач}$ | $x_{кон}$ | $n$ |
| 1 | $\sum_{i=1}^{n} \frac{2x + \sum_{j=1}^{m}(i+j)^2}{x + i \sum_{k=1}^{m}(k+1)}$ | 2 | 20 | - | - | 10 |
| 2 | $x^2 + \sum_{i=2}^{m} \frac{x+i^2}{2x+i} + x \prod_{j=2}^{m+1} \sqrt{j}$ | - | 10 | 1 | 5 | 20 |
| 3 | $\sum_{i=1}^{n} \frac{10 + \sum_{j=1}^{m}(j+i)^2}{20 + i^2 \sum_{k=1}^{m} k^2}$ | 1,5 | 25 | - | - | 7 |
| 4 | $\sqrt{2x} + \prod_{j=2}^{m} \sqrt{j+1} + \sum_{k=1}^{m}(x+k)^2$ | - | 12 | 2 | 10 | 20 |
| 5 | $\sum_{i=2}^{n} \frac{x^3 + (\sum_{k=1}^{m} k^3)^2}{x + \sum_{j=1}^{m} \sqrt{i+j}}$ | 3 | 20 | - | - | 12 |
| 6 | $\sin(x + \sum_{k=1}^{m}(x+k)^2) + \sum_{j=1}^{m} j^3$ | - | 15 | 1 | 20 | 15 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 7 | $\sum_{i=1}^{n} \frac{5x + \sum_{k=1}^{m} (k+x)^2}{x + \sum_{j=1}^{m} \sqrt{2j+3i}}$ | 2,5 | 12 | - | - | 9 |
| 8 | $\sqrt[3]{x + \sum_{k=1}^{m} (k+1)^2 + \sum_{j=1}^{m} \sqrt{x+j}}$ | - | 10 | 1 | 7 | 15 |
| 9 | $\sum_{i=1}^{n} \frac{x + \sum_{k=1}^{m} (-1)^k k}{1 + \sum_{j=1}^{m} (2i+j)}$ | 2 | 15 | - | - | 12 |
| 10 | $\frac{x + \sum_{k=1}^{m} k^3}{x^2 + \sum_{j=1}^{m} (j+x)^2}$ | - | 12 | 1 | 8 | 15 |
| 11 | $\sum_{i=1}^{n} \frac{3x + \sum_{k=1}^{m} (k+i)^2}{4x + \sum_{j=1}^{m} \sqrt{2+j}}$ | 2 | 10 | - | - | 11 |
| 12 | $2x^2 + \prod_{k=1}^{m} (x+k) + x \sum_{j=1}^{m} j^3 + 2$ | - | 10 | 3 | 8 | 15 |
| 13 | $\sum_{i=1}^{n} \frac{0,5x + \sum_{j=1}^{m} (2j+1)^2}{x + \sum_{k=1}^{m} (i+2k)^3}$ | 2 | 20 | - | - | 14 |
| 14 | $3x + \sum_{j=1}^{m} (x+j)^2 + \prod_{k=1}^{m} \frac{1+k^2}{4+k} + 1$ | - | 4 | 1 | 5 | 20 |
| 15 | $\sum_{j=1}^{n} \frac{2x + \sum_{i=1}^{m} (i+1)^2}{4x + \sum_{k=1}^{m} (k+j)^2}$ | 1,5 | 10 | - | - | 12 |
| 16 | $5x + \sum_{k=1}^{m} \left(\frac{x}{k} + \frac{k}{x}\right) + x \sum_{i=1}^{m} i^2$ | - | 1 | 4 | 10 | 15 |
| 17 | $\sum_{i=1}^{n} \frac{1 + x \sum_{k=1}^{m} \left(\frac{1}{k} + k\right)}{3i + \sum_{j=1}^{m} \left(x + \frac{j}{i}\right)^2}$ | 4,5 | 15 | - | - | 10 |
| 18 | $\sin\left(x + \sum_{i=1}^{m} \frac{1}{i}\right) + \cos\left(1 + \sum_{j=1}^{m} \left(\frac{j}{x} + \frac{x}{j}\right)\right)$ | - | 0,5 | 1 | 10 | 15 |
| 19 | $\sum_{i=2}^{n} \frac{xi + \sum_{k=2}^{m} k^3}{2 + \sum_{j=2}^{m} (i+jx)^2}$ | 1,2 | 15 | - | - | 9 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 20 | $\sqrt{2x^2 + \sum_{j=1}^{m}(2 + x + \frac{j}{x}\sum_{k=1}^{m}k^2)^2}$ | - | 1,5 | 2,5 | 15 | 20 |
| 21 | $\sum_{j=1}^{n}\frac{jx + \sum_{k=1}^{m}(\frac{k}{j} + \frac{j}{k})}{2j + x\sum_{i=1}^{m}\frac{1}{i}}$ | 5 | 15 | - | - | 10 |
| 22 | $\sqrt[4]{1 + (2 + \sum_{i=1}^{m}i^2 + \sum_{k=1}^{m}(1 + kx)^2)^2}$ | - | 1 | 2 | 10 | 15 |
| 23 | $\sum_{k=1}^{n}\frac{kx + \sum_{j=1}^{m}j^3}{1 + \sum_{i=1}^{m}(1 + \frac{i}{k})}$ | 2 | 15 | - | - | 14 |
| 24 | $\frac{x + \sum_{i=1}^{m}(1 + \frac{1}{i})}{1 + \sum_{k=1}^{m}(x + \frac{5}{k})^2}$ | - | 10 | 1 | 5 | 15 |
| 25 | $\sum_{k=1}^{m}(\frac{k}{x} + \frac{x}{k}) + \sum_{i=1}^{n}\frac{4 + \sum_{j=1}^{m}\sqrt{j+i}}{x + \sum_{j=1}^{m}\sqrt{j+i}}$ | 2 | 20 | - | - | 6 |
| 26 | $2x + \sum_{k=1}^{m}(\frac{k}{x} + \frac{x}{k}) + \sum_{j=1}^{m}\sqrt{10 + j}$ | - | 10 | 1 | 3 | 15 |
| 27 | $\prod_{k=1}^{m}(\frac{k}{x} + \frac{x}{k}) + \sum_{j=1}^{n}\frac{20x + \sum_{i=1}^{m}\sqrt{j+i}}{1 + \prod_{i=1}^{m}\sqrt{j+i}}$ | 3 | 16 | - | - | 7 |
| 28 | $\sqrt{\left\lvert 2x - \prod_{k=1}^{m}\sqrt{x+k} + \sum_{j=1}^{m}\sqrt{j}\right\rvert}$ | - | 10 | 0 | 5 | 20 |
| 29 | $2x + \sum_{i=1}^{n}\frac{x + \sum_{j=1}^{m}(i-j)^2 + \sum_{k=1}^{m}k^2}{i + \prod_{j=1}^{m}\sqrt{\lvert i-j\rvert}}$ | 5 | 25 | - | - | 11 |
| 30 | $\sqrt{1 + 20x + \sum_{k=1}^{m}(k - \frac{1}{k})^2 + \sum_{j=1}^{m}(j+x)^2}$ | - | 15 | 0 | 3 | 25 |

### *Методические указания по выполнению работы*

Характеристики программы в значительной степени зависят от ее структуры. Поэтому разработке структуры программы должно быть уделено большое внимание. При разработке структуры программы необходимо

определить количество циклов и их взаимное расположение. При этом следует определить, какие из них должны быть вложенными. Кроме того, некоторые циклические вычислительные процессы могут быть реализованы с помощью одного цикла. Рассмотрим вопрос о разработке программы применительно к варианту 29.

При решении задачи для варианта 29 необходимо организовать четыре циклических вычислительных процесса. Первый из них должен использоваться для вычисления внешней суммы и три других - для вычисления внутренних сумм и произведения. Для вычисляемых сумм и произведения введем следующие обозначения: S – внешняя сумма, S1, S2 – внутренние суммы и P – произведение. Указанные величины должны вычисляться по следующим формулам:

$$S1 = \sum_{j=1}^{m} (i - j)^2,$$

$$S2 = \sum_{k=1}^{m} k^2,$$

$$P = \prod_{j=1}^{m} \sqrt{|i - j|},$$

$$S = \sum_{i=1}^{n} \frac{x + S1(i) + S2}{i + P(i)}.$$

Если не учитывать особенности расчетных формул для вычисления величин *S*, *S1*, *S2* и *P*, то можно придти к следующей структуре программы. Программа должна содержать четыре циклических алгоритма (по числу циклических вычислительных процесса). Причем три цикла (циклы для вычисления величин *S1*, *S2* и *P*) должны быть вложены в цикл, предназначенный для вычисления величины *S*.

Выполняя более детальный анализ указанных формул, следует обратить внимание на следующее. В формулу, определяющую значение величины *S2*, не входит переменная суммирования *i*, которая используется для вычисления величины *S*. Это позволяет выполнить вычисления величины *S2* один раз. Для этого цикл, определяющий величину *S2*, необходимо вынести из вложенных циклов. Затем, как не трудно видеть, вычисления величин *S1* и *P* можно выполнить в одном цикле.

Изложенные выше соображения позволяют предложить структуру программы, изображенную на рис.5.1. Структура программы представлена с помощью диаграмм Насси – Шнейдермана.

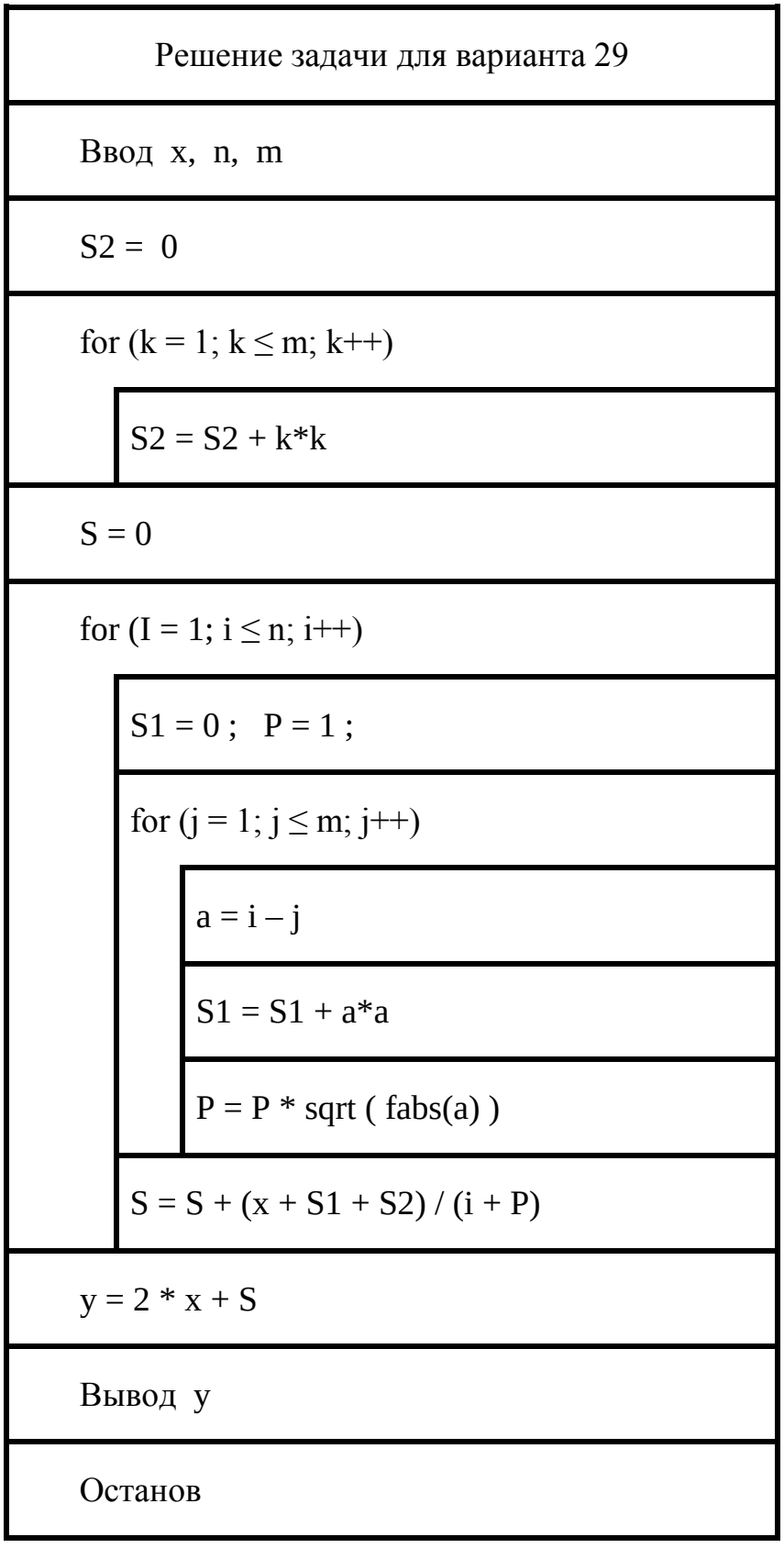

Решение задачи для варианта 29
Ввод x, n, m
S2 = 0
for (k = 1; k ≤ m; k++)
S2 = S2 + k*k
S = 0
for (I = 1; i ≤ n; i++)
S1 = 0 ; P = 1 ;
for (j = 1; j ≤ m; j++)
a = i – j
S1 = S1 + a*a
P = P * sqrt ( fabs(a) )
S = S + (x + S1 + S2) / (i + P)
y = 2 * x + S
Вывод y
Останов

