1. Основы алгоритмизации и программирования

1.1. Этапы подготовки и решения задач на ЭВМ

На ЭВМ могут решаться задачи различного характера: научно-инженерные; разработки системного программного обеспечения; обучения; управления производственными процессами и т. д. В процессе подготовки и решения на ЭВМ научно - инженерных задач можно выделить следующие ***этапы:***

- содержательная постановка задачи;
- математическая постановка (формализация) задачи;
- выбор и обоснование метода решения;
- алгоритмизация вычислительного процесса;
- составление программы;
- отладка программы;
- решение задачи на ЭВМ и анализ результатов.

В задачах разных классов некоторые этапы могут отсутствовать. Например, в задачах разработки системного программного обеспечения отсутствует математическая постановка задачи. Перечисленные этапы связаны друг с другом. Например, анализ результатов может показать необходимость внесения изменений в программу, алгоритм или даже в постановку задачи. Для уменьшения числа подобных изменений необходимо на каждом этапе по возможности учитывать требования, предъявляемые последующими этапами. В некоторых случаях связь между различными этапами, например, между постановкой задачи и выбором метода решения, между составлением алгоритма и программированием, может быть настолько тесной, что разделение их становится затруднительным.

Содержательная постановка задачи. На данном этапе формулируется цель решения задачи и подробно описывается ее содержание. Анализируются характер и сущность всех величин, используемых в задаче, и определяются условия, при которых она решается. Корректность постановки задачи является важным моментом, так как от нее в значительной степени зависят другие этапы.

Математическое описание задачи. Настоящий этап характеризуется математической формализацией задачи, при которой существующие соотношения между величинами, определяющими результат, выражаются посредством математических формул. Так формируется математическая модель процесса с определенной точностью, допущениями и ограничениями. При этом в зависимости от специфики решаемой задачи используются различные разделы математики и других дисциплин.

Математическая модель должна удовлетворять, по крайней мере, двум требованиям: реалистичности и реализуемости. Под ***реалистичностью*** понимается правильное отражение моделью наиболее существенных *черт* исследуемого явления.

Реализуемость достигается *разумной* абстракцией, отвлечением от второстепенных деталей, чтобы свести задачу к проблеме с известным решением. Условием реализуемости является возможность практического выполнения необходимых вычислений за отведенное время при доступных затратах требуемых ресурсов.

Выбор и обоснование метода решения. Модель с учетом ее особенностей должна быть доведена до реализации при помощи конкретных методов и алгоритмов решения. Само по себе математическое описание задачи в большинстве случаев трудно перевести на язык машины. Выбор и использование метода решения задачи позволяет привести процесс решения задачи к конкретным машинным операциям. При обосновании выбора метода необходимо учитывать различные факторы и условия, в том числе точность вычислений, время решения задачи на ЭВМ, требуемый объем памяти и другие.

Одну и ту же задачу можно решить различными методами, при этом в рамках каждого метода можно составить различные алгоритмы.

Алгоритмизация вычислительного процесса. На данном этапе составляется алгоритм решения задачи согласно действиям, задаваемым выбранным методом решения. Процесс обработки данных разбивается на отдельные относительно самостоятельные блоки, и устанавливается последовательность их выполнения. Разрабатывается блок-схема алгоритма.

Составление программы. При составлении программы алгоритм решения задачи переводится на конкретный язык программирования. Для программирования обычно используются языки высокого уровня, поэтому составленная программа требует перевода ее на машинный язык. После такого перевода выполняется уже соответствующий машинный код.

Отладка и тестирование программы. Отладка заключается в поиске и устранении синтаксических и логических ошибок в программе.
В ходе синтаксического контроля программы транслятором выявляются конструкции и сочетания символов, недопустимые с точки зрения правил их построения или написания, принятых в данном языке. Сообщения об ошибках ЭВМ выдает программисту, при этом вид и форма выдачи подобных сообщений зависят от вида языка и версии используемого транслятора.
После устранения синтаксических ошибок проверяется логика работы программы в процессе ее выполнения с конкретными исходными данными. Для этого используются специальные методы, например, в программе выбираются контрольные точки, для которых вручную рассчитываются промежуточные результаты. Эти результаты сверяются со значениями, получаемыми ЭВМ в данных точках при выполнении отлаживаемой программы. Кроме того, для поиска ошибок могут быть использованы отладчики, выполняющие специальные действия на этапе отладки, например, удаление, замена или вставка отдельных операторов или целых фрагментов программы, вывод или изменение значений заданных переменных.

Решение задачи на ЭВМ и анализ результатов. После отладки программы ее можно использовать для решения прикладной задачи. При этом обычно выполняется многократное решение задачи на ЭВМ для различных наборов исходных данных. Получаемые результаты интерпретируются и анализируются специалистом или пользователем, поставившим задачу.
Разработанная программа длительного использования устанавливается на ЭВМ, как правило, в виде готовой к выполнению машинной программы. К ней прилагается документация, включающая инструкцию для пользователя.
Чаще всего, при установке программы на диск для ее последующего использования помимо файлов с исполняемым кодом устанавливаются различные вспомогательные программы (утилиты, справочники, настройщики и т. д.), а также необходимые для работы программы разного рода файлы с текстовой, графической, звуковой и другой информацией.

1.2. Алгоритмы и способы их описания

Понятие алгоритма

Для составления программы, предназначенной для решения на ЭВМ какой-либо задачи, требуется составить алгоритма ее решения.

Алгоритм — это точное предписание, которое определяет процесс, ведущий от исходных данных к требуемому конечному результату. Алгоритмами, например, являются правила сложения, умножения, решения алгебраических уравнений, умножения матриц и т.п. Слово алгоритм происходит от *algoritmi*, являющегося латинской транслитерацией арабского имени хорезмийского математика IX века **аль-Хорезми**. Применительно к ЭВМ алгоритм определяет вычислительный процесс, начинающийся с обработки некоторой совокупности возможных исходных данных и направленный на получение определенных этими исходными данными

результатов. Термин ***вычислительный процесс*** распространяется на обработку числовой и других видов информации, например, символьной, графической или звуковой.
Если вычислительный процесс заканчивается получением результатов требуемой точности, то говорят, что соответствующий алгоритм применим к рассматриваемой совокупности исходных данных. В противном случае говорят, что алгоритм неприменим к совокупности исходных данных. Любой применимый алгоритм обладает следующими ***основными свойствами:***

- результативностью или конечностью алгоритма;
- определенностью;
- массовостью.

Результативность означает возможность получения результата после выполнения конечного количества операций.

Определенность состоит в совпадении получаемых результатов независимо от пользователя и применяемых технических средств.

Массовость заключается в возможности применения алгоритма к целому классу однотипных задач, различающихся конкретными значениями исходных данных.

Для задания алгоритма необходимо описать следующие его элементы:

- набор объектов, составляющих совокупность возможных исходных данных, промежуточных и конечных результатов;
- правило начала;
- правило непосредственной переработки информации (описание последовательности действий);
- правило окончания;
- правило извлечения результатов.

Алгоритм всегда рассчитан на конкретного исполнителя. В нашем случае таким исполнителем является ЭВМ. Для обеспечения возможности реализации на ЭВМ алгоритм должен быть описан на языке, понятном компьютеру, то есть на языке программирования.

Таким образом, можно дать следующее определение программы.

Программа для ЭВМ представляет собой описание алгоритма и данных на некотором языке программирования, предназначенное для последующего автоматизированного выполнения.

