

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

С.С. Хохлова, М.А. Бугенко, Н.М. Кузьмин

Программирование на языках высокого уровня

Учебно-методическое пособие

Волгоград 2016

Рекомендовано к опубликованию Ученым советом
института математики и информационных технологий
Волгоградского государственного университета
(протокол №10 от 21.11.2016)

Рецензент

доктор. физ.-мат. наук, профессор кафедры Теоретической
физики и волновых процессов
Волгоградского государственного университета *В.А. Михайлова*

С. С. Хохлова, М. А. Бутенко, Н. М. Кузьмин

Программирование на языках высокого уровня [Текст] : учеб.-метод. пособие / С.С. Хохлова, М.А. Бутенко, Н.М. Кузьмин; Фед. гос. авт. образоват. учреждение выс. образования «Волгогр. гос. ун-т». – Волгоград: Изд. ВолГУ, 2016. – 52 с. Ил.

Настоящее пособие содержит первую часть лабораторного практикума по курсу «Языки высокого уровня» для студентов первого курса, обучающихся на направлениях подготовки бакалавриата 09.03.01 Информатика и вычислительная техника, 09.03.02 Информационные системы и технологии и 09.03.04 Программная инженерия в первом семестре. В качестве вспомогательного материала приведенная в пособии информация может быть использована студентами магистратуры 09.04.01 Информатика и вычислительная техника, 09.04.04 Программная инженерия. Учебно-методическое пособие ориентировано на изучение базовых конструкций языка Си и приобретение навыков алгоритмизации расчетных задач. Курс ориентирован на изучение структурного программирования и может рассматриваться как начальный этап обучения программированию. Пособие предполагает знание студентом основ информатики в объеме школьной программы и опыт работы с персональным компьютером под управлением операционных систем Windows и UNIX-подобных.

Введение

В данном учебно-методическом пособии, предназначенном для выполнения студентами первой части лабораторного практикума по программированию на языках высокого уровня, нашел отражение многолетний опыт авторского коллектива в области проведения лекционных занятий и лабораторного практикума по дисциплинам, связанным с программированием.

Текст содержит последовательное описание теоретических сведений и методических указаний для выполнения первых шести лабораторных работ по дисциплине «Языки высокого уровня». В двух приложениях представлены варианты индивидуальных заданий и сведения о математических функциях, наиболее часто используемых при написании программ.

Следует отметить, что изложенные здесь сведения могут пригодиться студентам не только при изучении непосредственно дисциплины «Языки высокого уровня», но и для последующего изучения дисциплин, связанных с проектированием и созданием программного обеспечения, а также применением современных технологий программирования (в том числе и объектно-ориентированного подхода): «Объектно-ориентированное программирование», «Визуальное программирование», «Технологии разработки программного обеспечения», «Архитектура информационных систем».

Отметим особо, что сведения, изложенные здесь, скорее дополняют (а отнюдь не заменяют!) соответствующий курс лекций. Кроме этого, добавим для студента, что круг вопросов, связанных с программированием, поистине безграничен. И книги, приведенные в списке литературы, конечно же, не содержат все ответы. Скорее, они являются лишь отправной точкой для увлекательного путешествия в мир программирования.

Лабораторная работа №1

Введение в программирование на языках высокого уровня

Цель работы: изучение основных этапов создания программного обеспечения и знакомство с каскадной моделью жизненного цикла программного обеспечения.

Этапы разработки программ

Процесс создания программы для электронно-вычислительной машины (ЭВМ) от составления алгоритма до ввода готового программного продукта в эксплуатацию называется **жизненным циклом программного обеспечения**¹ (ЖЦ ПО). Существует довольно много моделей ЖЦ ПО. Рассмотрим одну из самых распространенных - **каскадную модель**² (рисунок 1).