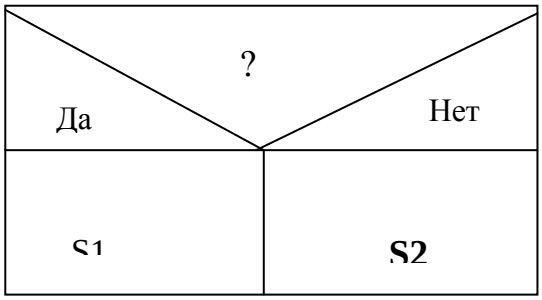
Рис. 5.1

### *Символы диаграмм Насси-Шнейдермана*

**1. Процесс**

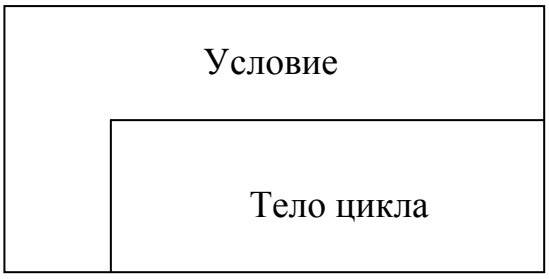
Один или несколько операторов, выполнение которых происходит последовательно.

**2. Если (условие) То _____ Иначе _____**




Проверка условия или принятие решения (верхний треугольник), в результате чего управление передается в один из нижних прямоугольников.

**3. Цикл “ПОКА”**




Тело цикла (внутренний прямоугольник) повторяется, пока выполняется некоторое условие. После этого управление выходит из внешнего прямоугольника. Горизонтальная полоса показывает место нахождения в цикле проверки условия.

**4. Цикл “ДО”**


Тело цикла (внутренний прямоугольник) также повторяется до тех пор, пока выполняется некоторое условие. После этого управление выходит из внешнего прямоугольника. Горизонтальная полоса показывает место проверки условия нахождения в цикле.

**Контрольные вопросы**

1. Какие циклы называются вложенными?
2. Укажите, какие компоненты Вашей программы относятся к внешнему циклу.
3. Укажите, какие компоненты Вашей программы относятся к внутреннему циклу?
4. Найдите в Вашей программе подготовку внешнего и внутреннего циклов. Как отразится на работоспособности программы их отсутствие?
5. Укажите, сколько раз за время работы Вашей программы выполнятся операторы, расположенные в теле внутреннего цикла?
6. Укажите, пришлось ли Вам при организации вложенных циклов использовать составной оператор? Рассмотрите вопрос о том, как будет работать программа в его отсутствие.

## Лабораторная работа 6

# Организация функций.

### *Цель работы*

Целью настоящей работы является ознакомление студентов с правилами организации функций.

### *Постановка задачи*

Вычислить значение величины, содержащей несколько однотипных сумм. Для вычисления сумм написать функцию пользователя.

### *Варианты заданий*

Варианты заданий приведены в таблице

| Номер варианта | Расчетная формула |
|---|---|
| 1 | $y = \frac{a + \sum_{i=1}^{m} (2*i^2 + i + 2)^2}{4 + \sum_{i=2}^{n} (i^2 + 3)^2}$ |
| 2 | $y = \frac{5 + \sum_{i=3}^{m} (i^2 + a^2 + 1)^3}{a^2 + \sum_{i=4}^{n} (a + 3 + 2*i^2)^3}$ |
| 3 | $y = \frac{c^2 + \sum_{j=2}^{n1} (j^3 + 5)}{4*a + \sum_{j=3}^{n2} (2*j^3 + j + c)}$ |
| 4 | $y = \frac{10 + \sum_{k=2}^{m} (k^2 + 2*k + c)}{a + \sum_{k=3}^{m} (2*k + 3)}$ |
| 5 | $y = \frac{6 + a * \sum_{i=2}^{m} (3*i^2 + 2*i + c)}{4 + \sum_{i=1}^{m} (i^2 + 2) + \sqrt{\sum_{i=2}^{m} (i + 3)}}$ |
| 6 | $y = \frac{2 * \sum_{k=1}^{m} (k^3 + 2) + a * \sum_{k=1}^{m} (l^3 + 3)}{6 + \sum_{k=3}^{m+2} (5*k^3 + a)}$ |
| 7 | $y = \frac{\sum_{j=1}^{m} (j^3 + 3*j^2 + 1) + a}{5 + \sum_{j=2}^{m+1} (j^2 + a)}$ |
| 8 | $y = \frac{3 * \sum_{i=3}^{m} (i + 2) + a * \sum_{i=4}^{m+1} (i^2 + 2)}{a + \sum_{i=5}^{m+2} (2*i^2 + i + 4)}$ |

| | |
|---|---|
| 9 | $y = \dfrac{2* \sum\limits_{i=2}^{m}(3*i^3+5)}{2* \sum\limits_{i=2}^{m}(3*i^3+i+1)+ \sum\limits_{i=3}^{m+1}(2*i^3+b)}$ |
| 10 | $y = \dfrac{a^2+2* \sum\limits_{j=2}^{n}(j^2+2)}{4+3* \sum\limits_{j=3}^{m}(a^2+j^2+j)}$ |
| 11 | $y = \dfrac{5+ \sum\limits_{j=2}^{n}(3*j^3+j^2+c)}{a+2* \sum\limits_{j=4}^{m}(2*j^2+3)}$ |
| 12 | $y = \dfrac{1+2* \sum\limits_{j=3}^{n}(3*j^3+j^2+1)}{2+ \sum\limits_{j=2}^{m}(2*j^3+2)}$ |
| 13 | $y = \dfrac{a* \sum\limits_{k=3}^{n}(2k+1)^2}{1+2* \sum\limits_{k=1}^{m}(3*k+a)^2+3* \sum\limits_{k=3}^{n}(k+3)^2}$ |
| 14 | $y = \dfrac{b+2* \sum\limits_{j=2}^{m}(j^2+2)}{1+3* \sum\limits_{j=3}^{n}(2*j^2+3)+4* \sum\limits_{j=2}^{m}(3*j^2+j+4)}$ |
| 15 | $y = \dfrac{a+ \sum\limits_{k=1}^{m}(k^3+3)+3* \sum\limits_{k=3}^{n}(k^2+3)}{10+ \sum\limits_{j=1}^{m}(2*j^3+j^2+1)}$ |
| 16 | $y = \dfrac{a+ \sum\limits_{l=2}^{n}(2*l^3+3*l^2+1)}{2+ \sum\limits_{k=3}^{m}(k^2+2)}$ |
| 17 | $y = \dfrac{3* \sum\limits_{i=2}^{n}(i^2+2)+2* \sum\limits_{j=3}^{m}(j^3+1)}{2+ \sum\limits_{k=3}^{m}(2*k^3+k+2)}$ |

| | |
|---|---|
| 18 | $y = \dfrac{5 + 3 * \sum\limits_{k=3}^{n} (2 * k^2 + 1)}{1 + \sum\limits_{k=2}^{m} (k^2 + k + 2) + 4 * \sum\limits_{i=3}^{n} (5 * k + 3)}$ |
| 19 | $y = \dfrac{1.5 + \sum\limits_{i=1}^{m} (4 * i^2 + i + 3)}{2 + \sum\limits_{j=2}^{n} (j^2 + 2)}$ |
| 20 | $y = \dfrac{a + \sum\limits_{k=2}^{n} (2 * k^3 + 1)}{2 + \sum\limits_{k=1}^{m} (k^3 + a)}$ |
| 21 | $y = \dfrac{1 + \sum\limits_{j=1}^{m} (j^3 + 2 * j^2 + 3)}{2 + \sum\limits_{k=3}^{n} (2 * k^2 - 1)}$ |
| 22 | $y = \dfrac{2 + \sum\limits_{k=4}^{n} (0.5 * k^2 + k - 2)}{1 + \sum\limits_{k=2}^{m} (k^2 + 2 * k - 3)}$ |
| 23 | $y = \dfrac{1 + \sum\limits_{j=1}^{m} (2 * j^3 + 1)}{2 + \sum\limits_{j=2}^{n} (j^3 - 2)}$ |
| 24 | $y = \dfrac{1 + \sum\limits_{j=2}^{n} (2 * j^2 + 3 * j + 1)}{2 + \sum\limits_{k=1}^{m} (k^2 + a)}$ |
| 25 | $y = \dfrac{4 + \sum\limits_{i=1}^{n} (3 * i^3 + 2 * i^2 + 1)}{1 + \sum\limits_{k=2}^{m} (k^3 + k^2 + 3)}$ |
| 26 | $y = \dfrac{1 + \sum\limits_{j=1}^{n} (2 * j^2 + 1)}{2 + \sum\limits_{k=2}^{m} (k^2 + a)}$ |

| | |
|---|---|
| 27 | $y = \dfrac{2 + \sum_{k=1}^{n}(k^2 + k + 2)}{\sum_{k=2}^{m}(2*k^2 + 3*k - 3)}$ |
| 28 | $y = \dfrac{1 + \sum_{i=1}^{n}(3*i^2 + a)}{2 + \sum_{i=2}^{m}(2*i^2 + i + 3)}$ |
| 29 | $y = \dfrac{5 + \sum_{j=1}^{m}(2*j^3 + 1)}{3 + \sum_{k=3}^{n}(k^3 + 2)}$ |
| 30 | $y = \dfrac{1 + \sum_{k=1}^{m}(k^2 + 1)}{2 + \sum_{j=3}^{n}(2*j^2 + j + 3)}$ |
| 31 | $y = \dfrac{1 + \sum_{k=1}^{n}(4*k^2 + k + 2)}{3 + \sum_{j=2}^{m}(j^2 + x)}$ |
| 32 | $y = \dfrac{2 + \sum_{k=3}^{n}(k^2 + 4)^2}{1 + \sum_{i=1}^{m}(i^2 + i + 1)^2}$ |

### *Методические указания по выполнению лабораторной работы*

В настоящей лабораторной работе необходимо вычислить значение величины, в расчетную формулу которой входит несколько “похожих” сумм. В таком случае целесообразно организовать функцию пользователя для вычисления этих сумм.

При разработке функции пользователя можно поступить следующим образом. Вначале на основе анализа отдельных сумм напишем общее выражение для вычисления суммы, модификация которого позволит определить значения каждой исходной суммы. Это оказывается возможным при введении некоторого количества параметров. Обратимся к задаче варианта 31. В расчетной формуле для этого варианта необходимо вычислить значение следующих двух сумм:

$$S1 = \sum_{k=1}^{n}(4 * k^2 + k + 2),$$

$$S2 = \sum_{j=2}^{m}(j^2 + x).$$

В данном примере искомое выражение для вычисления суммы может быть записано в следующем виде.

$$summa(n1, n2, a2, a1, a0) = \sum_{i=n1}^{n2}(a2 * i^2 + a1 * i + a0).$$

Сумма *S1* может быть вычислена с помощью следующего вызова функции *summa*(1, n, 4, 1, 2), а сумма *S2* – с помощью вызова функции *summa*(2, *m*, 1, 0, *x*).