Способы описания алгоритмов

К основным способам описания алгоритмов можно отнести следующие:

- словесно-формульный;
- структурный или блок-схемный;
- с помощью граф-схем;
- с помощью сетей Петри.

Перед составлением программ чаще всего используются словесно-формульный и блок-схемный способы. Иногда перед составлением программ на низкоуровневых языках программирования типа языка Ассемблера алгоритм программы записывают, пользуясь конструкциями некоторого высокоуровнего языка программирования. Удобно использовать программное описание алгоритмов функционирования сложных систем. Так, для описания принципов функционирования ОС использовался Алголоподобный высокоуровневый язык программирования.

При ***словесно-формульном способе*** алгоритм записывается в виде текста с формулами по пунктам, определяющим последовательность действий.
Пусть, например, необходимо найти значение следующего выражения:

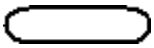
y = 2a – (x+6).

Словесно-формульным способом алгоритм решения этой задачи может быть записан в следующем виде:

1. Ввести значения *a* и *x.*
2. Сложить x и 6.
3. Умножить *a* на 2.
4. Вычесть из *2a* сумму (x+6).
5. Вывести *y* как результат вычисления выражения.

При ***блок-схемном*** описании алгоритм изображается геометрическими фигурами (блоками), связанными по управлению линиями (направлениями потока) со стрелками. В блоках записывается последовательность действий. Данный способ по сравнению с другими способами записи алгоритма имеет ряд преимуществ. Он наиболее нагляден: каждая операция вычислительного процесса изображается специальной геометрической фигурой. Кроме того, графическое изображение алгоритма наглядно показывает разветвления путей решения задачи в зависимости от различных условий, повторение отдельных этапов вычислительного процесса и другие детали.
Оформление программ должно соответствовать определенным требованиям. В настоящее время действует единая система программной документации (ЕСПД), которая устанавливает правила разработки, оформления программ и программной документации. В ЕСПД определены и правила оформления блок-схем алгоритмов (ГОСТ 10.002-80 ЕСПД, ГОСТ 10.003-80 ЕСПД).
Операции обработки данных и носители информации изображаются на схеме соответствующими ***блоками.*** Большая часть блоков по построению условно вписана в прямоугольник со сторонами *a и b.* Минимальное значение ***a* = 10 мм**, увеличение *a* производится на число, кратное **5 мм**. Размер **b=1,5a**. Для отдельных блоков допускается соотношение между *a и b,* равное 1:2. В пределах одной схемы рекомендуется изображать блоки одинаковых размеров. Все блоки нумеруются. Виды и назначение основных блоков приведены в табл.1.

Таблица 1. Условные обозначения блоков схем алгоритмов

Наименование	Обозначение	**Функции**
Процесс		Выполнение операции или группы операций, в результате которых изменяется значение, форма представления или расположение данных.
Ввод-вывод		Преобразование данных в форму, пригодную для обработки (ввод) или отображения результатов обработки (вывод).
Решение		Выбор направления выполнения алгоритма в зависимости от некоторых переменных условий.
Предопределенный процесс		Использование ранее созданных и отдельно написанных программ (подпрограмм).
Пуск-останов		Начало, конец, прерывание процесса обработки данных.
Соединитель		Указание связи между прерванными линиями, соединяющими блоки.
Межстраничный соединитель		Указание связи между прерванными линиями, соединяющими блоки, расположенные на разных листах.
Комментарий		Связь между элементом схемы и пояснением.

Линии, соединяющие блоки и указывающие последовательность связей между ними, должны проводиться параллельно линиям рамки. Стрелка в конце линии может не ставиться, если линия направлена слева направо или сверху вниз. В блок может входить несколько линий, то есть блок может являться преемником любого числа блоков. Из блока (кроме логического) может выходить только одна линия. Логический блок может иметь в качестве продолжения один из двух блоков, и из него выходят две линии. Если на схеме имеет место слияние линий, то место пересечения выделяется точкой. В случае, когда одна линия подходит к другой и слияние их явно выражено, точку можно не ставить.

Схему алгоритма следует выполнять как единое целое, однако в случае необходимости допускается обрывать линии, соединяющие блоки. Если при обрыве линии продолжение схемы находится на этом же листе, то на одном и другом конце линии изображается специальный символ ***соединитель*** — окружность диаметром **0,5 а.** Внутри парных окружностей указывается один и тот же идентификатор. В качестве идентификатора, как правило, используется порядковый номер блока, к которому направлена соединительная линия.

Если схема занимает более одного листа, то в случае разрыва линии вместо окружности используется ***межстраничный соединитель.*** Внутри каждого, соединителя указывается адрес — откуда и куда направлена соединительная линия. Адрес записывается в две строки: в первой указывается номер листа, во второй — порядковый номер блока.

Блок-схема должна содержать все разветвления, циклы и обращения к подпрограммам, содержащиеся в программе.

1.3. Компиляция и интерпретация программ

ЭВМ непосредственно выполняет программы на ***машинном языке*** данной ЭВМ. При этом программа представляет собой последовательность отдельных команд компьютера. Эти команды являются достаточно «простыми», например, сложение, умножение, сравнение или пересылка отдельных данных. Каждая команда содержит в себе сведения о том, какая операция должна быть

выполнена (код операции), с какими операндами выполняются вычисления (адреса данных или непосредственно сами данные) и куда (адрес) должен быть помещен результат.

Машинные языки были первыми языками программирования. Программирование на них затруднительно ввиду того, что, во-первых, эти языки различны для каждого типа ЭВМ, во-вторых, являются трудоемкими для большинства пользователей по причине необходимости знания особенностей конкретной ЭВМ и большого количества реализуемых ею операций (команд). Данные языки обычно используются для разработки системных программ, при этом чаще всего применяются специальные символические языки — Ассемблеры, близкие к соответствующим машинным языкам.

Человеку свойственно формулировать и решать задачи в выражениях более общего характера, чем команды ЭВМ. Поэтому с развитием программирования появились языки, ориентированные на более высокий уровень абстракции при описании решаемой на ЭВМ задачи. Эти языки получили название языков высокого уровня. Их теоретическую основу составляют алгоритмические языки, например, Паскаль, Си, Бейсик, Фортран, PL/1.

Для перевода программы, написанной на языке высокого уровня, в соответствующую машинную программу используются ***языковые процессоры.*** Различают два вида языковых процессоров: интерпретаторы и трансляторы.

Интерпретатор — это программа, которая получает исходную программу и по мере распознавания конструкций входного языка реализует действия, описываемые этими конструкциями.

Транслятор — это программа, которая принимает исходную программу и порождает на своем выходе программу уже на другом языке программирования (например, объектную программу). В частном случае, объектным может оказаться машинный язык, и в этом случае полученную на выходе транслятора программу можно сразу же выполнить на ЭВМ. В общем случае объектный язык необязательно должен быть машинным или близким к нему (автокодом). В качестве объектного языка может служить и некоторый промежуточный язык.

Транслятор с языка высокого уровня называют ***компилятором.***

1.4. Стили программирования

Одним из важнейших признаков классификации языков программирования является принадлежность их к одному из стилей, основными из которых являются следующие: процедурный, функциональный, логический и объектно-ориентированный.

Процедурное программирование

Процедурное (императивное) программирование является отражением архитектуры традиционных ЭВМ, которая была предложена фон Нейманом в 40-х годах. Программа на процедурном языке программирования состоит из последовательности операторов (инструкций), задающих процедуру решения задачи.

Выполнение программы сводится к последовательному выполнению операторов с целью преобразования исходного состояния памяти, то есть значений исходных данных, в заключительное, то есть в результаты. Таким образом, с точки зрения программиста имеются программа и память, причем первая последовательно обновляет содержимое последней.

