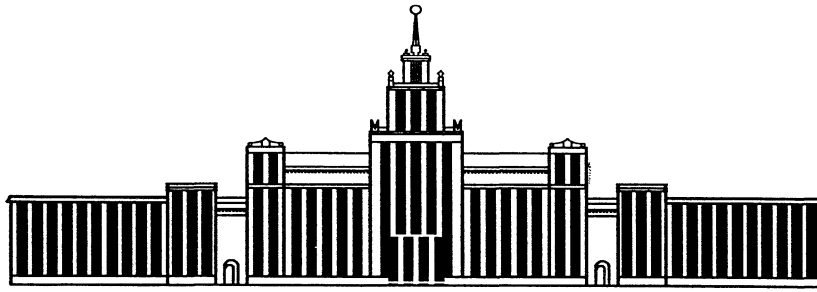
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ



ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

004.4(07)
Я411

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Задания для практических занятий
и методические указания по их выполнению

Челябинск
2013

Министерство образования и науки Российской Федерации
Южно-Уральский государственный университет
Кафедра системного программирования

004.4(07)
Я411

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Задания для практических занятий
и методические указания по их выполнению

Челябинск
Издательский центр ЮУрГУ
2013

УДК 004.43(075.8)
Я411

Одобрено учебно-методической комиссией факультета вычислительной математики и информатики

Методическое пособие подготовлено в соответствии с ФГОС ВПО 3-го поколения по образовательному направлению 010300.62 «Фундаментальная информатика и информационные технологии».

Рецензент:
П.С. Костенецкий

Я411

Языки программирования: задания для практических занятий и методические указания по их выполнению / сост.: Е.В. Иванова, Г.И. Радченко, Т.В. Речкалов, Н.С. Силкина, М.Л. Цымблер. – Челябинск: Издательский центр ЮУрГУ, 2013. – 29 с.

Методическое пособие предназначено для студентов, обучающихся по направлению «Фундаментальная информатика и информационные технологии» при изучении дисциплины «Языки программирования». Пособие содержит тексты задач, контрольные вопросы, указания по выполнению некоторых задач, пример оформления практического задания, а также список рекомендуемой литературы [1–7]. Приведены требования и рекомендации по оформлению программ на языке Си.

УДК 004.43(075.8)

ВВЕДЕНИЕ

Данные методические указания предназначены для выполнения практических работ по дисциплине "Основы программирования" студентами 1 курса направления 010300 "Фундаментальная информатика и информационные технологии ".

Структура пособия. Каждая практическая работа содержит тексты задач и контрольные вопросы, ответы на которые проверяются преподавателем при приеме работы у студента. Текст задачи включает в себя формулировку цели, описание форматов входных и выходных данных и примеры результатов работы программы, решающей данную задачу. Для некоторых задач приводятся указания по их решению.

Каждая задача имеет уникальный в пределах данной лабораторной работы номер. В ссылках на задачу указывается номер задачи и в скобках номер страницы. Например, задача 5 (см. с. 9) означает пятую задачу, описание которой приведено на девятой странице.

Список литературы [1–7] содержит библиографические ссылки на источники, рекомендуемые к прочтению при решении задач.

В приложение вынесены требования и рекомендации по оформлению исходных текстов программ, а так же пример оформления практической работы.

Порядок сдачи практической работы. Выполнение студентом практической работы и сдача ее результатов преподавателю происходит следующим образом.

1. Студент выполняет разработку программ для всех задач, назначенных ему преподавателем. В том числе разрабатывается модульная структура, описываются применяемые алгоритмы, специфицируются и реализуются подпрограммы и тело основной программы. В ходе разработки студент обязан следовать указаниям к данной задаче (в случае их наличия). Исходные тексты программ следует разрабатывать в соответствии с требованиями к оформлению, приведенными в приложении.

В случае затруднений, связанных с разработкой алгоритма, студенту следует обращаться с вопросами к преподавателю, ведущему занятие.

2. Студент выполняет самостоятельную проверку исходного текста каждой разработанной программы и правильности ее работы, а также свои знания по теме практической работы.

Исходные тексты программ должны соответствовать требованиям к оформлению, приведенным в приложении. Недопустимо отсутствие в тексте программы следующих важных элементов оформления: спецификации программного файла и подпрограмм, а также отступов, показывающих структурную вложенность языковых конструкций ("лесенка").

Самостоятельная проверка знаний по теме лабораторной работы выполняется с помощью контрольных вопросов и заданий, приведенных в конце текста лабораторной работы.

3. Студент защищает разработанные программы. Защита заключается в том, что студент должен ответить на вопросы преподавателя, касающиеся разрабо-

танной программы. В том числе вопросы класса, какой будет получен результат в случае использования указанных преподавателем тестовых данных.

К защите необходимо представить исходные тексты программ и сопровождающую информацию по логической организации программы, оформленных в соответствии с требованиями. Пример оформления программы приведен в приложении.

1. ЗАДАНИЯ И МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ИХ ВЫПОЛНЕНИЮ

1.1. Основы языка Си

Изучите базовые элементы языка Си: операции над переменными, управляющие операторы, применение препроцессора, функции ввода/вывода, правила объявления функций, области видимости переменных и классы памяти. Разработайте алгоритм решения задачи и реализуйте его в виде программы на языке высокого уровня Си с помощью изученных базовых элементов.

1. Квадрат

Напишите программу, которая считывает размер стороны квадрата и затем выводит полый квадрат в виде звездочек.

Входные данные: одно целое число – размер стороны квадрата – из отрезка от 0 до 80.

Выходные данные: рисунок полого квадрата в виде звездочек.

Примеры входных и выходных данных

№ п/п	Входные данные	Выходные данные
1	5	***** * * * * * * *****

2. Двоичное число

Введите целое число в двоичной системе счисления и выведите его десятичный эквивалент.

Входные данные: одно целое число.

Выходные данные: десятичное представление введенного числа.

Примеры входных и выходных данных

№ п/п	Входные данные	Выходные данные
1	1101	13

Указание: примените операции деления (/) и взятия по модулю (%) для отделения справа налево одного за другим «двоичного» числа.