Приведем реализацию программы для решения задачи варианта 31.

```
int     summa(int n1, int n2, int a2, int a1, int a0)
{
   int s, i;
   s = 0;
   for  (i = n1; i<= n2; i++)
        s = s + a2 * i * i + a1 * i  + a0;
  return  s;
      }
int  main(void)
{
int  n, m, x;
float  y;
   printf("Введи  n=");
   scanf("%d", &n);
   printf("Введи  m=");
   scanf("%d", &m);
   printf("Введи  x=");
   scanf("%d", &x);
   y =(float) (1 +summa(1, n, 4, 1, 2)) / (3 + summa(2, m, 1, 0, x));
!!!
//добавить замечание про (float)
  printf("y=%8.3f", y);
getch();
return 0;
}
```

В отчете по лабораторной работе следует привести две схемы алгоритма. Первая из них должна относиться к функции main(), а вторая - к подпрограмме (функции *summa()*)

### *Контрольные вопросы*

1. Назначение подпрограмм.
2. Структура программы при использовании подпрограмм.
3. Сравните два способа организации связи с подпрограммой: внешние переменные и параметры.
4. Какие существуют способы передачи параметров в функцию?
5. Опишите механизм передачи параметров по значению.
6. В чем состоит побочный эффект при использовании функций?
7. Когда используются локальные переменные?
8. Какова область видимости локальных переменных?

## Лабораторная работа 7

# Обработка одномерных массивов

### *Постановка задачи*

В настоящей лабораторной работе необходимо выполнить заданную обработку одномерного массива. Все основные действия следует выполнять с помощью функций (ввод исходных массивов, формирование новых массивов).

Даны числовые последовательности, состоящие из $n$ элементов вещественного типа ($n$ <= 20). Сформировать новые числовые последовательности в соответствии с заданным правилом (см. табл. 1.7.1).

Лабораторная работа выполняется в виде проекта, состоящего из двух файлов. Один файл содержит все функции, выполняющие обработку массивов, а второй файл содержит функцию **main()**, в которой происходит обращение к функциям из первого файла.

### *Варианты заданий*

Варианты заданий приведены в таблице

| N | Задание |
|---|---|
| 1 | Даны три числовые последовательности *a*, *b* и *c*. Сформировать две новые последовательности *x* и y в соответствии со следующим правилом<br>$x_i$ = (max_a + max_b) – ( $a_i$ + $b_i$ ) / 2,<br>$y_i$ = (max_b + max_c) – ( $b_i$ + $c_i$ ) / 2,<br>i = 1, 2, …, n.<br>Здесь max_a, max_b и max_c – значения максимальных элементов числовых последовательностей *a, b* и *c* |
| 2 | Даны три числовые последовательности *a*, *b* и *c*. Сформировать две новые последовательности *x* и y в соответствии со следующим правилом<br>$x_i$ = max ( $a_i$ , $b_i$ ) / 2,<br>$y_i$ = max ( $b_i$ , $c_i$ ) / 2,<br>i = 1, 2, …, n |
| 3 | Даны три числовые последовательности *a*, *b* и *c*. Сформировать две новые последовательности *x* и y в соответствии со следующим правилом<br>$x_i$ = min ( $a_i$ , $b_i$ ) / 2,<br>$y_i$ = min ( $b_i$ , $c_i$ ) / 2,<br>i = 1, 2, …, n |
| 4 | Даны три числовые последовательности *a*, *b* и *c*. Сформировать две новые последовательности *x* и y в соответствии со следующим правилом<br>$x_i$ = max ( $a_i$ , $b_i$ , d) / 2,<br>$y_i$ = max ( $b_i$ , $c_i$ , d) / 2,<br>i = 1, 2, …, n.; d – произвольное число |
| 5 | Даны три числовые последовательности *a*, *b* и *c*. Сформировать две новые последовательности *x* и y в соответствии со следующим правилом<br>$x_i$ = min ( $a_i$ , $b_i$ , r),<br>$y_i$ = min ( $b_i$ , $c_i$ , r) ,<br>i = 1, 2, …, n.; r – произвольное число |
| 6 | Даны три числовые последовательности *a*, *b* и *c*. Сформировать две новые последовательности *x* и y в соответствии со следующим правилом<br>$x_i$ = ( $a_i$ + $b_i$ ) / 2,<br>$y_i$ = ( $b_i$ + $c_i$ ) / 2,<br>i = 1, 2, …, n |
| 7 | Даны три числовые последовательности *a*, *b* и *c*. Сформировать две новые последовательности *x* и y в соответствии со следующим правилом<br>$x_i = \sqrt{a_i^2 + b_i^2}$ ,<br>$y_i = \sqrt{b_i^2 + c_i^2}$ ,<br>i = 1, 2, …, n |

| | |
|---|---|
| 8 | Даны три числовые последовательности *a*, *b* и *c*. Сформировать две новые последовательности *x* и y в соответствии со следующим правилом<br>$x_i = \sqrt{\lvert a_i * b_i \rvert}$ ,<br>$y_i = \sqrt{\lvert b_i * c_i \rvert}$ ,<br>i = 1, 2, …, n |
| 9 | Даны три числовые последовательности *a*, *b* и *c*. Сформировать две новые последовательности *x* и y. Формирование выполняется в два этапа. На первом этапе осуществляется нормировка исходных последовательностей *a*, *b* и *c*. В результате нормировки получаются последовательности $a'$, $b'$ и $c'$. Затем формируются последовательности *x* и y.<br>$a'_i = \frac{a_i}{\sum_{i=1}^{n} a_i}$, $b'_i = \frac{b_i}{\sum_{i=1}^{n} b_i}$, $c'_i = \frac{c_i}{\sum_{i=1}^{n} c_i}$,<br>$x_i = a'_i + b'_i$.<br>$y_i = b'_i + c'_i$,<br>i = 1, 2, …, n |
| 10 | Даны три числовые последовательности *a*, *b* и *c*. Сформировать две новые последовательности *x* и y в соответствии со следующим правилом<br>$x_i = \max(\lvert a_i \rvert, \lvert b_i \rvert)$,<br>$y_i = \max(\lvert b_i \rvert, \lvert c_i \rvert)$,<br>i = 1, 2, …, n |
| 11 | Даны три числовые последовательности *a*, *b* и *c*. Сформировать две новые последовательности *x* и y. Формирование выполняется в два этапа. На первом этапе осуществляется нормировка исходных последовательностей *a*, *b* и *c*. В результате нормировки получаются последовательности $a'$, $b'$ и $c'$. Затем формируются последовательности *x* и y.<br>$a'_i = \frac{a_i}{\max(a) - \min(a)}$, $b'_i = \frac{b_i}{\max(b) - \min(b)}$, $c'_i = \frac{c_i}{\max(c) - \min(c)}$,<br>$x_i = a'_i + b'_i$.<br>$y_i = b'_i + c'_i$,<br>i = 1, 2, …, n |
| 12 | Даны три числовые последовательности *a*, *b* и *c*. Сформировать две новые последовательности *x* и y. Формирование выполняется в два этапа. На первом этапе осуществляется нормировка исходных последовательностей *a*, *b* и *c*. В результате нормировки получаются последовательности $a'$, $b'$ и $c'$. Затем формируются последовательности *x* и y. |

| | |
|---|---|
| | $a_i' = \frac{a_i}{\sum_{i=1}^{n} \lvert a_i \rvert}$, $b_i' = \frac{b_i}{\sum_{i=1}^{n} \lvert b_i \rvert}$, $c_i' = \frac{c_i}{\sum_{i=1}^{n} \lvert c_i \rvert}$,<br>$x_i = a_i' - b_i'$.<br>$y_i = b_i' - c_i'$, |
| 13 | Даны три числовые последовательности *a*, *b* и *c*. Сформировать две новые последовательности *x* и y в соответствии со следующим правилом<br>$x_i = a_i + b_{n-i+1}$,<br>$y_i = b_i + c_{n-i+1}$,<br>i = 1, 2, …, n |
| 14 | Даны три числовые последовательности *a*, *b* и *c*. Сформировать две новые последовательности *x* и y в соответствии со следующим правилом<br>$x_i = \min(a_{n-i+1}, b_i, r)$,<br>$y_i = \min(b_{n-i+1}, c_i, r)$,<br>i = 1, 2, …, n.;<br>r – произвольное число |
| 15 | Даны три числовые последовательности *a*, *b* и *c*. Сформировать две новые последовательности *x* и y в соответствии со следующим правилом<br>$x_i = \min(a_{n-i+1}, b_i)$,<br>$y_i = \min(b_{n-i+1}, c_i)$,<br>i = 1, 2, …, n.;<br>r – произвольное число |
| 16 | Даны три числовые последовательности a, b и c. Сформировать две новые последовательности x и y в соответствии со следующим правилом<br>$x_i = \max(a_{n-i+1}, b_i, r)$,<br>$y_i = \max(b_{n-i+1}, c_i, r)$,<br>i = 1, 2, …, n.;<br>r – произвольное число |
| 17 | Даны три числовые последовательности a, b и c. Сформировать две новые последовательности x и y в соответствии со следующим правилом<br>$x_i = a_i * b_i - \sqrt{\sum_{i=1}^{n} \lvert a_i * b_i \rvert}$<br>$y_i = b_i * c_i - \sqrt{\sum_{i=1}^{n} \lvert b_i * c_i \rvert}$,<br>i = 1, 2, …, n. |
| 18 | Даны три числовые последовательности a, b и c. Сформировать две новые последовательности x и y в соответствии со следующими правилами<br>$x_i = a_i$, *если* $a_i > 0$ *и* $b_i > 0$,<br>$x_i = b_i$, *если* $a_i < 0$ *и* $b_i < 0$,<br>0 *в оставшихся случаях* |

| | |
|---|---|
| | $y_i = \begin{cases} b_i, & \text{если } b_i > 0 \text{ и } c_i > 0, \\ c_i, & \text{если } b_i < 0 \text{ и } c_i < 0, \\ 0 & \text{в оставшихся случаях} \end{cases}$<br>i = 1, 2, …, n. |
| 19 | Даны три числовые последовательности a, b и c. Сформировать две новые последовательности x и y в соответствии со следующим правилом<br>$x_i = a_i - b_i - \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2}$<br>$y_i = b_i - c_i - \sqrt{\sum_{i=1}^{n} (b_i - c_i)^2}$ ,<br>i = 1, 2, …, n. |
| 20 | Даны три числовые последовательности a, b и c. Сформировать две новые последовательности x и y в соответствии со следующими правилами<br>$x_i = \begin{cases} a_i, & \text{если } a_i > 0 \text{ и } b_i < 0, \\ b_i, & \text{если } a_i < 0 \text{ и } b_i > 0, \\ 0 & \text{в оставшихся случаях} \end{cases}$<br>$y_i = \begin{cases} b_i, & \text{если } b_i > 0 \text{ и } c_i < 0, \\ c_i, & \text{если } b_i < 0 \text{ и } c_i > 0, \\ 0 & \text{в оставшихся случаях} \end{cases}$<br>i = 1, 2, …, n. |

## *Методические указания по выполнению лабораторной работы*

В общем случае программа, написанная на языке СИ, состоит из основной программы -- функции **main()** и некоторой совокупности подпрограмм, которые в СИ называются функциями. Функций позволяют сделать структуру программы более простой и ясной.

Основная программа должна быть по возможности компактной. Это повышает читабельность программы. Ознакомившись с функцией **main()**, читающий должен получить общее представление о программе. Наличие в функции **main()** цикла должно побудить программиста рассмотреть вопрос о целесообразности использования подпрограммы.

При решении задач настоящей работы целесообразно использовать подпрограммы при выполнении следующих операций:

- ввод исходных числовых последовательностей,
- формирование выходных числовых последовательностей,
- вывод результатов вычислений.

В задачах настоящей лабораторной работы следует использовать массивы. Применение массивов может быть оправдано следующими обстоятельствами:

- элементы некоторых последовательностей используются при формировании более чем одной последовательности,

- целесообразностью при выводе результатов вычислений наряду с выводом выходных последовательностей выводить и входные последовательности.

Необходимо рассмотреть вопрос о количестве массивов, которые следует выделить для хранения данных. При этом возможны следующие варианты.

1. Расточительный вариант. Для каждой входной и выходной последовательности отводится свой массив.
2. Экономный вариант. Отводится только три массива. Два массива отводятся для хранения входных последовательностей и один – для хранения выходной последовательности.

Сравнение таких вариантов и выбор рабочего варианта оставляем студенту в качестве самостоятельной работы.

Входные данные в задачах настоящей лабораторной работы состоят из числовых последовательностей и величины $n$ – количества элементов во входных последовательностях. Значение величины $n$ должно вводиться в основной части программы, а для ввода элементов числовых последовательностей следует применять функцию.

Возможны два варианта построения функции для ввода числовых последовательностей. В первом из вариантов функция возвращает обе числовые последовательности, и для выполнения ввода последовательностей необходимо написать только один вызов функции. Во втором варианте функция возвращает только одну числовую последовательность, и для ввода всех числовых последовательностей необходимо написать несколько вызовов функции.