Процедурные языки характеризуются следующими особенностями:

- необходимостью явного управления памятью, в частности, описанием переменных;
- малой пригодностью для символьных вычислений;
- отсутствием строгой математической основы;
- высокой эффективностью реализации на традиционных ЭВМ.

Одним из важнейших классификационных признаков процедурного языка является его уровень. Уровень языка программирования определяется семантической (смысловой) емкостью его конструкций и степенью его ориентации на программиста. Язык программирования частично ликвидирует разрыв между методами решения различного рода задач человеком и вычислительной машиной. Чем более язык ориентирован на человека, тем выше его уровень. Дадим краткую характеристику реализованным на ПЭВМ языкам программирования в порядке возрастания их уровня.

Язык программирования C (Си) первоначально был разработан для реализации операционной системы UNIX в начале 70-х годов. В последующем приобрел высокую популярность среди системных и прикладных программистов. В настоящее время этот язык реализован для большинства ЭВМ.

В C сочетаются достоинства современных высокоуровневых языков в части управляющих конструкций и структур данных с возможностями доступа к аппаратным средствам ЭВМ на уровне, который обычно ассоциируется с языком низкого уровня типа языка Ассемблера. Язык C имеет синтаксис, обеспечивающий краткость программы, а компиляторы способны генерировать эффективный объектный код.

Одна из наиболее существенных особенностей C состоит в нивелировании различий между выражениями и операторами, что приближает его к функциональным языкам. В частности, выражение может обладать побочным эффектом присваивания, а также может использоваться в качестве оператора. Нет также четкой границы между процедурами и функциями, более того, понятие процедуры не вводится вообще.

Синтаксис языка затрудняет программирование и восприятие составленных программ. Отсутствует и строгая типизация данных, что предоставляет дополнительные возможности программисту, но не способствует написанию надежных программ.

Язык C++ был разработан в начале 80-х годов Б. Страуструпом, сотрудником лаборатории Bell корпорации AT&T. Им была создана компактная компилирующая система, в которой за основу был взят язык C, дополненный элементами языков BCPL, Simula-67 и Algol-68. К июлю 1983 года появился язык C с классами, а чуть позднее — C++. К 1990 году была выпущена третья версия языка C++, принятая комитетом ANSI в качестве исходного материала для его стандартизации.

2. Основы алгоритмического языка C/C++

2.1. Структура основной программы

Простейшая программа на C/C++

Такая программа состоит всего из 12 символов, но заслуживает внимательного рассмотрения. Вот она:

```
void main()
{

}
```

Основная программа в C/C++ всегда называется именем main (будьте внимательны – C/C++ различает большие и маленькие буквы, а все стандартные операторы C/C++ записываются маленькими буквами). Пустые скобки означают, что main не использует аргументов, а слово void (пустой) говорит о том, что она также и не возвращает никакого значения, то есть, является ***процедурой***.
Фигурные скобки обозначают начало и конец процедуры main - поскольку внутри них ничего нет, наша программа ничего не делает, она просто соответствует правилам языка C/C++, ее можно скомпилировать и получить **исполняемый** файл.

Составим теперь программу, которая делает что-нибудь полезное, например, выводит на экран слово «Привет».

```
#include "stdio.h"  /*подключение заголовочного файла stdio.h, в котором
находится описание стандартных функций ввода и вывода, */

int  main()
{
printf("Привет");   // вызов функции вывода на экран
return 0;

}
```

Что новенького?

Перечислим новые элементы, использованные в этой программе:

Для вывода информации на экран используется функция printf. В простейшем случае она принимает единственный аргумент - строку в кавычках, которую надо вывести на экран.

Чтобы использовать стандартные функции, необходимо сказать компилятору, что есть функция с таким именем и перечислить тип ее аргументов - тогда он сможет определить, верно ли мы ее используем. Это значит, что надо подключить ***описание*** этой функции.

Описания стандартных функций C++ находятся в так называемых ***заголовочных файлах*** с расширением ***.h**

Для подключения заголовочных файлов используется директива (команда) ***препроцессора*** "***#include***", после которой в угловых скобках или кавычках ставится имя файла с заголовками. Все команды препроцессора начинаются символом "***#***" в начале строки. Для подключения каждого нового заголовочного файла надо использовать новую команду "***#include***".

В программу могут быть включены комментарии - тексты, поясняющие программу. Комментарии – не обязательная принадлежность программы. Описание нетривиального алгоритма, как правило, снабжается пояснениями, которые помогут разобраться в тексте программы персоналу сопровождения программных продуктов. В языке C/C++ предусматривается две разновидности комментариев – *многострочные* и *однострочные.*

Однострочный комментарий начинается вслед за парой символов "/ /" и продолжается до конца программной строки. Обычно с его помощью записывается комментарий к текущей строке или исключается фрагмент текущей строки из области обслуживания. Комментарии, заключенные в /* */ и записываются в любом месте программы и могут располагаться на нескольких строках.

2.2. Алфавит языка

При написании программ на алгоритмическом языке можно пользоваться лишь символами, предусмотренными алфавитом этого языка.

Алфавит языка C/C++ включает в себя следующие символы.

Строчные и прописные буквы латинского алфавита

|a|b|c| ... | x | y | z | A | B | C | . . . | X | Y | Z |

Арабские цифры

|0|1 |2| ...|9|

Специальные символы

| + | - | * | / | =| >| >=| < | <=| ==| ! | . | , | : | ; | ’ | (|) | [|] | { | } | & | @ | $ | # |

Язык C/C++ также включается набор зарезервированных слов, имеющих строго определенное назначение. Например, слова IF, ELSE используются в условном операторе, слово DEFAULT обозначает выбор по умолчанию.

2.3. Константы и переменные

Константы

Константами являются программные элементы, имеющие определенный тип и не меняющие своего значения при выполнении программы. В качестве констант на языке C/C++ используются целые и вещественные числа, логические значения, символы и строки.

Целые числа записываются со знаком или без него по обычным арифметическим правилам.

Например: 15 +1000 -47 02

Вещественные числа могут записываться либо в форме десятичного числа, либо в экспоненциальной форме.

В записи десятичного числа целая часть отделяется от дробной части точкой.

Например: 2.5 -14.0 +0.33 0.0

Числа в экспоненциальной форме имеют вид: $a \cdot 10^p$, где a является мантиссой; p - десятичным порядком.

На языке C/C++ числа с порядком записываются с использованием буквы E, за которой следует порядок. Буква E читается как «умножить на 10 в степени».

Мантисса может быть целым или десятичным числом. Порядок всегда является целым числом. Следует помнить, что в написании вещественного числа с порядком должны обязательно присутствовать мантисса и порядок.

Примеры записи чисел с порядком

Математическая запись	Запись на языке C/C++
$3{,}14 \cdot 10^5$	3.14E5
$-17 \cdot 10^{-2}$	-17E-2
25,625	2.5625E1
10^{-6}	1E-6
0,00048	4.8E-4

Символьная константа - это любой символ ПК, заключенный в апострофы, например: ‘7’ или ‘S’ или ‘ *’.

Строковая константа (строка) - любая последовательность символов из набора символов ПК, заключенная в кавычки, например:

"X="

"Максимальное значение ="

Строки используются также при выводе текстов и оформлении комментариев.

Переменные

Программа оперирует информацией, представленной в виде различных объектов и величин. ***Переменная*** – это символическое обозначение величины в программе. Как ясно из названия, **значение** *переменной* (или величина, которую она обозначает) во время выполнения программы может изменяться.