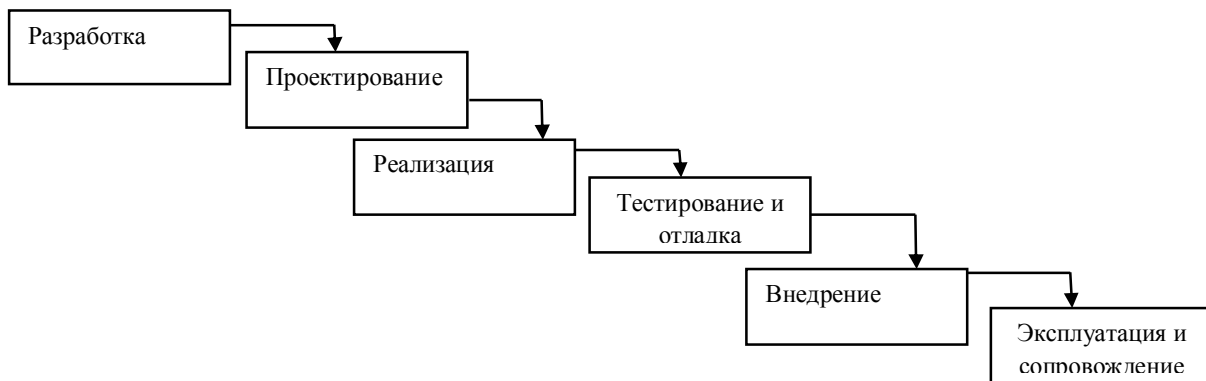


Рисунок 1 – Каскадная модель ЖЦ ПО

Модель включает в себя нескольких этапов, которые мы далее рассмотрим. На каждом этапе формируется законченный набор проектной документации, отвечающий критериям полноты и согласованности. Выполняемые в логичной последовательности этапы работ позволяют планировать сроки завершения всех работ и соответствующие затраты.

1 этап - разработка требований. На этом этапе определяются входные и выходные данные, области их допустимых значений, ограничения ПО, реакция ПО на возможные ошибки пользователя. Результатом этого этапа является документ, называемый *спецификацией*. Спецификация содержит постановку задачи, анализ этой задачи и подробное описание действий, которые должна выполнять программа. Спецификация содержит описание входных, выходных и промежуточных данных; отвечает на вопросы: какие входные данные яв-

¹ Среди наиболее известных стандартов можно выделить следующие: ГОСТ 34.601-90, ISO/IEC 12207:1995.

² Каскадная модель (водопад) ЖЦ ПО получила наибольшее распространение в 70-х-80-х годах прошлого века. Данная модель хорошо зарекомендовала себя при построении простых программных продуктов, однако она обладает некоторыми недостатками. Помимо каскадной модели достаточно известными являются модели «водоворот» и спиральная.

ляются корректными, а какие ошибочными, кто является пользователем программы и каким должен быть интерфейс пользователя, какие ошибки должны выявляться и какие сообщения об ошибках должны выдаваться пользователю, какие ограничения имеет программа, а также содержать описание всех особых ситуаций, требующих специального рассмотрения.

Рассмотрим создание спецификации на примере программы для нахождения корня квадратного уравнения. Спецификацию составим в свободной форме так, чтобы она позволяла *однозначно понимать*, какая программа должна быть создана.

Спецификация:	<p>Программа предназначена для решения квадратного уравнения вида:</p> $ax^2 + bx + c = 0.$ <p><i>Входные данные:</i> вещественные коэффициенты a, b, c. <i>Выходные данные:</i> вещественные корни x_1, x_2.</p> <p><i>Особые ситуации:</i></p> <ol style="list-style-type: none"> 1. если дискриминант меньше нуля, то действительных корней нет. В этом случае программа должна сообщить «нет действительных корней». 2. если дискриминант равен 0, то корни кратные (совпадают). Программа должна вывести одно и то же значение корня для x_1 и x_2. <p><i>Интерфейс пользователя</i> текстовый: ввод данных с клавиатуры, вывод данных на экран в виде строк.</p>
---------------	---

С этой помощью спецификации заданы достаточно жесткие рамки, которые не позволяют трактовать задачу произвольным образом. Следовательно, цель, для которой разрабатывалась спецификация, достигнута.

2 этап – проектирование. На данном этапе происходит выбор (или разработка) алгоритма для решения задачи, разработка интерфейса программы.

3 этап – реализация (кодирование). На этапе реализации происходит запись алгоритма на языке программирования в виде последовательности операторов.