3. Вычисление экспоненты

Известно, что e раскладывается в ряд Тейлора следующим образом: $e = \sum_{n=0}^{\infty} \frac{1}{n!} = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!} + \frac{1}{(n+1)!} + \cdots$.

Напишите программу, которая оценивает значение e с заданной точностью ε (0<ε<1) по приведенной выше формуле. Точность ε достигается, когда абсолютное значение очередного слагаемого меньше или равно ε (т.е. $\left|\frac{1}{n!}\right| \leq \varepsilon$).

Примечание. $e = 2.718281828459\ldots$

Входные данные: одно действительное число ε(10< <ε).

Выходные данные: одно действительное число– значение константы e.

Примеры входных и выходных данных

№ п/п	Входные данные	Выходные данные
1	0.1	2.7
2	0.000001	2.718281

Указание. Изучите преобразование типов данных в языке Си.

4. Запись числа прописью

В бухгалтерской деятельности, да и не только в ней, часто возникает необходимость записать некоторую денежную сумму прописью. Правила русского языка в данном случае относительно просты, и поэтому такая задача может быть поручена компьютеру.

Входные данные: одно число - сумма (в рублях), которую необходимо записать прописью. Сумма представляет собой неотрицательное десятичное число, не превосходящее 999.99 (девятьсот девяносто девять рублей 99 копеек) и содержащее не более двух знаков после десятичной точки. Незначащие ведущие нули отсутствуют.

Выходные данные: программа должна вывести запись суммы прописью. Число рублей передается соответствующим количественным числительным, число копеек- цифрами, например: тридцать пять рублей56 копеек. Запись должна быть грамматически верной. Запись всегда должна быть полной, т.е. включать количество рублей и копеек. Количество копеек должно быть записано двумя цифрами, т.е. при необходимости дополнено ведущими нулями.

Примеры входных и выходных данных

№ п/п	Входные данные	Выходные данные
1	654.65	Шестьсот пятьдесят четыре рубля 65 копеек

Указание: в реализации программы используйте оператор switch.

5*. Пифагоровы тройки

Прямоугольный треугольник может иметь стороны, каждая из которых является целым числом. Набор трех целочисленных значений для сторон прямоугольного треугольника называется пифагоровой тройкой. Эти три стороны должны удовлетворять следующему соотношению: сумма квадратов двух кате-

тов равна квадрату гипотенузы. Напишите программу, которая выводит все пифагоровы тройки для катетов и гипотенузы, не превосходящих целого числа n.

Входные данные: одно целое число n.

Выходные данные: все пифагоровы тройки для катетов и гипотенузы в виде таблицы (см. пример выходных данных).

Примеры входных и выходных данных

№ п/п	Входные данные	Выходные данные
1	5	Leg 1 Leg 2 Hypotenuse 3 4 5 4 3 5

Указание. Используйте цикл с тройной вложенностью, в котором перебираются все возможности. Определите количество операций в таком цикле. Можно ли сократить количество операций?

1.2. Встроенные типы данных языка Си

Изучите следующие элементы языка Си: работа с одномерными и многомерными массивами, работа со строками, создание рекурсивных функций и функции для работы с файлами. Разработайте алгоритм решения задачи и реализуйте его в виде программы на языке высокого уровня Си с помощью изученных средств.

1. Магический квадрат

Магическим квадратом порядка N называется квадрат из NxN ячеек, в каждой из которых записано одно натуральное число. Числа размещены так, что суммы элементов любого столбца, строки или главной диагонали одинаковы. На рис. 2 представлен пример магического квадрата порядка 3.

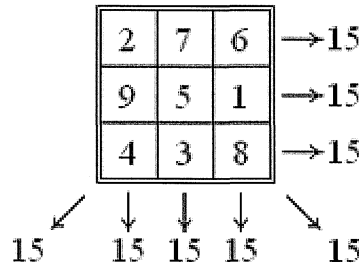


Рис. 1. Магический квадрат третьего порядка

Напишите программу, которая определяет, является ли квадрат магическим.

Входные данные: в первой строке файла записано целое число N – порядок магического квадрата. В следующих N строках записано по N целых чисел через пробел.

Выходные данные: слово ‘YES’, если квадрат является магическим, ‘NO’ – иначе.

Примеры входных и выходных данных

№ п/п	Входные данные	Выходные данные
1	4 13 8 12 1 2 11 7 14 3 10 6 15 16 5 9 4	YES

Указание. Используйте в реализации двумерный массив.

2. Выборочная сортировка

При выборочной сортировке происходит поиск наименьшего элемента в массиве. Когда наименьший элемент найден, его меняют местами с первым элементом массива. Затем процесс повторяется для подмассива, начинающегося со второго элемента массива.

Каждый проход по массиву приводит к помещению одного элемента на подходящее для него место. Если обрабатываемый массив содержит только один элемент, массив считается отсортированным.

Входные данные: целочисленные элементы массива, записанные по одному в каждой строке. Количество элементов не меньше 1.

Выходные данные: отсортированный массив.

Примеры входных и выходных данных

№ п/п	Входные данные	Выходные данные
1	7 5 2 0 2 -8 90 12	-8 0 2 2 5 7 12 90

Указание. Напишите рекурсивную функцию selectionSort для выполнения этого алгоритма.

3. Печать чека

Чеки, разработанные для компьютерного вывода, содержат фиксированное число полей, в которых компьютер может печатать сумму покупки. Пусть платежный чек содержит девять пустых полей. Если сумма большая, то будут заполнены все девять полей, например:

```
12'230.60 (сумма чека)
---------
123456789 (позиции цифр)
```

Если же сумма меньше 10000 рублей, то несколько полей останутся пустыми. Чек с пустыми полями может быть намеренно дописан мошенником, который предполагает получить деньги по подложному чеку. Для предотвращения

этого большая часть систем выдачи чеков печатает в пустых полях символы звездочек:

```
***519.15
---------
123456789
```

Напишите программу, которая вычисляет сумму покупки, складывающуюся из суммы стоимости товаров, и печатает ее на чеке, вставляя, при необходимости, символы звездочек.