С точки зрения архитектуры компьютера, ***переменная*** – это символическое обозначение ячейки оперативной памяти программы, в которой хранятся данные. Содержимое этой ячейки – это текущее значение *переменной*.

В языке C/C++ прежде чем использовать *переменную*, ее необходимо **объявить.**

В объявлении переменной первым стоит название *типа* переменной, а затем *идентификатор* или *имя переменной*. *Тип* ***переменной*** определяет, какие возможные значения эта *переменная* может принимать и какие операции можно выполнять над данной *переменной*. Тип *переменной* изменить нельзя, т.е. пока *переменная* существует, она всегда будет описанного типа.

Имя представляет собой последовательность латинских букв и цифр, которая начинается с буквы. В имени может присутствовать символ подчеркивания. Длина имени составляет от 1 до 63 символов. Следует помнить, что пробелы не должны входить в написание имени.

Примеры записи имен переменных

Математическая запись	***x***	a_p	y^1	α	∑	*d-21*
Запись на языке С/С++	x	ap	y1	alpha	S	d_27

С точки зрения компьютера все данные в памяти - это числа (более точно - наборы нулей и единиц). Тем не менее, и вы (и компьютер) знаете, что с целыми и дробными числами работают по-разному. Поэтому в каждом языке программирования есть разные типы данных (переменных), для обработки которых используются разные методы. Основными данными в языке С/С++ являются

- целые переменные (тип int - от английского *integer* - целый) занимают 2 или 4 байта в памяти;

- вещественные переменные, которые могут иметь дробную часть (тип float – от английского *floating point* - плавающая точка), занимают 4 байта в памяти;

-символы (тип char - от английского *character* - символ) занимают 1 байт в памяти.

Типы задаются стандартными зарезервированными словами:

int - целый тип;
long – длинный целый тип;
short – целый тип с меньшим диапазоном;
float - вещественный тип;
double - вещественный тип с двойной точностью;
char - символьный тип;

Для использования все переменные необходимо объявлять - то есть сказать компьютеру, чтобы он выделил для них ячейку памяти нужного размера и присвоил ей нужное имя. Переменные обычно объявляются в начале программы. Для объявления надо написать название типа переменных (int, float или char и др.), а затем через запятую имена всех объявляемых этим типом переменных. При желании можно сразу записать в новую ячейку нужное число, как показано в примерах ниже. Если переменной не присваивается никакого значения, то в ней находится "мусор", то есть то, что было там раньше.

По описанию переменной в памяти компьютера резервируется ячейка для хранения ее значения. В зависимости от объявленного *типа* переменной ячейка может иметь разную внутреннюю структуру, т.е. содержать различное число байт.

Примеры.

```
int a;       // выделить память под целую  переменную a
float    b, c;  // две вещественных переменных b и c
int    Tu104, Dl86=23, Yak42;     // три целых переменных,
                          //причем в D186  сразу записывается число 23.
float    x=4.56, y, z;             // три вещественных переменных,
                        // причем в x сразу записывается число 4.56.
char   c, c2='A', m;          // три символьных переменных,
                        //причем в c2  сразу записывается символ 'A'.
```

2.4. Арифметические выражения

Арифметические выражения строятся из операндов, арифметических операций и круглых скобок. Операндами могут быть константы, переменные и функции.

В бесскобочных арифметических выражениях операции выполняются слева направо в соответствии с их приоритетом.

1. * (умножение); / (деление); % (остаток от деления целых чисел).
2. + (сложение); - (вычитание).

Изменить порядок выполнения операций можно с помощью *круглых* скобок. Выражение, заключенное в круглые скобки, выполняется в первую очередь. Например, выражению: a/b*c соответствует математическая запись: $\frac{a}{b}c$, а выражению a/(b*c) – запись $\frac{a}{bc}$.

Тип арифметического выражения определяется типом входящих и него операндов.

Арифметическое выражение является целым, если все входящие и него операнды целого типа.

Если в арифметическом выражении содержится хотя бы один вещественный операнд, то результат - вещественный. Целые операнды в вещественном арифметическом выражении всегда преобразуются к вещественному типу.

Операция выделения остатка или деление по модулю (%) применима только к целым числам. Результат ее выполнения имеет целый тип.

Исключение составляет операция деления с использованием символа ‘/’ (косая черта). Результат выполнения этой операции всегда зависит от типа операндов.

Например, значением выражения 2/5 будет число 0.

Примеры вычисления арифметических выражений

Арифметические выражения	Результат	Тип результата
6 + 4 * (5 - 3)	14	Целый
6 + 4 * (5 - 3.0)	14.0	Вещественный
7 % 2	1	Целый
7/2.0	3.5	Вещественный

Стандартные функции

Часто используемые в арифметических выражениях математические функции оформлены в виде стандартных подпрограмм, которые хранятся в библиотеке системы C/C++ (файл с именем math.h). Для правильного обращения к стандартной функции необходимо записать имя функции, за которым в круглых скобках следует аргумент (параметр). Приоритет вычисления функций выше, чем приоритет арифметических операций.

В таблице ниже представлен набор стандартных функций с указанием типов функции и аргумента. В таблице приняты обозначения: I - целый тип, F - вещественный тип.

Математическая запись	Запись на языке С/С++	Тип аргумента	Тип функции
$\lvert x\rvert$	**abs(x)** **fabs(x)**	I F	I F
tg x	**tan(x)**	I или F	F
sin x	**sin(x)**	I или F	F
cos x	**cos(x)**	I или F	F
ln x	**log(x)**	I или F	F
$\sqrt{x}$	**sqrt(x)**	I или F	F
arctg(x) - арктангенс	**atan(x)**	I или F	F
arcctg(x)- арккотангенс arcctg(x)=$\frac{\pi}{2} - arctg(x)$	3.14/2.0-atan(x)	F	F
Округление до ближайшего целого > x	**ceil(x)**	F	I
Округление до ближайшего целого < x	**floor(x)**	F	I
Возведение в степень x^n	**pow(x,n)**	I F	I F
e^x	**exp(x)**	I или F	F
arcsin(x) - арксинус	**asin(x)**	I или F	F
arccos(x) - арккосинус	**acos(x)**	I или F	F
$\log_{10}(x)$ десятичный логарифм.	**log10(x)**	I или F	F
$\mathrm{sh}(x) = \frac{e^x - e^{-x}}{2}$ гиперболический синус	**sinh(x)**	I или F	F
$\mathrm{ch}(x) = \frac{e^x + e^{-x}}{2}$ гиперболический косинус	**cosh(x)**	I или F	F
$\mathrm{th}(x) = \frac{e^x - e^{-x}}{e^x - e^{-x}}$ гиперболический тангенс	**tanh(x)**	I или F	F
Округление по правилам математики	**round(x)**	I или F	I
Генерация случайного числа	**rand()** **rand() /x**	нет x - F	I F

Примеры программирования арифметических выражений

Математическая запись	Запись на языке C/C++
$\frac{a+12b}{c_1-1{,}8\cdot 10^3}$	(a+12*b)/(c1 - 1.8E3)
$e^{\frac{2\sin 4x+\cos^2 x^2}{3x}}$	exp((2*sin(4*x)+pow((cos(x* x),2)))/(3*x))
$\ln\lvert tg\alpha-\sin\alpha^3\rvert$	log(fabs(tan(al)-sin(pow(al,3))))