4 этап – тестирование. Данный этап связан с проверкой адекватности выбранного алгоритма и правильности работы программы. На данном этапе происходит сравнение результатов решения задачи с помощью разработанной программы с результатами, полученными другими способами (вручную, либо с помощью специального ПО). Если при тестировании установлена ошибочность работы разрабатываемой программы, то осуществляется *отладка* - поиск ошибок в программе (алгоритме) и их устранение.

5 этап – внедрение. На этапе внедрения происходит внедрение ПО в эксплуатацию и обучение пользователей работе с ним.

6 этап – эксплуатация и сопровождение. На данном этапе происходит устранение дополнительных недостатков в работе программы, которые не были выявлены на этапе тестирования.

тирования и обнаруживаются в процессе работы пользователей с ПО. Также в процессе эксплуатации у пользователей могут появиться новые пожелания по расширению функциональных возможностей и адаптации ПО к конкретным условиям его эксплуатации.

Контрольные вопросы

1. Перечислите и охарактеризуйте этапы каскадной модели ЖЦ ПО.
2. Какими достоинствами и недостатками обладает каскадная модель ЖЦ ПО?
3. Какие еще модели ЖЦ ПО существуют? Приведите примеры.
4. Охарактеризуйте спиральную модель ЖЦ ПО. Какими достоинствами и недостатками она обладает по сравнению с каскадной моделью?
5. Охарактеризуйте модель ЖЦ ПО «водоворот». Какими достоинствами и недостатками она обладает по сравнению с каскадной моделью?
6. В каких случаях следует применять такие модели ЖЦ ПО как экстремальное программирование, SCRUM и инкрементальная модель?
7. Что такое спецификация программы? Для чего нужна спецификация?
8. Что происходит на этапе тестирования ПО?
9. В чем принципиальная разница между отладкой и тестированием?

Порядок выполнения и сдачи работы³

1. Составьте конспект теоретической части лабораторной работы, при необходимости изучив материалы из альтернативных источников информации.
2. Приготовьте ответы на контрольные вопросы.
3. Получите индивидуальное задание у преподавателя.
4. Оформите выполнение индивидуального задания в тетради.
5. Представьте устный отчет о выполнении работы преподавателю.

Отчет

Фамилия	Лабораторная работа №	Отчет
Имя	Название лабораторной работы	
Отчество		
Группа		

Конспект лабораторной работы должен содержать одну-две страницы теоретического описания, индивидуальное задание, спецификацию, блок-схему, исходный код программы. Помимо этого отчет должен включать результаты тестирования программы для всех возможных тестовых значений и общие выводы по выполнению лабораторной работы.

³ Порядок выполнения и сдачи всех последующих работ такой же.

Лабораторная работа №2

Алгоритмы. Графическое представление алгоритмов

Цель работы: изучение теоретических сведений о графическом представлении алгоритмов и приобретение практических навыков по созданию блок-схем.

Алгоритм

Понятие алгоритма является фундаментальным в математике и компьютерных науках. Существует множество определений алгоритма. В самом общем смысле алгоритм – это понятная и точная последовательность действий, описывающая процесс перехода объекта из начального состояния в конечное. В рамках настоящего курса, под *алгоритмом* мы будем понимать набор инструкций, состоящий из конечного числа шагов, выполнение которых приводит к решению конкретной задачи.

Основные свойства алгоритма:

1. *Дискретность*. Алгоритм представляет собой процесс решения задачи в виде последовательности простых шагов, для выполнения каждого из которых требуется конечное время. Получение результата из входных данных осуществляется пошагово, дискретно.
2. *Детерминированность (определенность)*. Результат, полученный на текущем шаге, полностью определяется результатом предыдущего шага. Алгоритм всегда выдает один и тот же результат для одних и тех же исходных данных.
3. *Понятность*. В алгоритм входят только те действия, которые понятны конкретному исполнителю и могут быть им выполнены.
4. *Конечность*. Для корректных входных данных алгоритм должен выдать верный результат за конечное число шагов. При вводе некорректных входных данных алгоритм должен после конечного числа шагов выдать сообщение о том, что задачу с его помощью решить невозможно.
5. *Массовость*. Алгоритм должен быть таким, чтобы его можно было применять к различным входным данным. Например, алгоритм нахождения корней квадратного уравнения, приведенный в лабораторной работе 1, не зависит от выбора коэффициентов.