Входные данные: в каждой строке файла записано одно вещественное число– стоимость товара.

Выходные данные: чек в следующем формате: первая строка– в девяти полях сумма покупки, при необходимости, вставить разделительный апостроф на 3-ю позицию и дополнить звездочками. Вторая строка– девять символов тире. Третья строка – позиции цифр.

Примеры входных и выходных данных

№ п/п	Входные данные	Выходные данные
1	456.9 18239 23 90.1	18'809.00 --------- 123456789
2	15.5 60	****75.50 --------- 123456789

Указание. Изучите функции работы со строками. Обратите внимание на функцию sprintf.

4*. Угадай число

Напишите программу, которая реализует игру «Угадай число». Правила игры следующие. Играют двое. Один задумывает число, второй – угадывает. На каждом шаге угадывающий делает предположение, а задумавший число – говорит, сколько цифр числа угаданы и сколько из угаданных цифр занимают правильные позиции в числе. Например, если задуманное число 725 и выдвинуто предположение, что задумано число 523, то угаданы две цифры (5 и 2) и одна из них (2) занимает верную позицию.

Входные данные: предположения о загаданном числе– трехзначные целые числа. Предусмотреть случаи некорректного ввода.

Выходные данные: сообщения об общем количестве угаданных цифр и количестве угаданных цифр, которые находятся на своих местах.

Примеры входных и выходных данных

№ п/п	Входные данные	Выходные данные
1	123 476 654 650 658	Компьютер задумал трехзначное число. Вы должны его отгадать. После очередного числа вам будет сообщено, сколько цифр угадано и сколько из них находятся на своих местах. После ввода числа нажмите<Enter>.

№ п/п	Входные данные	Выходные данные
		Для завершения игры нажмите<Esc>. Ваш вариант-> 123 Угадано 0. На своих местах0 Ваш вариант-> 476 Угадано 1. На своих местах0 Ваш вариант-> 654 Угадано 2. На своих местах2 Ваш вариант-> 650 Угадано 2. На своих местах2 Ваш вариант-> 658 Угадано 3. На своих местах3 *** ВЫ УГАДАЛИ! *** Нажмите <Enter> для завершения.

Указание. Изучите функции cprintf, textcolor, textbackground. Используйте их в своей программе.

1.3. Указатели и структуры данных

Изучите указатели и структуры данных в языке Си. Разработайте алгоритм решения задачи и реализуйте его в виде программы на языке высокого уровня Си с помощью изученных средств.

1. Польская запись

Пусть выражение – это конструкция следующего вида:

```
<выражение> ::= <терм> | [(]<терм> <знак> <выражение>[)]
<знак> ::= +|-|*|/
<терм> ::= <множитель> * <терм>
<множитель> ::= <число> | <выражение>
<число> ::= 0|1|…|9
```

Польской записью выражения A <знак> B называется запись, в которой знак операции размещен за операндами A B <знак>.

Напишите программу, которая переводит обычное выражение в польскую запись.

Входные данные: в файле в одной строке записано выражение.

Выходные данные: в файл записать выражение в польской записи.

Примеры входных и выходных данных

№ п/п	Входные данные	Выходные данные
1	`(6 + 2) * 5 - 8 / 2`	`6 2 + 5 * 8 2 / -`

Указания.

Программа должна считывать выражение в массив символов `infix`, выражение сохранять в массиве `postfix` и использовать стек. Алгоритм создания польской записи выражения:

1. Ввести в стек левую скобку `'('`.
2. Добавить правую скобку `')'` в конец `infix`.

3. Пока стек не пуст, считывать `infix` слева направо и выполнять следующие действия:

a. Если текущий символ в `infix` – цифра, скопировать его в следующий элемент `postfix`.

b. Если текущий символ в `infix` – знак операции, извлекать знаки операций из стека (если они там есть), пока соответствующие им операции имеют равный или более высокий приоритет по сравнению с текущей операцией, и вставлять извлеченные знаки операций в `postfix`.

c. Вставить текущий символ из `infix` в стек.

d. Если текущий символ в `infix` – правая скобка, извлекать знаки операций из стека и вставлять их в `postfix` до тех пор, пока на вершине стека не появится левая скобка.

e. Извлечь из стека левую скобку и отбросить ее.

Напишите модуль, реализующий структуру данных «Стек» с элементами типа `char`.

Реализуйте следующие операции: помещение значения в стек, извлечение значения из стека, просмотр значения в верхнем элементе стека без его извлечения, проверка стека на пустоту.

2. Двоичное дерево поиска

Напишите программу, которая делает следующее:

1. Строит двоичное дерево из случайных чисел.
2. Выдает все обходы построенного дерева.
3. Ищет заданный элемент и удаляет его, если элемент найден. Выдает сообщение о результате.
4. Строит копию данного дерева.
5. Сравнивает первоначальное дерево с его копией и выдает результат.
6. Строит другое двоичное дерево.
7. Выдает все обходы построенного дерева.
8. Сравнивает первое и второе дерево и выдает результат.
9. Уничтожает все деревья.

Входные данные: в файле записаны целые числа – элементы поиска.

Выходные данные: обходы дерева, результат поиска заданного(ых) элемента(ов), результат сравнения дерева с его копией, обходы второго дерева, результат сравнения первого и второго дерева.

Указание. Напишите модуль, реализующий структуру данных «Двоичное дерево» элементов типа `int`. Реализуйте следующие операции над двоичным деревом: инициализация дерева, уничтожение дерева, добавление элемента, обход дерева (прямой, обратный, концевой), поиск и удаление узла дерева.

1.4. Технология OpenMP

Изучите основные принципы работы технологии OpenMP. Разработайте алгоритм решения задачи и реализуйте его в виде программы на языке высокого уровня Си с использованием распараллеливания на основе технологии OpenMP.

1. Простая программа на OpenMP

Создайте параллельную область с количеством нитей заданным пользователем. Каждая нить должна вывести на экран свой номер. Кроме того, корневая нить должна вывести общее количество нитей.

Входные данные: одно целое неотрицательное число.

Выходные данные: от каждой нити - строка "I'm thread <номер_нити>". От корневой нити - дополнительно строка "Number of threads <количество_нитей>".