Следует отдать предпочтение второму варианту организации функции. Дело в том, что функция, которая “занимается” вводом только одной числовой последовательности, может потребоваться чаще, чем функция, используемая для ввода сразу нескольких массивов. Общее правило таково, что функция должна выполнять по возможности небольшую работу. Чем меньше эта работа, тем в общем случае более возможностей для ее применения.

Эти же соображения могут быть положены в основу разработки функции для формирования выходных числовых последовательностей. Такая функция должна возвращать только одну числовую последовательность.

### *Контрольные вопросы*

1. Дайте определение массива.
2. Для каких целей используют массив?
3. Что такое размерность массива?
4. Как объявить в программе массив?
5. К какому типу могут относиться индексы массива?
6. Какие операции допустимы с переменными, имеющими тип массива?

7. Может ли функция языка СИ возвращать значения, имеющие тип массива?
8. Как организовать ввод (вывод) элементов одномерного массива?
9. Использование одномерных массивов в качестве параметров функций.

## Лабораторная работа 8

# Обработка двумерных массивов

### *Постановка задачи*

В настоящей лабораторной работе необходимо выполнить заданную обработку числового двумерного массива, имеющего произвольное количество строк(*N*) и столбцов(*M*). В каждом из вариантов содержатся ограничения на максимальные значения величин *N* и *M*. Элементы массива должны вводиться с клавиатуры и для их ввода следует разработать функцию. Обработка, предусмотренная заданием, и вывод нескалярного результата выполняются с помощью функций.

### *Варианты заданий*

Варианты заданий приведены в таблице

| N | Задание |
|---|---|
| 1 | Вычислить сумму чисел в каждой строке. N <= 10, M <= 10 |
| 2 | Вычислить произведение чисел в каждом столбце. N <= 10, M <= 5 |
| 3 | Вычислить количество положительных чисел в каждом столбце. N <= 7, M <= 8 |
| 4 | Вычислить сумму чисел для каждого столбца, удовлетворяющих условию $x_{i,j} > a$. Здесь $a$- произвольная величина. $N$ <= 5, $M$ <= 5 |
| 5 | Вычислить произведение чисел для каждого столбца, удовлетворяющих условию $x_{i,j} < b$. Здесь $b$ – произвольная величина. $N$ <= 5, $M$ <= 10 |
| 6 | Вычислить значение наибольшего по модулю элемента для каждой строки массива. $N$ <= 5, $M$ <= 5 |

| | |
|---|---|
| 7 | Вычислить значение наименьшего элемента для каждого столбца массива. $N <= 7$, $M <= 5$ |
| 8 | Вычислить для каждой строки значение наименьшего элемента и его индекс. $N <= 10$, $M <= 5$ |
| 9 | Вычислить значение наибольшего по модулю элемента и его индекс для каждого столбца массива. $N <= 10$, $M <= 10$ |
| 10 | Вычислить сумму отрицательных чисел в каждой строке. N <= 10, M <= 10 |
| 11 | Вычислить среднее значение чисел в каждой строке массива. $N <= 5$, $M <= 5$ |
| 12 | Вычислить среднее значение чисел в каждом столбце массива. $N <= 5$, $M <= 5$ |
| 13 | Вычислить для каждой строки массива отклонение ее элементов от среднего значения для этой строки. $N <= 10$, $M <= 10$ |
| 14 | Вычислить для каждого столбца массива отклонение его элементов от среднего значения для этого столбца. $N <= 12$, $M <= 10$ |
| 15 | Вычислить для каждого столбца значение разности между наибольшим и наименьшим элементами. $N <= 5$, $M <= 10$ |
| 16 | Вычислить для каждой строки значение разности между наибольшим и наименьшим элементами. $N <= 15$, $M <= 10$ |
| 17 | Вычислить сумму элементов для каждого столбца. Поменять местами столбцы с максимальным и минимальным значением суммы. $N <= 5$, $M <= 10$ |
| 18 | Вычислить сумму элементов для каждой строки. Поменять местами строки с максимальным и минимальным значением суммы. $N <= 5$, $M <= 10$ |
| 19 | Вычислить произведение элементов для каждого столбца. Поменять местами столбцы с максимальным и минимальным значением произведения. $N <= 4$, $M <= 5$ |

| | |
|---|---|
| 20 | Вычислить произведение элементов для каждой строки. Поменять местами строки с максимальным и минимальным значением произведения. *N* <= 5, *M* <= 10 |
| 21 | Вычислить сумму элементов для каждой строки. Вывести на экран строки с максимальным и минимальным значениями суммы. *N* <= 5, *M* <= 10 |
| 22 | Вычислить сумму элементов матрицы, которые расположены выше главной диагонали. *N <= 5, M <= 5* |
| 23 | Выполнить нормировку элементов каждой строки матрицы, поделив ее элементы на значение максимального элемента *N* <= 4, *M* <= 5 |
| 24 | Выполнить нормировку элементов каждого столбца матрицы, поделив его элементы на значение максимального элемента *N* <= 4, *M* <= 5 |

### *Методические указания по выполнению лабораторной работы*

При выполнении лабораторной работы рекомендуется учитывать следующее.

1. Средством обращения к элементам двумерного массива (матрицы) является переменная с двумя индексами (например *x*[*i*][ *j*]). Первый индекс (в данном примере *i*) – номер строки, а второй индекс (в данном примере *j*) – номер столбца.
2. Для обработки матрицы, как правило, следует использовать вложенные циклы.
3. При обработке матрицы по строкам внешний цикл должен изменять номер строки, а внутренний цикл – номер столбца. Если в цикле средством обращения к элементам является переменная *y*[*k*][*l*], тогда в операторе цикла *for* внешнего цикла следует использовать переменную *k,* а в операторе цикла *for* внутреннего цикла – переменную *l.*
4. При обработке матрицы по столбцам внешний цикл должен изменять номер столбца, а внутренний цикл – номер строки. Если в цикле средством обращения к элементам является переменная *y*[*k*][*l*], тогда в операторе цикла *for* внешнего цикла следует использовать переменную *l,* а в операторе цикла *for* внутреннего цикла – переменную k.

### *Контрольные вопросы*

1. Какие массивы называются двумерными?
2. Как объявить в программе двумерный массив?
3. Каким образом можно в программе организовать ввод (вывод) элементов двумерных массивов?
4. Каким образом в программе организовать обработку массива по строкам (столбцам)?

## Лабораторная работа № 9
## Работа со строками

### *Постановка задачи*

Имеется текст, состоящий из n(n <= 20) строк, который вводится с клавиатуры. Длина каждой строки не превосходит 128 символов. В каждой строке содержится не менее двух слов. Количество слов в строке не более 20. Отдельные слова отделяются друг от друга одним или более пробелами. Необходимо выполнить заданную обработку введенного текста. Вид обработки зависит от варианта задания и определяется данными табл. 9.1. Вывод результатов обработки текста должен выполняться только после завершения его ввода. При обработке текста необходимо учитывать возможность наличия во введенной строке ведущих и завершающих пробелов. Количество пробелов во введенном и обработанном тексте может не совпадать. При разработке программы для решения поставленной задачи необходимо в максимальной степени использовать функции.

### *Варианты заданий*

Варианты заданий приведены в табл. 9. 1

Таблица 9.1

| N | Задание |
|---|---|
| 1 | Удалить из каждой строки слова с четными номерами. |
| 2 | Записать в конец каждой строки количество содержащихся в ней слов. |
| 3 | Удалить из каждой строки слова с нечетными номерами. |
| 4 | Записать в конец каждой строки количество содержащихся в ней гласных букв. |
| 5 | Записать в конец каждой строки текста количество содержащихся в ней согласных букв. |
| 6 | Удалить из каждой строки два первых слова. |
| 7 | Удалить из каждой строки два последних слова. |
| 8 | Удалить из каждой строки последнее слово. |
| 9 | Удалить из каждой строки первое слово. |
| 10 | Перенести первое слово каждой строки в ее конец. |
| 11 | Перенести последнее слово каждой строки в ее начало. |

| | |
|---|---|
| 12 | Поменять местами первое и последнее слово в каждой строке. |
| 13 | Поменять местами первое и второе слово в каждой строке. |
| 14 | Поменять местами последнее и предпоследнее слово в каждой строке. |
| 15 | Удалить из каждой строки ее второе слово. |
| 16 | Перенести в конец каждой строки ее второе слово. |
| 17 | Удалить из каждой строки ее предпоследнее слово. |
| 18 | Удалить из каждой строки все слова, длина которых `l` удовлетворяет отношению `l > Lmin..` |
| 19 | Удалить из каждой строки все слова, длина которых `l` удовлетворяет отношению `l < Lmax..` |
| 20 | Удалить из каждой строки ее второе слово при условии, что длина слова `l` удовлетворяет отношению `l < Lmax..` |
| 21 | Записать в начало каждой строки количество содержащихся в ней слов. |
| 22 | Записать в начало каждой строки количество содержащихся в ней гласных слов. |
| 23 | Записать в начало каждой строки количество содержащихся в ней согласных букв. |
| 24 | Удалить из каждой четной строки первое и последние слова. |
| 25 | Выполнить лексикографическую сортировку строк. |
| 26 | Выполнить обратную лексикографическую сортировку строк. |

### *Методические указания к лабораторной работе*

Приведем ряд соображений, использование которых может быть полезным при разработке программы.

1. Для отделения вывода результатов обработки текста от ввода исходных данных целесообразно использовать массив строк.
2. Ввод текста удобно выполнить с помощью функции.
3. Обработка текста может быть выполнена с помощью функций.
4. Возможны два способа обработки исходного текста. В первом из них текст рассматривается как двумерный массив символов. Во втором способе используется буферная строка (назовем ее `str`). В `str` из массива строк копируется очередная строка текста. Затем `str` обрабатывается как одномерный массив символов. Необходимо сравнить эти два способа и выбрать лучший из них.
5. Обработку очередной строки массива можно начать с удаления пробелов. Дело заключается в том, в зависимости от варианта решаемой задачи может оказаться полезным удаление начальных или конечных (или начальных и конечных) пробелов в обрабатываемой строке. Такую обработку предпочтительнее выполнить с помощью подпрограммы. Пример такой подпрограммы приведен в разделе 9.5 (функция trim).
6. Для выполнения основной обработки может оказаться полезной разработка подпрограммы обработки очередной строки текста (функция form_arr_word).

При выполнении лабораторной работы рекомендуется использовать стандартные библиотечные подпрограммы: strlen, strcpy, strcat, strtod и др. Основной операцией может оказаться операция выделения в строке отдельных слов. Ниже приводится пример подпрограммы, в которой реализуется операция разделения строки на слова.

## *Справочные материалы*

### Понятие о строках

Строкой в программировании называется последовательность символов переменной длины. В языке Си отсутствует тип данных, предназначенный для работы со строками. Для этой цели используются массивы символов. Для управления длиной строки, находящейся в таком массиве, в языке Си предусмотрено специальное соглашение. Существо этого соглашения состоит в том, что последовательность символов, из которых состоит строка символов, должна заканчиваться специальным символом (маркером). В качестве такого маркера используется так называемый нуль-символ (\0). Нуль-символ – это символ языка Си, код которого равен нулю. Следует учитывать, нуль-символ выполняет чисто служебные функции. Таким образом, в языке Си строка хранится в символьном массиве в виде последовательности символов, которая заканчивается нуль-символом.

### Строковые литералы

В языке Си строковый литерал – последовательность символов, заключенная в кавычки. Например:

```
"Hello, world"
```

В конец строкового литерала компилятор автоматически добавляет нуль-символ. Возможны две точки зрения на строковый литерал. Во-первых, строковый литерал можно рассматривать как массив символов, который заканчивается нуль-символом. Во-вторых, строковый литерал можно рассматривать как указатель на его первый символ. Последнее позволяет использовать строковый литерал в любом контексте, в котором ожидается использование указателей.

### Символические строковые константы

Возможны три вида строковых констант. Во-первых, символическая строковая константа может быть представлена в виде символического строкового литерала. Такой подход предполагает использование директивы препроцессора define. Например:

```
# define GREETING "Hello, world"
```

Во-вторых, строковый литерал может использоваться для инициализации константного указателя. Например:

```
const char* const ptr_greeting = "Hello, world";
```

В определении указателя ptr_greeting дважды используется зарезервированное слово **const**. Первое из этих слов, с которого и начинается

определение, запрещает изменять символы строкового литерала, а втрое запрещает изменять значение самого указателя.