Контрольные задания

1. Составить описания для заданных переменных

Вариант	Математическая запись	Тип	Вариант	Математическая запись	Тип
1	C_2,Lt, j , Z_1, T, rs	Целый Символьный	9	Sum, max, w T, Z_1, R	Вещественный Целый
2	M, n_1, a_p, B, γ_0, η	Целый Вещественный	10	A C_3, k_b, β_1, X_1, $\sum$	Символьный Вещественный
3	Δ, h, E, st, w, y_1	Вещественный Символьный	11	I, d_0, K_H Ch, Q, rez	Целый Символьный
4	T3, j, x , φ, w, Z_1	Целый Вещественный	12	N, Kc, a_2 Z,fd, η	Целый Символьный
5	N, x_1, b, A, η, min	Целый Вещественный	13	I, kd, l_1 ϑ,v,T_n	Символьный Вещественный
6	J, I, K, F, S, R_2	Целый Символьный	14	M_1, j, r, b_0,A, σ	Символьный ВещественныйI
7	β, y_4, Ω, n, a_0, k	Вещественный Символьный	15	Mo, j, r_{cm} E, £, f_1	Целый Вещественный
8	λ, σ, δ, l, k_1, m	Вещественный Символьный	16	α, μ, c str, P_1,φ	Вещественный Символьный

2. Записать на языке C++ числовые константы

а) целые;

б) вещественные (в форме десятичных чисел и чисел с порядком).

Вариант	Математическая запись
1	а)+15; $72{,}93\cdot 10^3$; -500 б) $\frac{1}{8}$; -16,7; $0{,}054\cdot 10^2$; $62{,}7\cdot 10^{-5}$
2	а) 02; $32\cdot 10^3$; $-8\cdot 10^2$ б) -0,25; $2{,}89\cdot 10^2$; 3,164; $23{,}7\cdot 10^{-4}$

3	а) 600; -041; $52{,}625\cdot10^{4}$ б) $\frac{2}{3}$; $0{,}03\cdot10^{-2}$; -715,4; $2{,}1\cdot10^{3}$
4	а)735; $-24\cdot10^{3}$; $13{,}14\cdot10^{2}$ б) $0{,}0265\cdot10^{3}$; -3,7; 54,6; $0{,}275\cdot10^{-2}$
5	а)014; $25\cdot10°$; $27{,}3\cdot10^{3}$ б) 0,518; $2{,}546\cdot10^{2}$; $-0{,}07\cdot10^{-3}$; 3,04
6	а)01; $25\cdot10$; $2{,}73\cdot10^{2}$ б) 15,37; $4{,}52\cdot10^{-3}$; 23; 0,0083
7	а)07; $94\cdot10^{2}$; $16{,}27\cdot10^{3}$ б) 8,4; $-35{,}07\cdot10^{-1}$; $6{,}3\cdot10^{2}$; 0,004
8	а) 41; $286\cdot10°$; $23{,}7\cdot10^{2}$ б) 0,817; $0{,}645\cdot10^{2}$; $-0{,}718\cdot10^{3}$; $0{,}3556\cdot10°$
9	а)377; $-15\cdot10^{3}$; $25{,}4\cdot10^{3}$ б) $-0{,}572\cdot10^{6}$; 6.28; 5886; $0{,}0695\cdot10^{2}$
10	а)012; $7{,}5\cdot10^{3}$; $-0{,}05\cdot10^{2}$ б) $-0{,}084\cdot10^{-2}$; 7,12; -52; 0,0085
11	а)0015; $-9{,}4\cdot10^{3}$; 99,0 б)-5,14; $27{,}06\cdot10^{-6}$; $\frac{1}{4}$; $0{,}008\cdot10^{2}$
12	а) 276; $9{,}275\cdot10^{3}$; -02 б) $-0{,}56\cdot10^{6}$; -0,07425; 6; -0,873
13	а) 03; $52\cdot10°$; $-1{,}0\cdot10^{3}$ б) $\frac{1}{5}$; $631{,}5\cdot10^{2}$; -8,25; $92{,}1\cdot10^{4}$
14	а)-016; $95{,}72\cdot10^{2}$; -576 б)-2,75; $81{,}2\cdot10^{-6}$; $-15{,}0\cdot10^{3}$; $0{,}033\cdot10^{-3}$
15	а) 05; $-91{,}0\cdot10^{3}$; $14{,}5\cdot10^{2}$ б) 3,48; $1{,}5\cdot10^{2}$; $-912\cdot10^{-5}$; 0,0385
16	а) 008; $7{,}2\cdot10^{4}$; $-12\cdot10^{2}$ б)54; 17,8; $2{,}041\cdot10^{-3}$; $513\cdot10^{3}$

3. Записать на языке C/C++ арифметические выражения

Вариант	Математическая запись
1	$\frac{3\cos^2(x-\frac{\pi}{6})}{0{,}5+\sin y^2}$
2	$\frac{\lvert x-y\rvert}{(1+2x)^2}-e^{\sqrt{1+\mu}}$
3	$\frac{5a^{nx}}{b+c}-\sqrt{\lvert\cos x^3\rvert}$

4	$\ln\left\|a^7\right\| + arctgx^2 + \frac{\pi}{\sqrt{\left\|a+x\right\|}}$
5	$\sqrt[4]{\frac{(a+b)^2}{c+d}} + e^{\sqrt{x+1}}$
6	$\frac{1}{2\pi}\sqrt{\frac{2mg}{m(a\sin\alpha + b\cos\alpha)}}$
7	$\frac{1}{4}\left(\frac{1+x^2}{1-x} + \frac{1}{2}tgx\right)$
8	$a + \sqrt{b} + \ln\left\|x^3\right\| + e^{\sqrt[3]{c}}$
9	$\left\|\sin\alpha\right\|\cos\frac{\alpha}{2} + \sqrt{a^2+b^2}$
10	$\sqrt[5]{x}(arctgz + \cos^2 y)$
11	$\sqrt[4]{y + \sqrt[3]{x-y}}$
12	$\frac{2}{\sqrt{x}} \cdot \frac{x-\lambda}{\sqrt[3]{(x-\lambda)^2 + \lambda^2}}$
13	$e^{x\cos\frac{\pi}{4}}\cos(x\sin\frac{\pi}{4})$
14	$\sqrt{\left\|\ln\left(1 - 2x\cos\frac{\pi}{3}\right) + x^2\right\|}$
15	$\sqrt{a_0 + a_1 x + a_2\sqrt[3]{\left\|\sin x\right\|}}$
16	$\frac{tgx}{x} + \sqrt[3]{\sin x^3}$

4. Записать в виде алгебраической формулы

Вариант	Запись на языке C/C++
1	log(abs(x*x*x))+atan(x)+exp(omega)
2	x*sin(3.14/4.0)/(1-2*x*cos(3.14/4.0)+exp(1/3*log(x)))
3	sqrt(fabs(log(sin(x)+2)))
4	3*sin(x)+1/3*log(1+x*x)+sqrt(exp(x))
5	sqrt(abs(x-y))/(pow(sin(z),2)+1)
6	exp(abs(x+y))*exp(x*log(1 +sin(y)))

7	2*((x+y+z)*pow((x+y+z),2)/(2*x)-sin(x*x*x))
8	(sin(al)+atan(al))/(omega+pow(cos(al),2))
9	exp(cos(x))*pow(cos(sin(x)),2)+5.7E-1
10	exp(x)/(4E-1 +x*x)-sin(log(x))
11	4*a*a+x*(b*b*b+x*(pow(c,4)+x*(a-exp(x))))
12	18*a*x*x+exp(2/3*log(x))+sin(al/2)
13	2*b*b*b+log(a)-exp(2.5*log(x))
14	(x+y+z)/(pow(sin(a*x),2)+b*b)
15	exp(y/x*log(abs(x)))+exp(1/3*log(y/x))
16	atan(x*x*x)/(log(x)+sin(x)/cos(x))+1