Примеры входных и выходных данных

№ п/п	Входные данные	Выходные данные
1	3	I'm thread 0 I'm thread 1 I'm thread 2 Number of threads 3

Указания:

1. Создайте проект в MS Visual Studio согласно инструкции.
2. Для создания параллельной области воспользуйтесь директивой `parallel`.
3. Чтобы узнать количество нитей в параллельной области, воспользуйтесь функцией `omp_get_num_thread`.
4. Чтобы узнать номер нити в параллельной области, воспользуйтесь функцией `omp_get_thread_num`.
5. Чтобы замерить время выполнения программы, вместо функции `clock` воспользуйтесь функцией `omp_get_wtime`.

Задача 2: "Вычисление числа ПИ"

Напишите параллельную версию программы, решающей задачу "Вычисление числа ПИ". Произведите замеры времени работы последовательной и параллельной версий программы.

Указания:

1. Изучите последовательную программу "Вычисление числа ПИ" (http://parallelschool.susu.ru/task.php?step=c&id=131).
2. Для распараллеливания воспользуйтесь директивой `for`.
3. Выберите наиболее подходящее значение параметра `schedule` директивы `for`.
4. Изучите параметр приведения `reduction`. Используйте его в своей программе для суммирования частичных сумм, вычисленных каждой нитью.

3. Магический квадрат

Напишите параллельную версию программы, решающей задачу "Магический квадрат" из практической работы 2 части 1. Произведите замеры времени работы последовательной и параллельной версий программы.

Указания: для распараллеливания воспользуйтесь директивой `sections`.

4*. Вычисление числа ПИ без параметра reduction

Перепишите без использования параметра приведения `reduction` свою программу, вычисляющую число ПИ.

Указания: при выполнении задания воспользуйтесь директивой `critical`.

1.5. Технология MPI

Изучите основные принципы работы технологии MPI. Разработайте алгоритм решения задачи и реализуйте его в виде программы на языке высокого уровня Си с использованием распараллеливания на основе технологии MPI.

1. Простая программа на MPI

Создайте параллельную программу с помощью MPI, в которой каждый процесс выводит на экран свой номер. Кроме того, процесс с номером 0 должен вывести общее количество процессов.

Входные данные: нет.

Выходные данные: от каждого процесса - строка "`I'm process <номер_процесса>`". От нулевого процесса - дополнительно строка "`Number of processes <количество_процессов>`".

Примеры входных и выходных данных

№ п/п	Входные данные	Выходные данные для 3 процессов
1	-	I'm thread 0 I'm thread 1 I'm thread 2 Number of threads 3

Указания:

1. Создайте проект в MS Visual Studio согласно инструкции.
2. Чтобы узнать количество процессов в приложении воспользуйтесь функцией `MPI_Comm_size`.
3. Чтобы узнать номер (ранг) процесса в приложении воспользуйтесь функцией `MPI_Comm_rank`.

2. Пинг-Понг

Напишите параллельную программу, в которой два процесса обмениваются между собой сообщениями следующим образом:

Первый процесс отправляет второму сообщение. После этого выводит на экран "`I'm process <номер_процесса>. Sent message to <номер_процесса_получателя>`".

Второй процесс принимает сообщение от первого процесса и только после этого отправляет свое сообщение первому процессу. При этом выводит на эк-

ран "I'm process <номер_процесса>. Received message from <номер_процесса_отправителя>" и "I'm process <номер_процесса>. Sent message to <номер_процесса_получателя>" после наступления соответствующих событий.

Первый процесс принимает сообщение от второго процесса. После выводит на экран "I'm process <номер_процесса>. Received message from <номер_процесса_отправителя>"

Входные данные: нет.

Выходные данные: от каждого процесса - строки "I'm process <номер_процесса>. Sent message to <номер_процесса_получателя>" и "I'm process <номер_процесса>. Received message from <номер_процесса_отправителя>".

Примеры входных и выходных данных

№ п/п	Входные данные	Выходные данные
1	-	I'm process 0. Sent message to 1 I'm process 1. Received message from 0 I'm process 1. Sent message to 0 I'm process 0. Received message from 1

Указания: для обмена сообщениями воспользуйтесь блокирующей функцией MPI_Send.

3. Эстафетная палочка

Напишите параллельную программу, в которой N-процессов передают друг другу сообщение по принципу эстафетной палочки: 0-й отправляет 1-му, 1-й отправляет 2-му, ..., N-1-й отправляет N-му, N-й отправляет 0-му.

Входные данные: -

Выходные данные: выводятся на экран

Указания: для обмена сообщениями воспользуйтесь блокирующей функцией MPI_Send или MPI_Sendrecv.

2. ВОПРОСЫ ДЛЯ САМОКОНТРОЛЯ

2.1. Основы языка Си

1. Как выглядит минимальная программа на языке Си, которая компилируется и выполняется? Где находится точка входа в Си-программу?

2. Перечислите типы данных языка Си. Для чего используются ключевые слова signed и unsigned? Для чего используется функция sizeof?

3. Что выведет на экран следующая программа? Почему? Как получить правильный результат?

```
#include <stdio.h>
void main()
{
  double x = 5 / 2;

  printf("%g\n", x);
  return;
}
```

4. Что выведет на экран следующая программа?

```
#include <stdio.h>
void main()
{
  int a = 4, b = 3, c;

  c = a-- + b;
  printf("%d, %d, %d\n", a, b, c);
  c = --a + b;
  printf("%d, %d, %d\n", a, b, c);
  return;
}
```

5. Что выведет на экран следующая программа? Объясните результат.

```
#include <stdio.h>
void main()
{
  int a = 5, b = -5;

  while (a-- > 0 || b++ > 0)
  printf("%d %d\n", a, b);
  return;
}
```

6. Напишите макроопределение ABS(x) (ABS - модуль числа). Воспользуйтесь условной операцией ?:

7. Как с помощью функций scanf и printf считать и вывести:
   a. число типа int?
   b. число типа long int?
   c. число типа unsigned int?
   d. число типа float с двумя знаками после запятой?
   e. число типа double?

f. символ?
g. строку?