Существует несколько способов представления алгоритмов:

- словесное описание (список действий);
- псевдокод⁴;
- код программы, записанный на одном из языков программирования;

⁴ Псевдокод – компактный и неформальный язык описания алгоритмов, использующий ключевые слова языков программирования, который опускает подробности, несущественные для понимания алгоритма, и специфический синтаксис реализации конкретных языков программирования.

- блок-схема.

Таким образом, *программа* – это один из способов записи алгоритма, с использованием синтаксиса и правил какого-либо языка программирования.

Элементы блок-схемы

Представление алгоритмов в виде блок-схем отличается большой наглядностью. Кроме того, данный подход является достаточно простым в применении. Дадим основные определения элементов блок-схемы согласно действующему стандарту ГОСТ 19.701-90⁵.





Схемы алгоритмов, программ, данных и систем (далее – *схемы*) состоят из имеющих заданное значение символов, краткого пояснительного текста и соединяющих линий.

Схемы программ отображают последовательность операций в программе. Схема программы состоит из следующих частей:


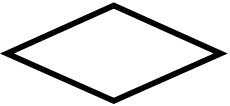

1. символов процесса, указывающих фактические операции обработки данных (включая символы, определяющие путь, которого следует придерживаться с учетом логических условий);
2. линейных символов, указывающих поток управления;
3. специальных символов, используемых для облегчения написания и чтения схемы.

В таблице 1 приведены основные символы блок-схем с пояснениями к ним.

Таблица 1 – Основные элементы блок-схемы

Элемент блок-схемы	Название	Пояснение
	Данные (блок ввода-вывода)	Символ отображает данные, носитель данных не определен. Служит для изображения блока ввода-вывода в общем виде.
	Документ	Символ отображает данные, представленные на носителе в удобочитаемой форме. Обычно служит для изображения вывода результатов на консоль.
	Процесс	Символ отображает функцию обработки данных любого вида. Служит для изображения вычислительных действий или их последовательности.
	Предопределенный процесс	Символ отображает предопределенный процесс, состоящий из одной или нескольких операций или шагов программы, которые определены в другом месте. Служит для изображения вычислений, производимых в подпрограмме.

⁵ Соответствует международному стандарту ISO 5807-85.

	Модификация	Символ отображает модификацию команды или группы команд с целью воздействия на некоторую последующую функцию. Служит для изображения начала цикла со счетчиком.
	Решение (ветвление)	Символ отображает решение или функцию переключательного типа, имеющую один вход и ряд альтернативных выходов, один и только один из которых может быть активизирован после вычисления условий, определенных внутри этого символа. Служит для изображения ветвлений.
	Терминатор	Символ отображает начало или конец схемы программы, внешнее использование и источник или пункт назначения данных. Служит для изображения начала и конца блок-схемы.

Базовые алгоритмические структуры

Можно выделить три *базовые алгоритмические структуры*, с помощью которых можно описать алгоритм:

1. следование (последовательное выполнение);
2. ветвление;
3. цикл.

Следование представляет собой простое последовательное выполнение действий одного за другим (рисунок 2).

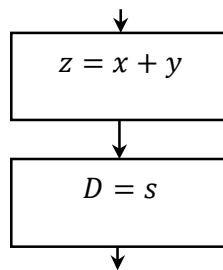


Рисунок 2 – Следование

Ветвление используется в тех случаях, когда выполнение того или иного действия (или их набора) зависит от выполнения некоторого условия. Можно выделить две основные конструкции ветвления:

1. *Неполное ветвление* или ЕСЛИ – ТО (рисунок 3а);
2. *Полное ветвление* или ЕСЛИ – ТО – ИНАЧЕ (рисунок 3б).