5. Вычислить арифметическое выражение

Вариант	Запись на языке С/С++	Исходные
1	18.75 - 16.4E0+1/4	-
2	floor(sqrt(a+b))	*a* = 0,84; *b* = 0,16
3	3/x+x*x	*x = 2*
4	(x*x*t+2.1)*ceil(1/4)+5	*x = 36; t* =5,287
5	i+3-i*pow(i,2)	i = -3
6	4+exp(ceil(3/2)*log(6.25E-1))	-
7	1%3+1/3	-
8	2*10/4	. -
9	x/(y*x-y)+x/y+1.2E1	x = 3; y= 1,5
10	19/4+1%4	-
11	100*(99/100)	-
12	m+k*((m+k)/n)-n	*M=2; k=7; n*= 10
13	2*floor(4/10)+2E0	-
14	2.0*(1E5)	-
15	19/4+5%4	-
16	exp(1/3*log(27))+2	-

2.5. Линейные вычислительные процессы

Линейный вычислительный процесс представляет собой последовательность действий, выполняемых одно за другим. Основу программы линейного вычислительного процесса составляют оператор присваивания и операторы ввода-вывода данных.

Оператор присваивания

Оператор присваивания служит для вычисления выражения и записи результата в память компьютера.

Общий вид записи оператора

переменная=выражение;

Знак = читается как «присвоить». Конец любого оператора на языке С/С++ фиксируется точкой с запятой.

Работа оператора присваивания

При выполнении оператора присваивания происходит замена текущего значения переменной, стоящей слева от знака присваивания, новым значением, полученным в результате вычисления выражения, стоящего справа от знака равенства, например

c = a + b; // вычислит сумму ***a*** и ***b*** и запишет результат в ***c.***

Тип результата выполнения оператора

При записи оператора присваивания необходимо строго следить за типом переменной, стоящей в левой части оператора, и типом выражения:

- если переменная слева - вещественного типа, то арифметическое выражение может быть как целым, так и вещественным.

В случае целого арифметического выражения происходит преобразование его результата к вещественному типу. Например, при выполнении оператора

m=20+15/4.0;

вещественной переменной ***m*** будет присвоено значение 23.0;

- если переменная слева - целого типа, то арифметическое выражение будет приведено к целому типу. Например, задано следующее описание типа переменных.

```
float a, b, c;
int n;
```

Тогда записать оператор присваивания в виде: n=a*b/c; приведет к усечению результата до целого типа.

В операторе присваивания конечный результат вычисления выражения в правой части приводится к типу переменной, которой должно быть присвоено это значение. Данный процесс может привести к *понижению типа*, как описано выше, или к повышению, при котором величина приводится к типу данных, имеющему более высокий приоритет. Последовательность имен типов, упорядоченных от высшего типа к низшему типу, выглядит так: double, float, long, int, short, char. Применение ключевого слова unsigned повышает ранг соответствующего типа данных со знаком. *Повышение типа* обычно происходит гладко, в то время как понижение может привести к затруднениям (скрытым ошибкам). Запомним, что тип значения выражения в правой части оператора присваивания всегда преобразуется к типу переменной, которой присваивается это значение.

Примеры использования оператора присваивания

1. Записать оператор присваивания, который позволяет вычислить расстояние между двумя точками на плоскости с координатами *(x1, y1) и (x2, y2).*

Ответ: `d=sqrt(pow(x1-x2,2)+pow(y1-y2,2));`

2. Записать последовательность операторов присваивания, обеспечивающих обмен значениями переменных *x* и *y* в памяти компьютера.

Ответ :
```
c=x;
x=y;
y=c;
```

Странные операторы присваивания

В программировании часто используются несколько странные операторы присваивания, например:

i = i + 1;

Если считать это уравнением, то оно бессмысленно с точки зрения математики. Однако с точки зрения информатики этот оператор служит для увеличения значения переменной ***i*** на единицу.

Буквально это означает: взять старое значение переменной ***i***, прибавить к нему единицу и записать результат в ту же переменную ***i.***

Инкремент и декремент

В языке C/C++ определены специальные операторы быстрого увеличения на единицу

(***инкремента***)

i ++; (постфиксная форма) или ++ i; (префиксная форма)

что равносильно оператору присваивания

i = i + 1;

и быстрого уменьшения на единицу (***декремента***)

`i--;` или `--i;`

что равносильно оператору присваивания

```
i=i-1;
```

Между первой и второй формами этих операторов есть существенная разница, но только тогда, когда они входят в состав более сложных операторов или условий (разберемся позже).

Операторы ввода-вывода

Операторы ввода-вывода данных служат для обмена информацией между внутренней памятью компьютера и внешними устройствами (монитор, клавиатура, принтер, накопители на жестких и гибких магнитных дисках и др.).

Информация, представленная на внешних устройствах, организуется в файлы. Файлы, предназначенные для хранения исходных данных, называются входными. Результаты вычислений (выходные данные) записываются в выходные файлы (они же могут стать потом входными).

Операторы ввода исходных данных с клавиатуры

Работа оператора ввода

Операторы ввода обеспечивают чтение данных с клавиатуры, преобразование их из внешнего представления во внутреннее и присваивание значений переменным, указанным в списке.

Потоковый ввод данных числового типа

Для начинающих программистов проще всего организовать в программе ввод числовой информации из буфера потока, связанного с так называемым стандартным устройством ввода (`stdin`). Данные в этом буфере оказываются в тот момент, когда программа обращается к пользователю и ждет окончания набора затребованных числовых значений с клавиатуры. Программа может запросить одно или несколько значений:

```
  cin >> d;
...........
  cin >> x1 >> x2 >> x3;
```

Первая строка соответствует запросу на ввод значения единственной переменной `d`. Следующая строка программы представляет запрос на ввод трех числовых значений, первое из которых будет присвоено переменной `x1`, второе значение будет предназначаться для переменной `x2`, третье значение - для переменной `x3`. В ответ на запрос программы пользователь должен набрать на клавиатуре затребованные числа, разделяя их, по крайней мере, одним пробелом. Набор данных завершается нажатием клавиши Enter. Количество числовых значений, набираемых пользователем в пределах строки, может оказаться как меньше, так и больше, чем требуется программе. В первом случае продолжение программы задержится до тех пор, пока пользователь не введет дополнительные строки с недостающими значениями. Если пользователь наберет слишком много значений во вводимой строке, то лишние числовые данные сохранятся в буфере ввода и будут переданы программе при выполнении следующего оператора `cin`.

Тип набираемого числового значения и его величина должны соответствовать типу переменной, в которую это значение должно быть введено. Забота об этом целиком возлагается на программиста.

Для обеспечения потокового ввода к программе следует подключить заголовочный файл iostream и применить директиву using namespace.

using namespace std позволяет использовать имена из пространства имен std.

std - это имя стандартной библиотеки C/C++.

```
#include <iostream>
using namespace std;
  int main()
{ int i;
  float f;
  double d;
..........
  cin >> i >> f >> d;
..........
return 0;
}
```

В переменные типа `char` или `unsigned char` из потока числовые значения ввести невозможно, т.к. они в данном случае воспринимаются системой не как числовые данные, а как символьные. В такие переменные попадет только первый символ набранного значения.