Наконец, строковый литерал может использоваться для инициализации константной строковой переменной. Например:

```
const char greeting[] = "Hello, world";
```

### Строковые переменные

Строковая переменная – массив символов, содержащий последовательность символов, которая заканчивается нуль-символом. Рассмотрим следующий программный код:

```
#define MAXLENGTH 129
/* ... */
char str1[MAXLENGTH];
```

Для переменной str1 выделяется память объемом в 129 байт. Потенциально такая переменная может работать со строками, длина которых не превосходит 128 символов. Особенность рассматриваемой переменной состоит в отсутствии явной инициализации. Неявная инициализация этой переменной будет иметь место в том случае, когда ее определение будет находиться вне тела функции. В этом случае массив str1 будет инициализирован нулями, а сама переменная может рассматриваться как пустая строка. В противном случае переменная str1 будет содержать "мусор" и не может использоваться в качестве строки. Это делает целесообразным явно инициализировать строки при их объявлении.

### Инициализация строковых переменных

Возможны два способа инициализации строковой переменной при ее определении:

- инициализация строковым литералом,
- инициализация массивом символов.

Приведем примеры явной инициализации строковых переменных при их определении.

**Пример 1.**

```
#define MAXLENGTH 129
char str1[MAXLENGTH] = "Hello";
```

В этом примере для хранения элементов массива str1 предусмотрено выделение 129 байт памяти, в начало которой будет скопирован строковый литерал. Нуль-символ (\0) в массив str1 будет занесен автоматически.

**Пример 2.**

```
#define MAXLENGTH 129
char str2[MAXLENGTH] = " 'H', 'e', 'l', 'l', 'o', '\0'";
```

Второй пример отличается от первого только способом инициализации. Здесь для инициализации используется массив символов. Нуль-символ в этом случае должен быть включен в число инициализаторов.

Обычно используется первый способ инициализации.

Следует отметить, что инициализировать строковую переменную можно только в момент определения. Следующий код содержит ошибку:

**Пример 3**.

```
#define MAXLENGTH 129
char str3[MAXLENGTH];
str3 = “Hello”; /* Синтаксическая ошибка*/
```

Здесь имеет место попытка изменить значение константного указателя, которым является str3.

### Операции со строковыми переменными

Со строковыми переменными можно выполнять только поэлементные операции. Все остальные операции (ввод и вывод строк, их копирование, объединение, сравнение и др.) следует выполнять с помощью библиотечных функций или разрабатывать собственные функции пользователя.

Для работы с библиотечными функциями следует подключить стандартный заголовочный файл string.h.

### Вывод строк

Для ввода строк с клавиатуры можно воспользоваться следующими библиотечными функциями:

- `printf()`,
- `puts()`,
- `fputs()`.

Начнем рассмотрение с функции printf().

### Вывод строк с помощью функции printf()

Строки могут выводиться с помощью библиотечной функции printf(). Для вывода строкового литерала достаточно поместить его текст в форматной строке вызова рассматриваемой функции. Пусть необходимо вывести стандартное приветствие. Например, это можно выполнить с помощью следующего вызова функции printf():

```
printf(“Hello, world”);
```

В более сложных случаях следует при выводе строк использовать спецификацию типа s. В качестве примера приведем вывод заголовка таблицы:

```
printf(“%5s%10s%10s”, “НОМЕР”, “АРГУМЕНТ”, “ФУНКЦИЯ”);
```

Здесь также необходимо выводить строковые литералы. Однако форматирование вывода лучше предоставить компилятору, а не выполнять вручную в форматной строке.

Теперь приведем пример, в котором будет выводиться строковая переменная:

```
#include<stdio.h>
#define MAXLENGTH 129
int main(void)
{
    char st[MAXLENGTH] = “”;
    /* Работа со строкой st  */
    printf(“%s\n”, st);
}
```

### Функция puts()

Объявление функции `puts()` имеет следующий вид

```
#include<stdio.h>
int puts(const char* str);
```

Функция `puts()` выводит на экран дисплея строку, на которую установлен указатель `str`. Нуль – символ этой строки преобразуется в символ новой строки, который выводится на экран. Последнее приводит к тому, что после вывода строки `str` курсор перейдет на начало новой строки экрана.

При успешном выполнении функция `puts()` возвращает неотрицательное число, а в случае сбоя – значение `EOF`.

Рекомендуется использовать функцию `puts()` вместо `printf()` в тех случаях, когда необходимо вывести отдельное сообщение для пользователя, которое не сопровождается выводом и вводом данных.

Приведем пример применения функции `puts()`.

```
/* Демонстрация вывода строк с применением функции puts()*/
/*                    Файл puts.c                          */
#include<stdio.h>
int main(void)
{
    char*  msg1   = "Не";
    char*  msg2   = "применяйте";
    char*  msg3   = "функцию";
    char*  msg4   = "gets()";
    char*  msg5   = "в коммерческих";
    char*  msg6   = "приложениях!!!";
    puts(msg1);
    puts(msg2);
    puts(msg3);
    puts(msg4);
    puts(msg5);
    puts(msg6);
    return 0;
  }
```

Результат выполнения программы будет иметь следующий вид:

```
Не
применяйте
функцию
gets()
в коммерческих
приложениях!!!
```

### Функция fputs()

Функция fputs() предназначена для работы с файлами. Она не имеет особых преимуществ перед функцией puts(), которая используется для консольного вывода строк. Ее целесообразно применять совместно с

функцией ввода строк fgets(). Функция fgets() имеет очевидные преимущества перед функцией gets(), специально предназначенной для консольного ввода строк. Это и делает оправданным рассмотрение этой функции.

Прототип рассматриваемой функции имеет следующий вид:

```
#include<stdio.h>
int fputs(const char*s, FILE* stream);
```

Рассматриваемая функция имеет один дополнительный параметр, которого нет у функции puts().Этот дополнительный параметр (FILE* stream) при использовании функции fgets() определяет файл, с которым должна работать эта функция. Для консольного ввода достаточно в ее вызове в качестве параметра stream использовать имя стандартного потока, предназначенного для работы с клавиатурой (stdout).

Эта функция (в том случае, когда она вызвана со вторым параметром, равным stdout), выводит на экран дисплея строку символов, на которую установлен указатель s. При этом нуль-символ, которым заканчивается выводимая строка отбрасывается. В отличие от puts() функция fputs() не дополняет выводимую строку символом “новая строка”. Если во время вывода строки имеет место ошибка, то функция fputs() вернет значение EOF. При успешном завершении рассматриваемой функции возвращается неотрицательное число.

### Ввод строк

Для ввода строк с клавиатуры можно воспользоваться следующими библиотечными функциями:

- `scanf()`,
- `gets()`,
- `fgets()`.

Начнем рассмотрение с функции scanf().

#### Функция scanf()

Особенность функции scanf() состоит в том, что с ее помощью нельзя вводить строки, содержащие пробелы. Эта функция может использоваться только для ввода отдельных слов. Приведем пример.

Пример 4.

```
#define MAXLENGTH 129
#include<stdio.h>

int main(void)
{
   char st[MAXLENGTH];
   prinf(“Enter a string: ”);
   scanf(“%s”, st);
   printf(“%s”, st);
   /*   */
```

```
   return 0;
}
```

Следует учитывать, что переменная st является указателем и при ее передаче в функцию scanf() оператор & (оператор взятия адреса) не нужен. Протокол работы с программой, приведенной выше, будет иметь следующий вид (с помощью подчеркивания показан ввод пользователя):

```
Enter a string: Hello, world<Enter>
Hello,
```

Из протокола видно, что введенным оказалось только первое слово (Hello,).

Другой особенностью функции scanf() является возможность ограничения количества вводимых символов. Для этого в спецификации преобразования функции следует использовать дополнительный параметр, как это показано на примере, приводимом ниже:

Пример 5.

```
#define MAXLENGTH 129
#include<stdio.h>

int main(void)
{
   char st[MAXLENGTH];
   prinf("Enter a string: ");
   scanf("%5s", buffer);
   printf("%s", buffer);
   /*   */
   return 0;
}
```

Протокол работы с программой, приведенной выше, будет иметь следующий вид:

```
Enter a string: 1234567890<Enter>
12345
```

В этом примере спецификация преобразования содержит дополнительный параметр в виде числового литерала (5). В рассматриваемом случае воспринимается не более 5 вводимых символов.

Отметим, что при организации ввода строк программист обязан предусмотреть выделение памяти для вводимой строки. Ниже приведен пример, в котором отсутствует выделение памяти для вводимой строки.

Пример 6.

```
#include<stdio.h>

int main(void)
{
   char *s;
   prinf("Enter a string: ");
   scanf("%5s", s);  /* Ошибка*/
   printf("%s", s);
   /*   */
   return 0;
}
```

Ошибка состоит в том, что используется неинициализированный указатель s.

### Функция gets()

Прототип этой функции имеет следующий вид:

```
#include<stdio.h>
char* gets(char* str);
```

Рассматриваемая функция читает символы, вводимые с клавиатуры в символьный массив, на который установлен указатель `str`. Это чтение выполняется до тех пор, пока не встретится либо символ “новая строка” (`\n`), либо конец файла. После записи последнего прочитанного символа в массив `str` добавляется нуль – символ. Если встречается символ “новая строка”, то он отбрасывается. Если выполнение функции завершено успешно, то она возвращает указатель `str`. Функция `gets()` вернет значение `NULL` в том случае, когда при достижении конца файла ни один из символов не оказался прочитанным, причем содержимое массива, на который установлен указатель `str`, останется без изменения. Значение `NULL` является возвращаемым значением и в том случае, когда будет обнаружена ошибка чтения, но в этом случае содержимое массива считается не определенным. Приведем пример применения функции `gets()`. В этом примере вначале запрашивается имя пользователя, а затем компьютер выводит приветствие.

```
/* Greeting.c */
#include<stdio.h>
#include<string.h>
#define MAX 81

int main(void)
{
   char name[MAX];
   char out[MAX] = “Привет, Вам ”;
   puts(“Введите Ваше имя”);
   gets(name);
   printf(“Привет, Вам %s”, name);
   puts(out);
   return 0;
}
```

```
Протокол работы с программой greeting имеет следующий вид
Введите Ваше имя
Иван
Привет, Вам Иван
```

Существенным недостатком рассматриваемой функции является возможность переполнения массива, на который установлен указатель `str`. В связи с этим функцию `gets()` не рекомендуют использовать в коммерческих приложениях. Для этих целей можно воспользоваться функцией `fgets()`.

### Использование функции fgets() для консольного ввода строк

В связи с тем, что функция gets() пользуется плохой репутацией, в качестве альтернативы этой функции предлагается использовать функцию файлового ввода-вывода fgets().Для обсуждения возможности использования этой функции для консольного ввода приведем прототип функции fgets():

```
#include<stdio.h>
char*  fgets(char* str, int n, FILE* stream);
```

Рассматриваемая функция имеет два дополнительных параметра, которые отсутствуют у функции gets(). Первый из дополнительных параметров (**int** n) служит для ограничения количества символов, которые могут быть прочитаны в массив str из буфера клавиатуры. Второй дополнительный параметр (FILE* stream) при использовании функции fgets() определяет файл, с которым должна работать эта функция. Для консольного ввода достаточно в ее вызове в качестве параметра stream взять имя стандартного потока, предназначенного для работы с клавиатурой (stdin).

Функция fgets() в форме, предназначенной для ввода с клавиатуры, позволяет записать в массив, на который указывает указатель str, не более n – 1 символа. Ввод прекращается в следующих трех ситуациях:

1. встретился символ новая строка,
2. встретился конец файла.
3. условия 1 и 2 не выполнялись, но прочитан n – 1 символ,

После чтения последнего символа из потока в строку str, строка дополняется нуль символом. Если при чтении встретился символ новая строка (условие 1), то он записывается в строку str и нуль символ записывается за ним. Отсюда следует, что символ новая строка может записываться, а может и не записываться в строку, на которую установлен указатель str.

В случае успешного завершения функция fgets() вернет указатель на строку str. Если прочитан конец файла, а ни один символ не был введен, то содержимое массива оказывается неизменным, а функция вернет значение NULL. Если во время ввода имела место ошибка, то функция вернет значение NULL, а содержимое массива str оказывается неопределенным. Приведем пример.

```
#include<stdio.h>
#include<string.h>
#define MAXSIZE 81

int main(void)
{
    char buf[MAXSIZE];
    char* s = NULL;
    fgets(buf, sizeof(buf), stdin);
    s = strchr(buf, '\n');/* Ищем символ ‘\n’ в прочитанной
                             строке */
    if(s != NULL)
```

```
        *s = '\0';              /* Запись символа '\0' вместо
                                   символа '\n' */
    return 0;
}
```

С целью приблизить работу функции `fgets()` к работе функции `gets()`, которую она призвана заменить, в рассматриваемом примере добавлен программный код, удаляющий из массива, используемого для ввода строки (`buf`), символ новой строки (`\n`). Для этой цели используется функция `strchr()` и инструкция `if`.