8. Что печатает программа и почему?

```
/* пример Bjarne Stroustrup-a */
#include <stdio.h>

int a = 1;

void f()
{
  int b = 1;
  static int c = 1;

  printf("a=%d b=%d c=%d\n", a++, b++, c++);
  return;
}

void main()
{
  while (a < 4)
  f();
  return;
}
```

9. Найдите ошибки в следующей программе. Объясните причины возникновения ошибок.

```
#include <stdio.h>

double x = 1.17;
double y = x * 2.0;

void main()
{
  double z = x + 1.9;
  // ...
  return;
}
```

10. Что такое прототип функции? Для чего он используется?

2.2. Встроенные типы данных языка Си

1. Какая разница между седьмым элементом массива и элементом массива с индексом семь?

2. Найдите ошибки в следующем фрагменте кода. Объясните, почему они возникают?

```
#define MAX (10)

void f(int n, int m)
{
  int a[10];
  int b[n];
  int c[MAX];
```

```
  int d[n*2];
  int e[n * m];

  // ...
  return;
}
```

3. Как передать массив в качестве параметра функции?

4. Как объявляется строка в языке Си?

5. Что выведет на экран следующая программа? Объясните результат.

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

void main()
{
  char str[] = "ab\0cd\nxyz";

  printf("%s\n", str);
  printf("%d\n", sizeof(str));
  printf("%d\n", strlen(str));
  return;
}
```

6. Пусть в программе объявлены строки str1 и str2:

```
char str1[6];
char str2[6] = "Hello";
```

Укажите синтаксически верный(ые) оператор(ы) из ниже приведенных:

```
str1 = "Hello";
str1 = str2;
strcpy(str1, str2);
```

7. Скомпилируйте и запустите следующую программу. Объясните полученный результат. Исправьте программу так, чтобы она верно выводила строчку "Hello, World! Hello!".

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

void main()
{
  char str3[5] = {'W','o','r','l','d'};
  char str2[3];
  char str1[5] = {'H','e','l','l','o'};

  strcpy(str2, str1);
  printf("%s, %s! %s!\n", str1, str3, str2);
  return;
}
```

8. Назовите и объясните назначение параметров функции main.

2.3. Указатели и структуры данных

1. Что такое структуры данных в языке Си?
2. Приведите пример объявления и использования структуры данных.
3. Что такое указатели в языке Си?
4. Какие функции предназначены для динамического выделения памяти?
5. Приведите пример использования указателей.
6. Чему равно значение переменной result после выполнения данного фрагмента кода:

```
int array[10];
int* a = array;
int* b = &array[9];
int result = b - a;
```

2.4. Технология OpenMP

1. Какой формат записи директив в OpenMP?
2. Что такое параллельная область? С помощью каких директив в OpenMP создается параллельная область? Как задается количество нитей в параллельной области OpenMP-программы?
3. Дайте определение общим и частным переменным в OpenMP. Как задаются общие и частные переменные в OpenMP-программе?
4. Что выведет следующая программа?

```
int a, b;
#pragma omp parallel private(a)
{
  int c;
  a = omp_get_thread_num();
  b = omp_get_thread_num();
  c = omp_get_thread_num();
  printf("%d %d %d\n", a, b, c);
}
```

5. Как осуществляется распараллеливание циклов в OpenMP? Какое условие должно выполняться, чтобы циклы могли быть распараллелены?
6. Укажите циклы for, которые могут быть распараллелены? Объясните свой выбор?

```
// 1).
for(i=0;i<100;i++) {
  sum = sum + i;
}

// 2).
for(i=0;i<100;i++) {
  z[i] = x[i-1];
}

// 3).
for(i=0;i<100;i++) {
  a[i] = a[i-1]+3;
}
```

2.5. Технология MPI

1. Какой минимальный набор функций MPI позволяет начать разработку параллельных программ?
2. Как описываются передаваемые сообщения?
3. Как можно организовать прием сообщений от конкретных процессов?
4. Как определить время выполнения MPI программы?
5. В чем различие парных и коллективных операций передачи данных?
6. Какая функция MPI обеспечивает передачу данных от одного процесса всем процессам?
7. Что понимается под операцией редукции?
8. Какие режимы передачи данных поддерживаются в MPI?
9. Что понимается в MPI под коммуникатором?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Керниган, Б.. Язык программирования Си / Б. Керниган, Д. Ритчи; пер. с англ., 3-е изд., испр. – СПб.: "Невский Диалект", 2001. – 352 с.
2. Громов, Ю.Ю. Программирование на языке СИ: Учебное пособие / Ю.Ю. Громов, С.И. Татаренко. – Тамбов, 1995. – 169 с.
3. Справочник по библиотечным функциям языка Си / URL: http://www.codenet.ru/progr/cpp/sprd/ (дата обращения 9.08.2013)
4. Учебный курс. Параллельное программирование с использованием OpenMP. / URL: http://www.intuit.ru/department/se/openmp/ (дата обращения 9.08.2013)
5. The OpenMP® API specification for parallel programming. / URL: www.openmp.org (дата обращения 9.08.2013)
6. Что такое OpenMP? / URL: http://www.parallel.ru/tech/tech_dev/openmp.html (дата обращения 9.08.2013)
7. Евсеев, И.В. MPI для начинающих. / URL: http://www.opennet.ru/docs/RUS/MPI_intro/ (дата обращения 9.08.2013)

ПРИЛОЖЕНИЯ

Приложение 1. Требования и рекомендации по оформлению исходных текстов программ

1. Соглашения по идентификаторам

Подбор идентификаторов

1. Все идентификаторы должны выбираться из соображений читаемости и максимальной семантической нагрузки. Например:

```
#define EPS (0.0001) // точность

int sum; // сумма
char *message; // сообщение
```

Неудачными можно считать идентификаторы:

```
#define uU (0.0001) // точность
```

```
int  uu; // сумма
char *zz; // сообщение
```