Если в последнем примере убрать строку подключения пространства имен using namespace std, то надо будет указывать наименование пространства имен явно:

```
std::cin >> i >> f >> d;
```

Тема «Пространство имен» не входит в круг вопросов, освещаемых в данном пособии. Для самостоятельного изучения данного вопроса рекомендуем воспользоваться источником [3].

Форматный ввод

Гораздо более сложным для начинающих программистов является ввод числовых данных, организуемый с помощью функции `scanf`, использующей так называемую форматную строку:

Общий вид записи оператора ввода

```
scanf(" форматная строка ", список адресов переменных);
```

форматы ввода адреса вводимых переменных

Форматы ввода - это строка в кавычках, в которой перечислены один или несколько форматов (спецификаторов) ввода данных (% - признак спецификатора):

%d - ввод целого числа (для переменной типа int)

%f - ввод вещественного числа (для переменной типа float)

%c - ввод одного символа (для переменной типа char)

После форматов ввода через запятую перечисляются **адреса** ячеек памяти, в которые надо записать введенные значения. Почувствуйте разницу:

a - значение переменной ***a***

&a - адрес переменной ***a***

Например:

```
#include "stdio.h"
  int main()
{ int p;
  float k;
  double m;
..........
  scanf("%d %f %lf",&p,&k,&m);
..........
}
```

Для обеспечения форматного ввода к программе следует подключить заголовочный файл `stdio.h`. Строка вводимых данных поступает со стандартного устройства ввода (`stdin`), которым по умолчанию считается клавиатура. Завершение набора строки ввода - нажатие клавиши Enter.

Первый аргумент функции `scanf` представляет форматную строку, управляющую процессом преобразования числовых данных, набранных пользователем в строке ввода, в машинный формат, соответствующий типам переменных, адреса которых указаны вслед за форматной строкой.

В приведенном примере переменной `p` (в списке ввода указан ее адрес - `&p`), объявленной с помощью спецификатора типа `int`, соответствует спецификатор `%d`. Это означает, что первым числовым значением, набранным на клавиатуре, может быть только целое десятичное число со знаком (`d` - от `decimal`, десятичный). Вещественной переменной `k` типа `float` в форматной строке соответствует формат `%f`. Это означает, что второе числовое значение в строке ввода должно принадлежать диапазону, предусмотренному для коротких вещественных данных. Для переменной `m` типа `double` использован формат `%lf` (`l` (это буква эль)- от `long`).
Количество форматов в строке должно быть равно количеству адресов в списке.

Кроме того, тип переменных должен совпадать с указанным форматом: например, если ***a*** и ***b*** - целые переменные, то следующие вызовы функций ошибочны

```
scanf ( "%d %d", &a );
```

неясно, куда записывать второе введенное число;

```
scanf ( "%d %d", &a, &b, &c );
```

переменная ***c*** не будет введена, так как для нее не задан спецификатор;

```
scanf ( "%f %f", &a, &b );
```

нельзя вводить целые переменные по вещественному формату.

Если значения данных вводятся в одной строке, то они отделяются друг от друга одним или несколькими пробелами.

Пример ввода:

Пусть в задаче определены следующие исходные данные:

$a = 4$; x = 3,8; y =$1{,}2*10^4$; к=5; $m = 74$

Оператор ввода:

```
scanf("%d %f %f %d %d", &a, &x, &y, &k, &m);
```

вызывает чтение данных с клавиатуры, набранных следующим образом:

4 3.8 1.2E4 5 74 [Enter]

%d %f %f %d %d - такой список форматов будет соответствовать следующему описанию переменных в программе:

```
main()
{  int a,k,m;
  float x,y;
```

Операторы вывода данных на экран

Потоковый вывод

Для начинающих программистов организация потокового вывода числовой информации на стандартное устройство вывода экран (`stdout`) представляется более простой:

```
#include <iostream>
using namespace std;

  int main()
{ int i;
  float f;
  double d;
..........
  cout << i+1 << f << f*d;
..........
}
```

Здесь на первых порах не надо заботиться о форматах выводимых результатов. Для каждого типа данных существуют соответствующие системные соглашения. Для вывода пробелов между числами на экране достаточно включить в поток символьные строчки, как это показано в следующем примере:

```
cout << i+1 << "   "<<f << "    "<< f*d;
```

Переход в начало следующей строки вывода на экране осуществляется путем включения в список вывода признака конца строки `endl`, например:

```
cout << i+1 << endl << f << "    "<<f*d;
```

В таком случае значения переменных f и f*d будут выведены на новую строку экрана.

Форматный вывод

Форматный вывод числовых результатов на стандартное устройство вывода (stdout), которым по умолчанию является экран дисплея, осуществляется с помощью функции *printf*.

Общий вид записи оператора

```
printf("список форматов",список имен переменных и выражений);
```

Например:

```
printf ("Результат: %d + %d = %d \n", a, b, c );
```

Содержание скобок при вызове функции printf очень похоже на функцию scanf. Сначала идет символьная (управляющая) строка — форматы вывода, — в которой можно использовать специальные символы:

%d - вывод целого числа
%f - вывод вещественного числа
%c - вывод одного символа
%s - вывод символьной строки
\n - переход в начало новой строки экрана
все остальные символы (кроме некоторых других специальных команд) просто выводятся на экран.

В форматной строке мы сказали компьютеру, *какие* данные (целые, вещественные или символьные) надо вывести на экран, но не сказали *откуда* (из каких ячеек памяти) их брать. Поэтому через запятую после форматов вывода надо поставить список чисел или имен переменных, значения которых надо вывести. В список вывода могут входить арифметические выражения, значения которых предварительно будут подсчитаны, а потом выведены в соответствии с использованными форматными указателями. Выражения и имена переменных в списке вывода перечисляются через запятую. В качестве простейших выражений могут быть константы, переменные, символы, строки.

```
printf ( "Результат: %d + %d = %d \n", a, 5, a+5 );
```

Так же, как и для функции scanf, надо следить за совпадением типов форматов и типов переменных, а также количества форматов вывода и переменных.

Пример1:

```
#include "stdio.h"
  int main()
{ int i;
  float f;
  double d;
..........
  printf("%d %f %lf", i+1, f, f*d);
..........
}
```

Пример 2:

Операторы

```
 printf("Выходные данные:\n");
printf("K=%d  y=%f", 8+5,  7.0/10);
```

выводят на экран результаты в следующем виде:

Выходные данные:

K=13 y=0.7

Управление выводом данных

При стандартной форме вывода вещественные числа отображаются на экране с шестью знаками после запятой. Мантисса выводится в нормализованном виде, т.е. с одной значащей цифрой в целой части и десятью цифрами в дробной части. На порядок отводится четыре позиции: первая позиция - под букву E, вторая позиция - под знак порядка, третья и четвертая позиции - под цифры порядка.

Если программиста не устраивает стандартная форма вывода, он может использовать специальный форматированный вывод, в котором предусмотрены следующие параметры:

1. Ширина поля, определяющая число позиций на экране, отводимых для вывода всего числа, включая целую, дробную части, знак и десятичную точку.
2. Точность представления вещественного числа, определяющая число позиций в дробной части.

Вещественное число с указанием форматов всегда выводится в десятичной форме. Для целого числа используется только формат ширины поля.

В операторах вывода параметры форматов записываются после знака % и отделяются друг от друга точкой. Например, при записи оператора

```
printf("K=%3d    y=%5.2f",8+3,7.0/10);
```

результаты отображаются на экране в следующем виде:

K= 11 y = 0.70

В приведенном примере в форматах `%3d и %5.2f` параметры 3 и 5 задают ширину поля, 2 - точность выводимого числа, т.е. значение выражения 7./10 будет выведено на экран с двумя значащими цифрами после запятой.