## Стандартные функции для обработки строк

### Функция strlen

Эта функция предназначена для определения длины строки. Ее прототип имеет следующий вид:

```
#include<string.h>
size_t strlen(const char* str);
```

Тип size_t является разновидностью целочисленного типа. Функция strlen() возвращает длину строки, на которую установлен указатель str, причем строка должна заканчиваться “нуль – символом”. “Нуль – символ” во время определения длины строки не учитывается. Пример применения функции strlen().

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

int main(void)
{
    char str[81];
        printf("Введите строку:");
        gets(str);
    printf("Длина введенной строки =%d", strlen(str));
    return 0;
}
```

```
Протокол работы с программой имеет следующий вид.
Введите строку:Это строка

Длина введенной строки=10
```

### Функции strcpy() и strncpy()

Прежде всего, следует отметить, что в языке См отсутствуют встроенные средства для копирования строк. Применять для этой цели оператор присваивания нельзя. При компиляции приведенного ниже кода будет выдано сообщение об ошибке.

```
/*       Программный код, содержащий ошибку     */
char str1[30] = "Hello";
char str2[30];
str2 = str1;   /* Недопустимый код, т.к. str2 – константный
  указатель   */
```

Для копирования строк в языке Си следует использовать библиотечные функции strcpy() и strncpy(). Обращает на себя похожесть имен этих функций. В соответствии с принципом образования имен, принятым в библиотеке string это означает, что функция strncpy() имеет дополнительный параметр n.

Прототипы функций strcpy() и strncpy() имеют следующий вид:

```
#include<string.h>
char* strcpy(char* out_str, const char* in_str);
char* strncpy(char* out_str, const char* in_str, size_t n);
```

Обе функции (strcpy() и strncpy) копируют содержимое строки in_str в строку out_str. Параметр in_str должен указывать на строку, которая заканчивается нуль - символом. До вызова рассматриваемых функций необходимо выделить память для хранения новой строки. Функции strcpy() и strncpy() эту память не выделяют. Обе функции возвращают значение указателя out_str. Обе функции заканчивают копирование в том случае, когда в строке str_in встречается нуль – символ. Функция strncpy() выполняет копирование более осторожным образом. Это связано с наличием у этой функции третьего параметра (параметр n), который ограничивает количество копируемых символов. Количество символов, которые могут быть скопированы функцией strncpy() не может быть больше n. Заметим, что в предельном случае, когда в скопированных n символах строки in_str не встретился нуль - символ, то выходная строка не будет заканчиваться нуль – символом. Если массивы in_str и out_str перекрываются поведение функции strcpy() не определено.

В следующем фрагменте кода строка Hello копируется в строку `str`.

```
char str[81];
strcpy(str, "Hello");
```

Типичной ошибкой при работе с функцией strcpy() является передача ей неправильного указателя на строку str_in. Например, некорректным оказывается следующий фрагмент кода:

```
void foo()
{
   char str1[25] = "Hello";
   char* str2;
   strcpy(str1, str2);
   /* другой код          */
}
```

Ошибка в приведенном выше коде состоит в отсутствии инициализации указателя str2.

### Функции strcat() и strncat()

Для целей объединения строк можно использовать две функции: strcat() и strncat(). Вторая из этих функций (strncat()) в отличие от первой ограничивает количество символов, добавляемых в память, в которой происходит объединение строк.

Объявления рассматриваемых функций имеют следующий вид:

```
#include<string.h>
char* strcat(char* out_str, const char* in_str);
char* strncat(char* out_str, const char* in_str,
              size_t n);
```

Функция strcat() присоединяет содержимое строки in_str к строке out_str. Параметр in_str должен указывать на строку, которая заканчивается нуль - символом. Конечный “нуль – символ”, первоначально завершающий строку out_str, перезаписывается первым символом строки in_str. Функция strcat() возвращает значение указателя out_str. Если массивы in_str и out_str перекрываются поведение функции strcat() не определено.

В следующей программе две строки читаются с клавиатуры, затем объединенная строка выводится на экран дисплея.

```
#include<stdio.h>
#include<string.h>

int main(void)
{
   char buf1[64], buf2[64];
   gets(buf1);
   gets(buf2);
   strcat(buf1, buf2);
   printf(buf1);
   return 0;
}
```

Перейдем к рассмотрению функции strncat(). Эта функция присоединяет не более n символов строки in_str в конец строки out_str. Если “нуль – символ” строки in_str достигнут раньше, чем будут прочитаны n символов строки in_str, то этот символ копируется, и процесс присоединения заканчивается. Если среди скопированных n символов не встретился нуль – символ, то он добавляется в строку in_str и процесс присоединения на этом заканчивается. В этом случае в выходную строку записывается n + 1 символ. Если в вызове функции n или отрицательно, то функция не изменяет выходную строку out_str.

### Функция strcmp()

Объявление этой функции имеет следующий вид

```
#include<string.h>
int strcmp(const char* in_str1, const char* in_str2);
```

Функция strcmp() выполняет так называемое лексографическое сравнение строк. Функция возвращает нулевое значение, если строки

совпадают. Функция возвращает положительное значение, если строка in_str1 > in_str2. Наконец, функция возвращает отрицательное значение, если in_str1 < in_str2. Строка in_str1 считается больше строки in_str2, если первый несовпадающий ее символ имеет код, превышающий код соответствующего ему символа строки in_str2.

Ниже приводится пример применения функции strcmp().

```
#include<stdio.h>
#include<string.h>

int main(void)
{
   char name[32] = "Tom";
   /* Выводится положительное число */
   printf("%d\n", strcmp(name, "Alic");   /*

   /* Выводится отрицательное число */
   printf("%d\n", strcmp(name, "Victor");

   /*        Выводится нуль          */
   printf("%d\n", strcmp(name, "Tom");
   return 0;
}
```

### Функция strtok()

Для разделения строки на лексемы можно использовать функцию strtok(). Объявление этой функции имеет следующий вид:

```
#include<string.h>
char* strtok(char* str1, const char* str2);
```

Последовательные вызовы функции strtok() можно использовать для разбиения строки, адресуемой указателем str1, на цепочку лексем, завершаемых символом нуль - символом (\0). Символы, образующие строку, адресуемую параметром str2, представляют собой разделители, которые используются для выделения лексемы.

При первом вызове функции strtok() в качестве первого аргумента ей передается строка, которую следует разделить на лексемы. Вначале в строке, на которую указывает параметр str1, находится первый символ, отсутствующий в строке str2. Если такой символ в строке str2 не будет найден, то строка не будет разбиваться на лексемы, а функция strtok() вернет нулевой указатель. Если же такой символ будет обнаружен, то он станет началом первой лексемы. Затем процесс выполнения первого вызова будет продолжен. Функция strtok() будет искать в строке str1 любой символ, который содержится в строке разделителей str2. Если такой символ не будет найден, то текущая лексема расширяется до конца строки str1, и все последующие вызовы функции strtok() вернут нулевой указатель. В том случае, когда такой символ будет найден, то на его место будет записан нуль символ, завершающий лексему. Функция strtok() сохраняет значение

внутреннего указателя, ссылающегося на следующий символ, с которого должен начинаться поиск очередной лексемы.

Каждый последующий вызов функции strtok() должен принимать в качестве первого параметра нулевой указатель. Поиск очередной лексемы начинается с того символа, на который указывает внутренний указатель функции strtok(). Поиск выполняется по схеме, описанной выше. Функция strtok() возвращает указатель на первый символ лексемы или нулевой указатель.

Рассмотрим пример применения функции strtok(). Пусть дана строка, в которой слова отделяются одним или несколькими пробелами. Сформировать две строки. В первую из этих строк записать строки, длина которых len удовлетворяет условию len <= lmax, а во вторую - все оставшиеся слова исходной строки.

Программа состоит из двух функций. Основную работу по решению поставленной задачи выполняет функция select_word. Эта функция принимает четыре параметра. Первый из ее параметров in - указатель на исходную строку. Обратите внимание, на то обстоятельство, что функция select_word этот параметр не изменяет. Параметры out1 и out2 - указатели на выходные строки. Последний параметр lmax определяет критическое значение длины слова, которое используется для разделения исходной строки. Функция select_word использует внутренний буфер. Это позволяет обеспечить неизменность исходной строки.

```
#include<stdio.h>
#include<string.h>
void select_word(const char* in, char* out1, char* out2,
                 int len);

int main(void)
{
   char str[] = "1   22  333 4444 55555 666666 555555 4444
                 333 22\ 1";
   char out1[128], out2[128];
   select_word(str, out1, out2, 4);
   puts(out1);
   puts(out2);
   return 0;
}

/*  Выделение в строке in слов. Формирование выходных строк out1
    и out2. Строка out1 содержит те слова исходной строки in,
    которые удовлетворяют условию len <= lmax, а строка out1 -
    оставшиеся слова исходной строки.*/

void select_word(const char* in, char* out1, char* out2,
                 int lmax)
{
       char buf[128];
       char* p;
```

```
        strcpy(buf, in);
        strcpy(out1, "");
        strcpy(out2, "");
        p = strtok(buf, " ");
        while(p)
        {
            if(strlen(p) <= (unsigned)lmax)
            {
                strcat(out1, p);
                strcat(out1, " ");
            }
            else
            {
                strcat(out2, p);
                strcat(out2, " ");
            }
            p = strtok(NULL, " ");
        }
}
```

### Массивы строк

Массив строк в языке Си может быть представлен в виде двумерного массива, элементами которого являются символы. Приведем пример, Пусть в некоторой программе необходимо работа со строками, длина которых (без учета нуль-символа) не превышает 128, а количество строк не превосходит 25. В этом случае в программе можно предусмотреть следующее определение массива для работы со строками:

```
#define MAXLENGTH 129
#define MAXSIZE   25

int main(void)
{
    char arr_str[MAXSIZE][MAXLENGTH];
    /*     */
    return 0;
}
```

### Функции пользователя для обработки строк

#### Ввод строк

Приведем два варианта организации функций, предназначенных для ввода строк. В первом из этих вариантов функция предназначена для ввода заданного количества строк. Второй вариант предназначен для ввода произвольного количества строк.

```
/*
   Функция вводит заданное количество строк, определяемое
   параметром "n" в строковый массив strings
*/
void input_strings_1(char strings[][MAXLEN], int n)
```

```
{
   int i;
   for(i = 0; i < n; i++)
   {
       printf("?");
       gets(strings[i]);
   }
}

/*
   Функция позволяет ввести произвольное количество строк в
   строковый массив strings. Окончание работы определяется
   вводом конца файла. При работе в Dos и Windows для этого
   следует нажать клавиши Ctrl и Z (Ctrl + Z).
*/
int  input_strings_2(char strings[][MAXLEN])
{
   int count = 0;
   printf("?");
   while(gets(strings[count])){
     count++;
     printf("?");
   }
   return count;
}
```

### Вывод строк

```
/*
   Вывод на экран дисплея содержимого строкового массива
   strings. Количество выводимых строк определяется
   параметром "n"
*/
void print_strings(const char strings[][MAXLEN], int n)
{
    int i;
    printf("%-5s%-50s\n", "Ind", "String");
    for(i = 0; i < n; i++)
       printf("%-5d%-50s\n", i, strings[i]);
}
```

### Разделение строки на слова и запись полученных слов в строковый массив

```
/*
   Выделение в заданной строке from отдельных слов и запись
   их в строковый массив to. Функция возвращает количество
   слов, содержащихся в исходной строке
*/
int str_to_arr_words(const char* from, char to[][MAXLEN])
{
   char buf[MAXLEN];
```

```
   int count = 0;
   char* p = NULL;
   strcpy(buf, from);
   p = strtok(buf, " ");
   while(p)
   {
       strcpy(to[count++], p);
       p = strtok(NULL, " ");
   }
   return count;
}
```

**Объединение в строку слов, содержащихся в строковом массиве**