2. Идентификаторы рекомендуется подбирать из слов английского языка. Например:

```
/* выдает звуковой сигнал заданной частоты и длительности */
void beep(int hertz, int msec)

/* выдает 1, если файл с именем fname существует */
int exist_file(char *fname)

int done; // признак окончания работы с программой
double width, height; // размеры изделия (ширина, высота)
```

Не очень удачными можно считать идентификаторы:

```
/* выдает звуковой сигнал заданной частоты и длительности */
void zvuk(int chast, int dlit)

/* выдает 1, если файл с именем Im существует */
int est_file(char *Im)

int konec; // признак окончания работы с программой
double shirina, vysota; // размеры изделия (ширина, высота)
```

Написание идентификаторов

1. Идентификаторы констант и макроопределений рекомендуется писать заглавными буквами. Например:

```
#define PI (3.14) // значение числа пи

#define MAX(x,y) ((x)>(y))?(x):(y) // максимум двух чисел
```

2. Существуют разные подходы к написанию остальных идентификаторов. Например:

а) все буквы идентификатора пишутся маленькими, для разделения слов в идентификаторе используется символ "_"

```
int cnt_node; // количество звеньев

/* удалить звено с номером i из списка node */
void delete_node(list *node, int i)
```

б) в идентификаторах каждое слово, входящее в идентификатор, писать, начиная с большой буквы, остальные буквы - маленькие.

```
int CntNode; // количество звеньев

/* удалить звено с номером I из списка Node */
void DeleteNode(List *Node, int I)
```

в) аналогично б) за исключением того, что первая буква идентификатора пишется маленькой.

```
int cntNode; // количество звеньев

/* удалить звено с номером i из списка node */
void deleteNode(list *node, int i)
```

Принятого подхода нужно придерживаться во всем тексте программы.

2. Соглашения по самодокументируемости программ

Комментарии

1. Комментарии в теле программы следует писать на русском языке так, чтобы программист, не участвовавший в разработке программы (но имеющий опыт работы на языке Си), мог без особого труда разобраться в логике программы, и при необходимости, сопровождать данный программный продукт.

Спецификация пользовательской процедуры или функции

Для каждой пользовательской процедуры или функции должна быть описана в виде комментария спецификация, содержащая следующую информацию:

- назначение процедуры или функции;
- описание семантики параметров-значений (параметров, передаваемых по значению), если она неочевидна;
- описание семантики параметров-переменных (параметров, передаваемых по ссылке), если она неочевидна.
- для функции: описание семантики возвращаемого значения, если она неочевидна.

Например:

1) Семантика параметров и возвращаемого значения очевидна

```
/* возвращает 1, если год year -- високосный */
int is_leap_year(int year)
```

2) Семантика параметров очевидна, семантика возвращаемого значения неочевидна

```
/* Возвращает день недели даты d/m/y;
 год y должен быть в отрезке 1582..4902;
 результат: ВСК = 0, ПНД = 1, ВТР = 2, ... СБТ = 6 */
int day_of_week(int d, int m, int y)
```

3) Семантика параметров и возвращаемого значения неочевидна

```
#define MAXN (10)

typedef double matrix_t[MAXN, MAXN];
typedef double vector[MAXN];

/* Решение системы линейных алгебраических уравнений
  методом Гаусса.
  Входные данные:
  a   -- матрица коэффициентов системы;
  b   -- столбец свободных членов системы;
  eps -- точность вычислений.
  Выходные данные:
  x            -- вектор решения;
  has_solution -- флаг, устанавливаемый в 1, если решение
                  системы существует, и в 0 во всех
                  остальных случаях;
  num_of_roots -- число корней в решении системы, может
                  принимать значения:
                  0      -- если решение системы не существует,
                  MAXN   -- если решение системы существует и
                            единственно,
```

```
                  MAXINT -- если существует бесконечное
                            множество решений;
    det           -- значение определителя матрицы a;
    afor_reverse  -- нижняя треугольная матрица, полученная из a в
                     в результате выполнения прямого хода алгоритма
                     Гаусса;
    bfor_reverse  -- столбец свободных членов, полученный из b в
                     в результате выполнения прямого хода алгоритма
                     Гаусса.                                    */
  double gauss(matrix_t a, vector b, double eps,
    vector * x,
    int * has_solution,
    int * num_of_roots,
    double * det,
    matrix_t * afor_reverse,
    vector * bfor_reverse)
```

Замечание: Если функция реализует какой-либо вычислительный метод (например: нахождение площади фигуры методом трапеций, поиск минимума функции методом Ньютона и т.п.), необходимо в теле процедуры (функции) поместить комментарий с кратким описанием метода, либо ссылку на источник, где описан метод.

Прототипы функций достаточно снабдить кратким комментарием назначения функции. Например:

```
/* решение системы линейных алгебраических уравнений
 методом Гаусса */
double gauss(matrix_t, vector, double, vector *, int *, int *,
double *, matrix_t *, vector *);
```

Спецификация пользовательского программного файла или модуля

Программный файл или модуль (unit) должен начинаться со спецификации в виде комментария, содержащего следующую информацию:

- имя файла;
- фамилия автора;
- дата написания файла;
- версия языка и замечания по компиляции программы (модуля) в других версиях языка;
- назначение программы (модуля);

Например:

```
/* primes.c
 --------------------------
 (c)оздал: Иванов И.И.
 группа  : ММ-216
 дата    : 01/09/07
 для     : Borland C++ Builder 6.0
  ------------------------------------------------------
  Подсчет количества простых чисел в промежутке [1..200]. */
```

Замечание: В программном файле после заголовка "Program <имя программы>;" (в файле с пользовательским модулем — после заголовка "unit <имя модуля>;") рекомендуется поместить комментарий с указаниями по запуску программы и работе с ней (указаниями по использованию модуля другими про-

граммистами) или ссылку на источник, который использован при составлении программы (модуля).

3. Соглашения по читаемости программ

1. Для отражения структурной вложенности языковых конструкций в тексте программы должна использоваться "лесенка" (отступы от левого края). Рекомендуется отступ лесенки не менее 2-х и не более 4-х пробелов.

Принятого отступа нужно придерживаться во всем тексте программы. Правильное написание операторов языка Си приведено в таблице 1.