Пример линейной программы

Составить программу вычисления площади треугольника по формуле: $S = \sqrt{p(p-a)(p-b)(p-c)}$

где $p = \frac{a+b+c}{2}$ - полупериметр; a, b, c - стороны треугольника.

Исходные данные: a = 1; b = 2; c = 0,5.

```
#include "stdio.h"
#include<math.h>
int  main()
{
   float a, b, c, p, S;/* описание всех переменных, используемых в программе */
 printf("Введите исходные данные  ");
    scanf("%f %f %f",&a, &b,&c);      /* ввод с клавиатуры значений для a, b и c */
 p = (a + b + c) / 2;      // вычисление полупериметра
 S = sqrt(p *(p - a)*(p -b)*(p - c)); // вычисление площади
       /*Вывод на экран*/
 printf("Площадь треугольника S =%5.2f", S);
 return 0;
}
```

При выполнении оператора *scanf* процессор переходит в режим ожидания набора исходных данных с клавиатуры. Для продолжения работы надо набрать на клавиатуре:

1 2 0.5 [Enter]

После нажатия клавиши [Enter] программа продолжит выполнение и на экран будет выведен результат в следующем виде:

Площадь треугольника S = 3.87

Контрольные задания

1. Записать на языке C++ операторы присваивания

Вариант	Математическая запись
1	а) $x = \frac{a - \sqrt{\sin(\varpi t + \varepsilon)} - e^{-\varpi t}}{\sqrt[3]{\ln(2k+d) + d^{3k}}}$ б) $y(x) = \frac{\left(arctgx^3 + \cos\sqrt{x}\right)^{2x}}{e^x + \ln\lvert 2{,}4x^3 \rvert}$
2	а) $m = (1+\sin^2 z)^{\sqrt{\lvert y+6 \rvert}}$ б) $z(x,y) = y\frac{x^{y+1} + e^{y+1}}{1 + x\sin(y+1)^2}$

3	а) $y=\left(\sqrt{\frac{ax+b}{c+dx}}+\sqrt{arctgx}\right)^{2/3}-e^{2x}$ б) $z(x,y)=\left(\frac{ay}{a+b}+\frac{c}{ax^2+bx}\right)^3+\sin yt$
4	а) $y=\pi x^2-a^3n+m\sin(\varpi t+\varepsilon)$ б) $z(x,y)=\frac{a+b}{e^x+\sin x}16e^x\ln x^2$

5	а) $y=\sqrt[3]{\left(\frac{ax}{b+cx}+tgx\right)^2}-e^{x^2}$ б) $z(x,y)=\left(\frac{ax^2}{a+by}+\frac{\ln c}{b+cx^2}\right)^4+\sin^2 y$
6	а) $t=ae^{-\sqrt{\frac{\varpi}{2x}}}$ б) $f(x)=\frac{\sqrt{8+\lvert x-6\rvert^2+b}}{\ln x+2}+e^{bx}(\ln x+2)$
7	а) $c=\sin(1+\sqrt[3]{\ln z}$ б) $f(x)=1-\frac{1+e^{rx}}{\frac{r}{x}(1+x^2)}$
8	а) $t=\frac{1}{2\pi}\sqrt{\frac{m}{c}}$ б) $y(x)=arctgx^2+\frac{\sin\beta x}{2}+e^{\sqrt{2\lvert x\rvert}}$
9	а) $y=\ln\lvert x^7\rvert+\lvert x^3-2\rvert-tgx^3$ б) $z(x)=e^x+a^3\left(\frac{b}{a+b}\right)-2\sin x^2$
10	а) $y=\frac{1}{2}\ln\left\lvert\frac{1+\sin x^2}{1-\cos^2 x}\right\rvert$ б) $z(x)=\ln\lvert\sin ax+\ln x^2\rvert$

11	а) $y = b_1 \sin x + b_2 \sin x^3$ б) $z(y) = ay^5 + b\cos\|y\| + arctg\|y\|^5$
12	а) $y = \frac{\sin^3 x \cdot \cos^2 x}{5} + \frac{2}{5}\sin x^2$ б) $z(x) = x \cdot arctg\frac{x}{a} - \frac{a}{2}\ln\left(a^2 + x^2\right)$
13	а) $y = \sqrt{2\pi} \cdot x^{a+1} \cdot e^{-x}$ б) $f(x) = \frac{\varpi x}{2} a^2 \ln\|x + \varpi\|$
14	а) $b = 1 + \frac{\sin^2\|y\|}{\frac{3}{5} + \cos^2 z}$ б) $f(x) = \frac{1}{4} arctg\frac{\sqrt{3}}{x} + \frac{\sin 3x}{x + \cos^2 x}$
15	а) $y = \frac{3x^2 + 25e^{x^2}}{\|x^7\| + \sqrt{ax^2 + 2}} + \ln\|x\|$ б) $f(x) = \frac{e^a + \ln\|bx\| + 1}{1 + \ln\sqrt{\|dx\|} + e^x}$
16	а) $y = \frac{a_0 + a_1 x + a_2 x^2}{\sqrt[3]{a_2\|\sin x\|}}$ б) $z(x) = \|x^3\| + arctgx + \frac{e^x}{1 + \sin^2 x}$

2. Выполнить оператор присваивания

Вариант	Запись оператора на языке C/C++	Исходные данные	Тип переменных
1	y=3*7 /2 % 7/3;	-	*y* – веществ
2	a=exp(ceil(m+1/2)*log(x+2.5E-2));	x = 8,9 *m= I*	*a, x*- веществ. *m* - целая

3	k=n + round(m+b);	n = -6 b = 0,8 *m = 2*	*к,n,m* - целые *b* - веществ
4	m=a*a*c + floor(b);	a = 45 b = 0,8 c = 2	*a,c* - целые *b,m* - веществ.
5	k=exp(ceil(x*b)*log(y));	x=l,2 b = 2 *y*= 4	*k,x* - веществ. *b,y* - целые
6	d=-a % b + a / b*c;	*a* = 5 b = 2 c = 3	*a,b,c* - целые *d* -веществ.
7	i=floor(k/n*n + a);	*к* = 99 *n* = 100 *a* = 5,87	*i,k,n* - целые *a* - веществ.
8	f=a + ceil(b/c)-64/pow(a,2)+1;	a = 4 b = 0,8 c = 2	*f,b* - веществ. *a,c* - целые
9	b=4*d*1E-2/cos(4*d-c)+floor(57.12);	c=12 d=3	*c,d*- целые b - веществ.
10	k=a*b/c + floor(12.7E-1);	*a* = 6 b = 2 c = 4	a,b,c – целые *к* - веществ.
11	b=exp(ceil(n/m)*log(4+sqrt(4/m)));	m = 3 n = 2	*n,m* - целые b- веществ.
12	m=floor(a*(b / 4)+0.57E1);	a = 6 b=1	*m,a,b* - целые
13	a=exp(5 /2*log(4))-2;	-	*a* - веществ.
14	z=(4+sqrt(b+1)+24E-1 *c)*floor(b/4);	b = 1 c=10	*Ь,c* – целые z - веществ.
15	p=ceil(abs(b-a)) / 2-34 % 10;	a = 4,8 b = 0,8	*p,a,b* -*в*еществ.
16	a=19 /4*4+floor(5.4) % 4;	-	*a* - целая