```
/*
   Построение из массива слов from строки to. Параметр "n"
   определяет количество слов, содержащихся в строковом
   массиве
*/
void arr_words_to_str(const char from[][MAXLEN], int n,
                      char *to)
{
   int i = 0;
   strcpy(to, "");
   for(; i < n; i++)
   {
      strcat(to, from[i]);
      strcat(to, " ");
   }
}
```

## *Контрольные вопросы*

1. Дайте, определение понятия строка.
2. Дайте определение строки, принятое в языке Си.
3. Что понимается под строковым литералом в языке Си?
4. Для хранения строки, содержащей m элементов необходимо иметь символьный массив из m + 1 элемента. Для каких целей используется дополнительный элемент?
5. Что произойдет, если попытаться в символьный массив записать строку, длина которой превышает размер массива?
6. В чем состоит различие между массивом символов и строкой в стиле языка Си.
7. Какие операции можно выполнять со строкой как со структурной переменной?
8. Какие ограничения существуют при работе со строкой как с массивом символов?

9. Ввести два предложения и вывести на экран самые длинные слова, общие для этих предложений. Если нужных слов нет, то вывести сообщение.
10. Ввести предложение, в котором слова разделены пробелами и запятыми. Вывести на экран дисплея те слова, которые являются обращениями слов в исходном предложении. Если таких слов нет, то вывести сообщение.
11. Вывести на экран дисплея пары слов с наименьшим расстоянием. Расстояние между словами определяется количеством позиций, в которых слова различаются. Например, расстояние между словами МАМА и ПАПА или МЫШКА и КОШКА равно двум.
12. Дан произвольный текст. Напечатать в алфавитном порядке все буквы, которые входят в этот текст по одному разу.
13. Распечатать введенное предложение, удалив из него слова, которые состоят менее чем из трех символов.
14. Распечатать введенные слова, отличные от последнего, преобразовав их следующим образом:
15. перенести последнюю букву в начало слова;
16. оставить в слове только первые вхождения каждой буквы.
17. Написать и протестировать функцию strp(str1, str2), которая определяет, встретился ли в строке str1 какой-нибудь символ из строки str2. Функция должна возвращать номер позиции первого символа строки str1, совпадающего с каким-либо символом строки str2, или –1, если совпадений нет.
18. Распечатать введенную строку, удалив из нее слова с нечетными номерами и перевернув слова с четными номерами. Например, из строки: *во что бы то ни стало* должно получиться: *отч от олотс*
19. Написать и протестировать функцию `is_sub_str(str1, str2)`, которая выясняет, является ли строка `str1` подстрокой строки `str2`. Функция должна возвращать номер позиции, с которой начинается подстрока, либо –1, если подстрока не найдена.
20. Напишите функцию, возвращающую номер символа, с которой начинается N – ое вхождение подстроки `Subs` в строку `S`.

### *Лабораторная работа 10*

# Работа со структурами

**Постановка задачи**

В настоящей лабораторной необходимо сформировать массив структур. Структуры вводятся с клавиатуры. В ряде вариантов заданий с клавиатуры

вводится не вся структура, а только ее отдельные поля. Необходимо выполнить обработку сформированного массива структур и результаты обработки вывести на экран дисплея. В программе следует в максимальной степени использовать функции. Например, функции должны использоваться для ввода структур, обработки сформированного массива структур и вывода результатов обработки.

### Варианты заданий

Варианты заданий приведены в таблице

| N | Задание |
|---|---|
| 1 | Массив должен содержать сведения о книгах. Каждая структура должна иметь следующие поля: автор (авторы), название, год издания, цена и издательство. Вывести на экран дисплея список книг, изданных в заданном временном интервале (например, в интервале 1993..2000). |
| 2 | Массив должен содержать сведения об успеваемости студентов факультета. Каждая структура должна содержать следующие поля: шифр группы (например, СП-91), фамилия, имя и отчество (например, Иванов Петр Андреевич), оценки за последнюю сессию (например: 3, 5, 4, 4), средний балл. С клавиатуры вводятся только первые три поля. Вывести на экран сведения о конкретной группе (например, о группе СП-71). |
| 3 | Массив должен содержать сведения о поездах. Каждая структура должна содержать следующие поля: номер поезда, станция назначения, время в пути и цена билета. Вывести на экран дисплея сведения обо всех поездах, находящихся в пути более “k” часов и цена билета которых не превосходит “m” рублей. |
| 4 | Массив должен содержать сведения о товарах, хранящихся в некотором магазине. Каждая структура должна содержать следующие поля: наименование товара, дата поступления и цена. Выполнить обработку массива структур, уменьшив цену товара, хранящегося более “k” месяцев на “m%” (например, на 30%). Полученный массив вывести на экран дисплея. |
| 5 | Массив должен содержать сведения о книгах. Каждая структура должна иметь следующие поля: автор (авторы), название, год издания, цена, тираж. Вывести на экран дисплея сведения о книгах, имеющих тираж более определенной величины (например, более 5000 экз.) и стоимость которых не превосходит определенной величины (например, 50 руб.). |
| 6 | Массив должен содержать сведения о сотрудниках кафедры. Каждая структура должна содержать следующие поля: фамилия, имя и отчество, должность, оклад, год окончания контракта. Вывести на экран сведения о сотрудниках, занимающих определенную должность (например, доцента), контракт с которыми заканчиваются в заданном году. |
| 7 | Массив должен содержать сведения о городах. Каждая структура должна содержать следующие поля: названия города, количество жителей, год основания, количество музеев. Вывести на экран сведения о городах, в которых проживает более “m” жителей и проживает более “n” жителей. |
| 8 | Массив должен содержать сведения о странах. Каждая структура |

| | |
|---|---|
| | должна содержать следующие поля: название страны, название столицы, количество жителей, средняя продолжительность жизни. Вывести на экран сведения о странах, в которых проживает не более “n” жителей и средняя продолжительность жизни, в которых не превосходит “m” лет. |
| 9 | Массив должен содержать сведения о продаже книг некоторым магазином. Каждая структура должна содержать следующие поля: автор (авторы), название, год издания, затраты на покупку книги магазином, количество закупленных книг, цена при продаже. Вычислить прибыль от продажи книги каждого названия. Вывести на экран сведения о книгах, продажа которых не принесла прибыли. |
| 10 | Массив содержит сведения о зарплате сотрудников некоторой фирмы. Каждая структура должна содержать следующие поля: фамилия, имя и отчество, должность, год рождения, массив из 12 элементов с месячной зарплатой, среднемесячная зарплата. С клавиатуры вводятся первые четыре поля. Вывести сведения о сотрудниках моложе “k” и средняя зарплата которых не превосходит “n” рублей. |
| 11 | Массив сведений о погоде за 30 последних дней. Каждая структура должна содержать следующие поля: среднее давление, среднюю скорость ветра, основное направление ветра, состояние облачного покрова (ясный, туманный, хмурый и др.). Вывести сведения о погоде для заданного состояния облачного покрова. |
| 12 | Массив должен содержать сведения о сотрудниках кафедры. Каждая структура должна содержать следующие поля: фамилия, имя и отчество, должность, год поступления на работу. Вывести на экран сведения о сотрудниках, работающие на кафедре не менее “k” лет. |
| 13 | Массив структур, содержащие сведения о запасах некоторых товаров. Каждая структура должна содержать следующие поля: наименование товара, норматив на величину запаса (целое число в диапазоне до 5000), единица измерения (тонна, центнер, литр, упаковка и др.), текущее значение запаса. Вывести на экран сведения о товарах, запасы которых меньше норматива. |
| 14 | Массив структур, содержащих сведения о прибытии поездов на некоторый вокзал. Каждая структура должна содержать следующие поля: станция назначения, номер поезда, время прибытия в форме ЧЧ (часы) ММ (минуты), номер платформы. Вывести на экран сведения о поездах, прибывающих на платформу с номером “n” в заданном интервале времени (например, от 12 до 18 часов). |
| 15 | Массив структур о кафедрах некоторого вуза. Каждая структура должна содержать следующие поля: название кафедры, фамилия, имя и отчество заведующего кафедрой, количество преподавателей, количество потоков, в которых проводит занятия кафедра в текущем семестре. Вывести на экран сведения о кафедрах, штатный состав которых превосходит “k” единиц. |
| 16 | Массив структур, содержащих сведения о поездах, которые отправляются с некоторого вокзала. Каждая структура должна содержать следующие поля: станция назначения, номер поезда, время отправления в форме ЧЧ (часы) ММ (минуты), время в пути, номер платформы. Вывести на экран сведения о поездах, находящихся в пути более “k” часов. |
| 17 | Массив структур, содержащих сведения о фильмах. Каждая структура должна содержать следующие поля: режиссер, название, страна, жанр, |

| | |
|---|---|
| | год создания. Вывести на экран сведения о фильмах определенного жанра, созданных в некоторой стране. |
| 18 | Массив структур, содержащих сведения о футбольных командах. Каждая структура должна содержать следующие поля: название команды, тренер, место, занятое командой в чемпионате в прошлом году, место, занимаемое командой в чемпионате в настоящем году. Вывести на экран сведения о командах, которые занимают место в текущем чемпионате не меньшее, чем в чемпионате прошлого года. |
| 19 | Массив структур, содержащих сведения о вузах. Каждая структура должна содержать следующие поля: название вуза, адрес, количество факультетов, наличие военной кафедры, число обучающихся студентов. Вывести на экран дисплея сведения о вузах, в которых обучается более “n” студентов. |
| 20 | Массив структур, содержащих сведения о музеях. Каждая структура должна содержать следующие поля: название музея, адрес, год основания, цена входного билета. Вывести на экран дисплея сведения о музеях, действующие более “k” лет. |
| 21 | Массив сведений о кинотеатрах. Каждая структура должна содержать следующие поля: название, адрес, категория кинотеатра, количество мест. Вывести на экран дисплея сведения о кинотеатрах, количество мест в которых превосходит “k” . |
| 22 | Массив сведений о подпрограммах. Каждая структура должна содержать следующие поля: имя подпрограммы, язык программирования, вид подпрограммы (функция, процедура), назначение. Вывести на экран дисплея о функциях, у которых количество параметров не превосходит “k”. |
| 23 | Массив сведений о продаже товаров. Каждая структура должна содержать следующие поля: код товара (строка, длина которой не превосходит 10), год продаж, выручка за проданный товар. Вывести на экран дисплея сведения о товарах, выручка за которые превосходит “s” руб. |
| 24 | Массив сведений о фильмах, которые показывают в кинотеатрах города. Каждая структура должна содержать следующие поля: название кинотеатра, название фильма, список сеансов. Вывести на экран дисплея сведения о кинотеатрах, в которых показывают некоторый фильм. |

## Методические указания по выполнению лабораторной работы

Приведем ряд рекомендации, которые могут использоваться при выполнении настоящей работы:

1. Рекомендуется следующая структура программы:

- функция main(),
- функция для ввода массива структур,
- функция для обработки массива структур,
- функция для вывода на экран результатов вычислений.

2. Функция main() должна содержать:

- ввод количества структур “n”,
- вызовы трех других функций (ввода массива структур, обработки и вывода результатов вычислений),

### Контрольные вопросы

1. Какие структурные данные в языке Си называются структурами?
2. Каким образом в языке Си можно объявить тип данных, относящихся к структурам?
3. Каким образом можно объявить переменную, имеющую тип структуры?
4. Каким образом можно инициализировать структуру во время ее определения?
5. Каким образом можно обратиться к отдельным полям записи?
6. Какие операции можно применить к переменной, имеющей тип структуры?
7. Можно ли вернуть структуру в качестве значения функции?
8. Каким образом можно организовать ввод – вывод структур?
9. Каким образом объявляются и используются массивы структур?
10. Каким образом можно обратиться к отдельным полям элемента массива, содержащего структуры?
11. Какие возможности существуют в языке Си по работе со структурами при использовании функций? Можно ли использовать структуры в качестве параметров в функциях? Можно ли в функцию передавать массив структур?

## *Лабораторная работа 11*

# Работа с файлами

### Постановка задачи

Имеется текстовый файл, содержащий произвольное количество строк. Длина каждой строки не превосходит 255 символов. Необходимо выполнить заданную обработку файла.

### Варианты заданий

Варианты заданий приведены в таблице

| N | Задание |
|---|---|
| 1 | В каждой строке исходного файла имеется произвольное количество чисел, записанных в форме f. Количество чисел в строке не превосходит 10. Сформировать новый файл, содержащий нормализованные числа исходного файла. Каждая строка файла нормализуется отдельно путем деления ее элементов на значение максимального элемента строки. |
| 2 | В каждой строке исходного файла имеется произвольное количество чисел, записанных в форме f. Сформировать новый файл, дописав в начало каждой строки исходного файла сумму ее элементов. |
| 3 | В каждой строке исходного файла имеется произвольное количество чисел, записанных в форме f. Сформировать новый файл, дописав в начало каждой строки исходного файла количество, содержащихся в ней чисел. |
| 4 | В каждой строке исходного файла имеется произвольное количество |

| | |
|---|---|
| | слов. Сформировать новый файл, дописав в каждую строку исходного файла количество, содержащихся в ней слов. |
| 5 | В файле хранится числовая матрица. В первой строке файла записаны два числа: количество строк и столбцов матрицы, а затем сама матрица. Сформировать новый файл, в котором должна храниться транспонированная матрица. |
| 6 | В файле хранится числовая матрица. В первой строке файла записаны два числа: количество строк и столбцов матрицы, а затем сама матрица. Вычислить для каждой строки произведение ее элементов. Результаты вычислений должны быть записаны в конец исходного файла в виде таблицы следующего вида:<br>СТРОКА ПРОИЗВЕДЕНИЕ<br><номер строки> <вычисленное произведение> |
| 7 | В файле хранится числовая матрица. В первой строке файла записаны два числа: количество строк и столбцов матрицы, а затем сама матрица. Все отрицательные числа исходной матрицы заменить нулями. Полученную матрицу записать в новый файл. |
| 8 | Имеется текстовый файл, содержащий не более 100 строк. Выполнить сортировку строк файла в прямом лексикографическом порядке. Результаты сортировки записать в новый файл. |
| 9 | В файле хранится числовая матрица. В первой строке файла записаны два числа: количество строк (N <= 20) и столбцов матрицы (M<=5), а затем сама матрица. Вычислить для каждого столбца сумму его элементов. Результаты вычислений должны быть записаны в конец исходного файла в виде таблицы следующего вида:<br>СТОЛБЕЦ СУММА<br><номер столбца> <вычисленная сумма> |
| 10 | Имеется текстовый файл, содержащий не более 100 строк. Выполнить сортировку строк файла в обратном лексикографическом порядке. Результаты сортировки записать в новый файл. |
| 11 | Имеется текстовый файл, содержащий действительные числа. Количество чисел в строке может быть любым. Количество строк не превосходит 100. Для каждой строки вычислить сумму содержащихся в ней чисел, а затем выполнить сортировку строк файла в порядке возрастания суммы. Результаты сортировки записать в новый файл. |
| 12 | Имеется текстовый файл, содержащий действительные числа. Количество чисел в строке может быть любым. Количество строк не превосходит 100. Для каждой строки вычислить сумму содержащихся в ней чисел, а затем выполнить сортировку строк файла в порядке убывания суммы. Результаты сортировки записать в новый файл. |
| 13 | В каждой строке записано произвольное количество чисел, записанных в форме f. Сформировать новый файл, дописав в конец каждой строки исходного файла значение ее максимального элемента. |
| 14 | В каждой строке записано произвольное количество чисел, записанных в форме f. Сформировать новый файл, дописав в начало каждой строки исходного файла значение минимального элемента этой строки. |
| 15 | В файле хранится числовая матрица. В первой строке файла записаны два числа: количество строк и столбцов матрицы, а затем сама матрица. Вычислить для каждой строки сумму ее положительных элементов. Результаты вычислений должны быть записаны в конец исходного |

| | |
|---|---|
| | файла в виде таблицы следующего вида:<br>СТРОКА СУММА ПОЛОЖИТЕЛЬНЫХ ЭЛЕМЕНТОВ<br><номер строки> <вычисленная сумма> |
| 16 | В каждой строке записано произвольное количество чисел, записанных в форме f. Сформировать новый файл, дописав в каждую строку исходного файла сумму ее положительных элементов. |
| 17 | В файле хранится числовая матрица. В первой строке файла записаны два числа: количество строк и столбцов матрицы, а затем сама матрица. Сформировать новый файл, содержащий только те строки исходной матрицы, сумма чисел в которых превышает заданную величину. |
| 18 | В файле хранится числовая матрица. В первой строке файла записаны два числа: количество строк и столбцов матрицы, а затем сама матрица. Сформировать новый файл, содержащий только те строки исходной матрицы, в которых отсутствуют отрицательные числа. |
| 19 | В каждой строке записано произвольное количество чисел, записанных в форме f. Сформировать новый файл, переписав в него только те строки, в которых отсутствуют положительные числа. |
| 20 | В каждой строке файла имеется произвольное количество чисел, записанных в форме f. Сформировать новый файл, содержащий нормализованные числа исходного файла. Каждая строка нормализуется путем деления ее элементов на полусумму значений максимального и минимального элемента строки. |
| 21 | В каждой строке записано произвольное количество чисел, записанных в форме f. Для каждой строки вычислить сумму ее элементов. Дописать в конец файла строку с максимальным значением этой суммы. |
| 22 | В файле хранится числовая матрица. В первой строке файла записаны два числа: количество строк и столбцов матрицы, а затем сама матрица. Все отрицательные числа исходной матрицы возвести в квадрат. Полученную матрицу записать в новый файл. |
| 23 | Дан текстовый файл. Сформировать новый файл, состоящий из строк исходного файла. Порядок строк в новом файле должен быть обратным по отношению к порядку строк в исходном файле. |
| 24 | Дан текстовый файл. Записать в новый текстовый файл все строки исходного файла, которые в качестве фрагмента содержат строку `Str`. |
| 25 | В файле хранится числовая матрица. В первой строке файла записаны два числа: количество строк и столбцов матрицы, а затем - сама матрица. Для каждой строки матрицы вычислить корень квадратный из суммы квадратов ее элементов. Результаты вычислений необходимо записать в новый файл. |
| 26 | В файле хранится числовая матрица. В первой строке файла записаны два числа: количество строк(N <= 10) и столбцов матрицы(M <= 15), а затем - сама матрица. Для каждого столбца матрицы вычислить корень квадратный из суммы квадратов ее элементов. Результаты вычислений необходимо записать в новый файл. |

### Методические указания по выполнению лабораторной работы

Приведем ряд рекомендаций, которые могут быть полезными при выполнении настоящей работы.

1. Требуемую обработку следует оформить в виде функции пользователя. Входной и выходной файлы в эту функцию целесообразно передавать с помощью аппарата формальных и фактических параметров, а подготовительные файловые операции предпочтительнее выполнять в клиентском коде этой функции. К таким операциям относятся следующие операции: открытие и закрытие файла.
2. В ряде задач в файле хранится числовая матрица. Это задачи 5 – 7, 9, 15, 17, 18, 22, 25 и 26. Первая строка файла в этих задачах содержит два числа, которые определяют количество строк и столбцов матрицы. Обработка файла может быть построена следующим образом. Вначале из файла читаются с помощью функции fscanf() два первых числа, которые записываются в две целочисленные переменные `n` и `m`. Затем следует организовать вложенные арифметические циклы.
3. В задачах 8, 10 - 12 необходимо выполнить сортировку строк файла. Для выполнения сортировки можно воспользоваться любым способом. Например, можно использовать простейший способ – метод пузырьковой сортировки. В качестве примера ниже приводится функция для сортировки строк в порядке их возрастания (bibble).