Таблица 1

Правильное написание операторов языка Си

Оператор	Правильное написание
if	if (<условие>) <оператор>;

	if (<условие>) {
	  <операторы>;
	}

	if (<условие>) {
	  <операторы>
	} else {
	  <операторы>
	}

	if (<условие>)
	  <оператор>;
	else
	  <оператор>;

while	while (<условие>) {
	  <операторы>
	}

	while (<условие>) <оператор>;

for	for (<выражение1>; <выражение2>; <выражение3>) {
	  <операторы>
	}

	for (<выражение1>; <выражение2>; <выражение3>) <оператор>;
do - while	do { <операторы> } while (<условие>);
	do <оператор>; while (<условие>);

switch	switch (<выражение>) {
	case <выражение>:
	  <операторы>
	  break;
	.....
	default:
	  <операторы>
	}

определение функции	<тип> <имя_функции>(<список_параметров>)
	{
	  <операторы>
	}

Например:

1)

```
int sign(double x)
  /* выдает знак числа x */
{
  int result;

  if (x > 0)
    result = 1;
  else
    if (x < 0)
      result = -1;
    else
      result = 0;
  return result;
}
```

2)

```
/* нахождение действительных корней квадратного уравнения;
  a, b, c -- коэффициенты
  x1, x2  -- корни (если действительного решения нет,
              то полагаются равными 0);
  num     -- число корней (0, 1, или 2)                  */
void equation(double a, double b, double c, double *x1, double
*x2, int *num)
{
  double d;

  d = pow(b, 2) - 4 * a * c;
  if (d < 0) {
    *num = 0;
    *x1 = 0.0;
    *x2 = 0.0;
  } else {
    *x1 = (- b + sqrt(d)) / (2 * a);
    *x2 = (- b - sqrt(d)) / (2 * a);
    if (*x1 == *x2)
      *num = 1;
    else
      *num = 2;
  }
}
```

4. Прочие рекомендации

1. Длина строк программы не должна превышать ширины экрана (80 символов). Инструкции длиннее 80 символов разбиваются на логические части, которые всегда значительно короче, чем изначальная строка, и располагаются со сдвигом вправо. То же самое относится к заголовкам функций с длинным списком аргументов и к длинным строковым константам. Например:

```
double function(double param1, double param2,
 double param3, double param4)
{
 double result;
```

```
  result = param1 + sqrt(param2) - param3 +
    pow(param3, param4) * cos(param4);
  printf("Внимание! Это длинный printf с 5-ью параметрами: "
    "1-й параметр = %f, 2-й параметр = %f, 3-й параметр = %f, "
    "4-й параметр = %f, результат: %f\n", param1, param2, param3,
    param4, result);
  return result;
}
```

2. Рекомендуется операнды бинарных операций отделять от знака операции одним пробелом. Например: `Sum = A + B;`

3. Рекомендуется при перечислении идентификаторов после запятой ставить один пробел. Например:

```
double a, b;
printf("Сумма: %f. Разность: %f.", a + b, a - b);
```

5. Рекомендуется всегда писать символ-разделитель операторов ";" непосредственно после оператора. Например:

1)

```
switch (num) {
case 1:
  printf("один...");
  break;
case 2:
  printf("два...");
  break;
case 3:
  printf("три..."); // <-- здесь
  break;
default:
  printf("много!"); // <-- и здесь
}
```

2)

```
if (n < 0) {
  printf("Введено неверное значение n, прерываем работу!");
  exit(0); // <-- здесь
}
```

Приложение 2. Пример оформления практической работы

Иванов Иван
Группа ВМИ-117

Домашняя работа № 2 – Определение вида треугольника по трем сторонам

1. Условие задачи

Заданы длины сторон треугольника. Если треугольник с такими сторонами существует, определить его вид: равносторонний, равнобедренный, общий. Иначе сообщить, что треугольник не существует.

Входные данные

Три вещественных числа (не обязательно положительные!) – длины сторон треугольника.

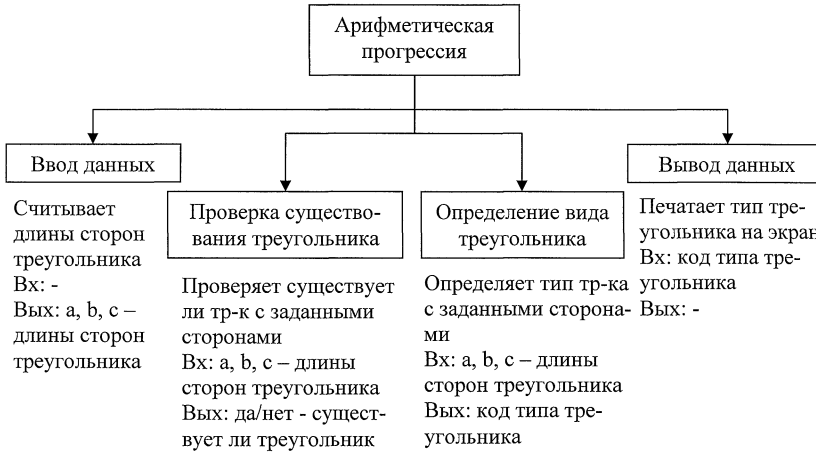
Выходные данные

Одно из сообщений: РАВНОСТОРОННИЙ, РАВНОБЕДРЕННЫЙ, ОБЩЕГО ВИДА, НЕ СУЩЕСТВУЕТ.