```
#define MAXLENGTH 128
    void bibble(char arr[][MAXLENGTH + 1], int n)
   {
      int i, j;
          char buf[MAXLENGTH + 1];
      for(i = 1; i < n; i++)
      {
             for(j = n - 1; j >=i; --j )
             {
                if(strcmp(arr[j - 1], arr[j]) < 0)
            {
                   strcpy(buf, arr[j - 1]);
               strcpy(arr[j - 1], arr[j]);
               strcpy(arr[j], buf);
                }
         }
      }
   }
```

4. В ряде задач в файле находится произвольное количество строк, каждая из которых содержит произвольное количество чисел. К таким задачам относятся задачи 1 – 3, 13, 14, 16, 19 – 21. В этом случае обработка может состоять из вложенных итерационных циклов. Во внешнем цикле можно читать очередную строку, а во

внутреннем цикле выполнять обработку строки с помощью функции strtod().

5. В процессе решения поставленной задачи следует продумать, какие структуры данных могут потребоваться для ее решения. В первую очередь, это относится к тем задачам, в которых в файле хранится матрица. В этих задачах необходимо выяснить требуется или нет при ее решении использовать двумерный массив. В ряде случаев необходимости в их применении нет. К числу таких задач относится, например задача 25. В этой задаче необходимо вычислить корень квадратный из суммы элементов строки числовой матрицы, хранящейся в текстовом файле. Здесь структура программы – вложенные арифметические циклы. Внутренний цикл должен выполнять суммирование квадратов элементов очередной строки, а внешний цикл – вычисляет корень квадратный из накопленной внутренним циклом суммы. Особенностью этой задачи является то положение, что прочитанное из файла число может сразу же быть обработано (просуммировано с квадратом) и необходимости в его хранении для последующей обработки нет. Совсем иначе дело обстоит при решении задачи 26. Здесь необходимо решить практически ту же задачу, что и в варианте 25, но применительно к столбцам матрицы. Здесь придется вначале прочитать весь файл в матрицу (двумерный числовой массив) и только затем приступить к его обработке.

### Контрольные вопросы и задачи для самостоятельной работы

1. В чем состоит назначение файлов?
2. В чем состоит отличие файлов от обычных переменных, объявляемых в программе?
3. Какие виды библиотечных функций имеются в языке Си?
4. В чем заключается различие между текстовыми и двоичными потоками?
5. В чем состоит различие между потоком и файлом в языке Си?
6. Каким образом объявляется файловый указатель?
7. Из каких этапов складывается работа с файлами?
8. Может ли текстовый файл использоваться в режиме прямого доступа?
9. В чем состоит назначение процедуры `Assign`?
10. Может ли текстовый поток быть открыт в режиме ввода – вывода?
11. В каждой строке записано произвольное количество слов. Сформировать новый файл, дописав в конец каждой строки исходного файла ее номер.
12. В конец каждой нечетной строки записать текст четной строки. Результаты вычислений записать в новый файл.
13. Удалить из исходного файла все строки, длина которых не превосходит заданной величины.

14. Поменять местами строки с четными и нечетными номерами. Результаты записать в новый файл.
15. Имеется текстовый файл, содержащий действительные числа. Количество чисел в строке может быть любым. Количество строк не превосходит 100. Для каждой строки вычислить произведение содержащихся в ней чисел, а затем выполнить сортировку файла в порядке убывания произведения. Результаты сортировки записать в новый файл.
16. Имеется текстовый файл, содержащий действительные числа. Количество чисел в строке может быть любым. Количество строк не превосходит 100. Для каждой строки вычислить сумму положительных чисел, а затем выполнить сортировку файла в порядке возрастания этой суммы. Результаты сортировки записать в новый файл.
17. Имеется текстовый файл, содержащий действительные числа. Количество чисел в строке может быть любым. Количество строк не превосходит 100. Для каждой строки вычислить сумму положительных чисел, а затем выполнить сортировку файла в порядке убывания этой суммы. Результаты сортировки записать в новый файл.
18. Имеется текстовый файл, содержащий действительные числа. Количество чисел в строке может быть любым. Количество строк не превосходит 100. Для каждой строки вычислить произведение положительных чисел, а затем выполнить сортировку файла в порядке убывания этого произведения. Результаты сортировки записать в новый файл.
19. Имеется текстовый файл, содержащий действительные числа. Количество чисел в строке может быть любым. Количество строк не превосходит 100. Для каждой строки вычислить значение медианы, которое затем записать в начало рассматриваемой строки. Результаты записать в новый файл.
20. Имеется текстовый файл, содержащий действительные числа. Количество чисел в строке может быть любым. Количество строк не превосходит 100. Для каждой строки вычислить значение медианы, которое затем записать в конец рассматриваемой строки. Результаты записать в новый файл.
21. Имеется текстовый файл, содержащий действительные числа. Количество чисел в строке может быть любым. Количество строк не превосходит 100. Для каждой строки вычислить значение медианы. Результаты записать в конец исходного файла.
22. Имеется текстовый файл, содержащий действительные числа. Количество чисел в строке может быть любым. Количество строк не превосходит 100. Для каждой строки вычислить значение медианы. Результаты вычислений записать в новый файл.

23. В файле хранится числовая матрица. В первой строке файла записаны два числа: количество строк и столбцов матрицы, а затем сама матрица. Количество столбцов матрицы не превосходит 10. Выполнить для каждой строки матрицы сортировку в порядке возрастания значений содержащихся в ней чисел. Результаты вычислений записать в новый файл.
24. В файле хранится числовая матрица. В первой строке файла записаны два числа: количество строк и столбцов матрицы, а затем сама матрица. Количество столбцов матрицы не превосходит 10. Выполнить для каждой строки матрицы сортировку в порядке уменьшения значений содержащихся в ней чисел. Результаты вычислений записать в новый файл.