Пример тестовых данных:

№	Входные данные	Выходные данные
1	1 1 1	РАВНОСТОРОННИЙ
2	1 2 3	НЕ СУЩЕСТВУЕТ
3	3 4 5	ОБЩЕГО ВИДА

2. Модульная структура



3. Листинг

```
#include<stdio.h>
/* TRIANGLE.PAS
  Нахождение вида треугольника.
  (c) Иванов И.И. (ММ-156)
  11-дек-13 */

// Запрос у пользователя и ввод длин трех сторон треугольника с клавиатуры.
void readTriangleSides(double* a,double* b,double* c){
  printf("Введите длины сторон треугольника: \n");
  scanf("%lf %lf %lf", a, b, c);
}
// Определение существования треугольника по заданным длинам сторон.
// Если треугольник существует, возвращает TRUE, иначе FALSE.
int isTriangleExist(double a,double b,double c){
  if ((a + b > c) && (a + c > b) && (b + c > a)){
    return 1;
  } else {
    return 0;
  }
}
/* Определение вида треугольника по заданным длинам сторон.
Возвращает:
3 треугольник равносторонний
```

```
2 треугольник равнобедренный
1 треугольник общего вида
Треугольник с заданными сторонами заведомо должен существовать. */
int getTriangleKind(double a,double b,double c){
  if ((a == b) && (b == c)) {
    return 3;
  } else if ((a == b) || (a == c) || (b == c)) {
    return 2;
  } else {
    return 1;
  }
}
/* Вывод на экран сообщения о виде треугольника в зависимости от зна-чения K:
0 ТРЕУГОЛЬНИК НЕ СУЩЕСТВУЕТ
1 ТРЕУГОЛЬНИК ОБЩЕГО ВИДА
2 ТРЕУГОЛЬНИК РАВНОБЕДРЕННЫЙ
3 ТРЕУГОЛЬНИК РАВНОСТОРОННИЙ */
void printTriangleType(int K) {
  switch(K){
  case 0:
    printf("ТРЕУГОЛЬНИК НЕ СУЩЕСТВУЕТ\n");
    break;
  case 1:
    printf("ТРЕУГОЛЬНИК ОБЩЕГО ВИДА\n");
    break;
  case 2:
    printf("ТРЕУГОЛЬНИК РАВНОБЕДРЕННЫЙ\n");
    break;
  case 3:
    printf("ТРЕУГОЛЬНИК РАВНОСТОРОННИЙ\n");
    break;
  }
}
int main( int argc, char **argv ){
  double a,b,c;
  int triangleExist, triangleKind;

  readTriangleSides(&a, &b, &c);
  triangleExist = isTriangleExist(a, b, c);
  if (triangleExist) {
    triangleKind = getTriangleKind(a, b, c);
  } else {
    triangleKind = 0;
  }
  printTriangleType(triangleKind);
  return 0;
}
```

4. Тесты

№	Входные данные	Ожидаемый результат	Действительный результат	Тест пройден
1	1 1 1	РАВНОСТОРОННИЙ	РАВНОСТОРОННИЙ	Да
2	1 2 3	НЕ СУЩЕСТВУЕТ	ОБЩЕГО ВИДА	Нет
3	3 4 5	ОБЩЕГО ВИДА	ОБЩЕГО ВИДА	Да
4	0 0 0	НЕ СУЩЕСТВУЕТ	РАВНОСТОРОННИЙ	Нет
5	4 4 5	РАВНОБЕДРЕННЫЙ	РАВНОБЕДРЕННЫЙ	Да
6	2 1 1	НЕ СУЩЕСТВУЕТ	РАВНОБЕДРЕННЫЙ	Нет
7	-1 -1 -1	НЕ СУЩЕСТВУЕТ	Runtime error 106 at 0BEB:0026.	Нет

Приложение 3. Сопоставление операторов Pascal и C

Примечание: в языке С регистр букв имеет значение. То есть, если “while” – это оператор, то “While” – это неизвестный идентификатор.

Таблица 2

Логические и бинарные операции

Наименование операции	Операция на языке Pascal	Операция на языке С
Отрицание	`NOT(A)`	`!(A)`
Логическое И	`A and B`	`A && B`
Логическое ИЛИ	`A or B`	`A || B`
Равенство	`A = B`	`A == B`
Неравенство	`A <> B`	`A != B`
Больше или равно	`A >= B`	`!(A < B)` или `A >= B`
Меньше или равно	`A <= B`	`!(A > B)` или `A <= B`
Побитовый сдвиг влево	`A chl n`	`A >> B`
Побитовый сдвиг вправо	`A chr n`	`A << B`
Побитовое И	`A and B`	`A & B`
Побитовое ИЛИ	`A or B`	`A | B`
Побитовое исключающее ИЛИ	`A xor B`	`A ^ B`

Таблица 3

Математические процедуры

Наименование операции	Операция на языке Pascal	Операция на языке С
Инкремент	`Inc(A)`	`A++`
Декремент	`Dec(A)`	`A--`
Увеличение на n	`Inc(A,n)`	`A+=n`
Уменьшение на n	`Dec(A,n)`	`A-=n`

Таблица 4

Алгоритмические операторы

Наименование операции	Операция на языке Pascal	Операция на языке С
Присваивание	`A:=B`	`A=B`
Составной оператор	`Begin … End;`	`{ … }`
Вывод данных	`Write(…);`	`printf(…);`
Ввод данных	`Read(...);`	`scanf(...);`
Условный оператор if	`IF "условие" Then "оператор1" Else "оператор2";`	`if ("условие") then "оператор1"; else "оператор2";`

Окончание таблицы 4

Наименование операции	Операция на языке Pascal	Операция на языке С
Оператор ветвления	Case "параметр" Of "список помеченных операторов" Else "оператор" End;	switch ("параметр") { case "вариант1": "оператор1"; break; case "вариант2": "оператор2"; break; ... default: "оператор" }
Оператор цикла for	For i:= N1 To N2 Do "оператор";	for (i=N1; i<N2+1; i++) "оператор"
Оператор цикла while	While "условие" DO "оператор";	while ("условие") "оператор"
Оператор цикла Repeat (do)	Repeat "операторы" Until "условие";	do "оператор" while ("условие")
Оператор ограничения и прерывания цикла	Continue; Break;	continue; break;

ОГЛАВЛЕНИЕ

ЯЗЫКИ ПРОГРАММИРОВАНИЯ

Задания для практических занятий
и методические указания по их выполнению

Техн. редактор *А.В. Миних*

Издательский центр Южно-Уральского государственного университета

Подписано в печать 19.12.2013. Формат 60×84 1/16. Печать цифровая.
Усл. печ. л. 1,86. Тираж 30 экз. Заказ 761/573.

Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, пр. им. В.И. Ленина, 76.