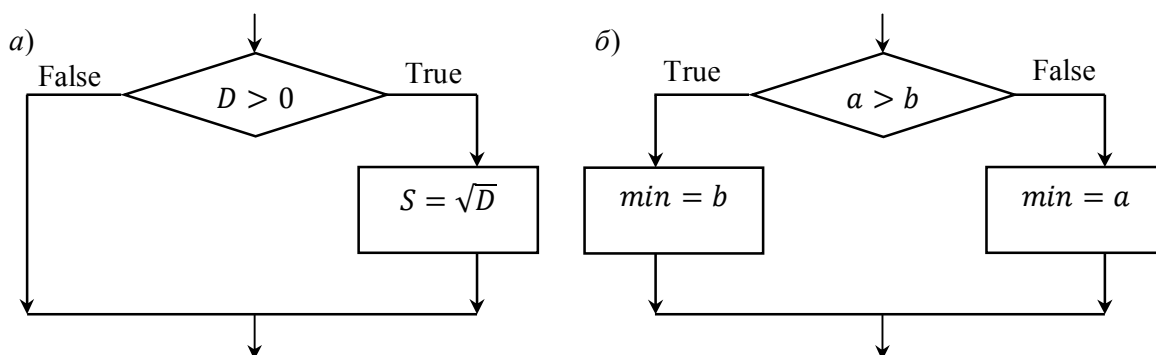


Рисунок 3 – а) неполное ветвление, б) полное ветвление

Цикл предназначен для обозначения выполнения повторяющегося действия или их набора. Разделяют заголовок («шапку», условие) цикла и тело цикла. В заголовке содержится условие, пока оно истинно, выполняется *тело* цикла - одно или несколько действий. Можно выделить два основных вида циклов:

1. Цикл «пока» – он выполняется до тех пор, пока выполняется некоторое условие.
 - а) Цикл с *предусловием*: сначала проверяется истинность условия, а потом выполняется тело цикла (рисунок 4а).
 - б) Цикл с *постусловием*: сначала выполняется тело цикла, а потом проверяется условие. Таким образом, тело цикла с постусловием гарантированно выполнится хотя бы один раз, вне зависимости от истинности условия (рисунок 4б).
2. Цикл «для каждого» (цикл со счетчиком): выполняется для каждого элемента из заданного набора или заданное изначально число раз (рисунок 4в).

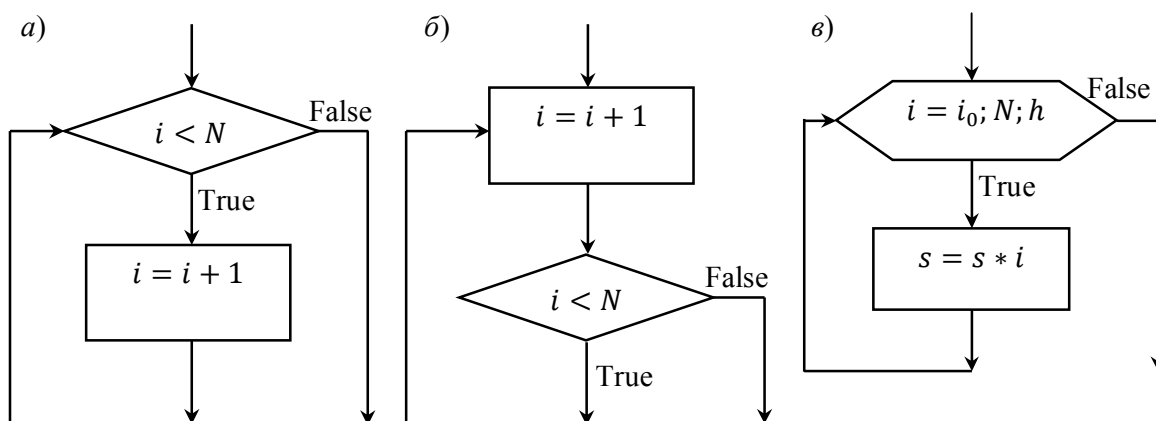


Рисунок 4 – а) цикл «пока» с предусловием, б) цикл «пока» с постусловием, в) цикл со счетчиком

Поясним работу цикла со счетчиком, представленного на рисунке 4в. В шестиугольнике записывается заголовок цикла. В нем указываются имя переменной-счетчика, а далее,

после символа « \Leftarrow », через точку с запятой, записываются ее начальное значение (i_0), ее конечное значение (N) и шаг ее изменения (h). На первом шаге цикла значение счетчика: $i = i_0$. На следующих шагах к текущему значению счетчика i прибавляется шаг h : $i = i + h$. Цикл заканчивается, как только переменная цикла достигает конечного значения N : $i = N$.

Приведем пример полной блок-схемы для решения задачи нахождения корней квадратного уравнения $ax^2 + bx + c = 0$ (рисунок 5).

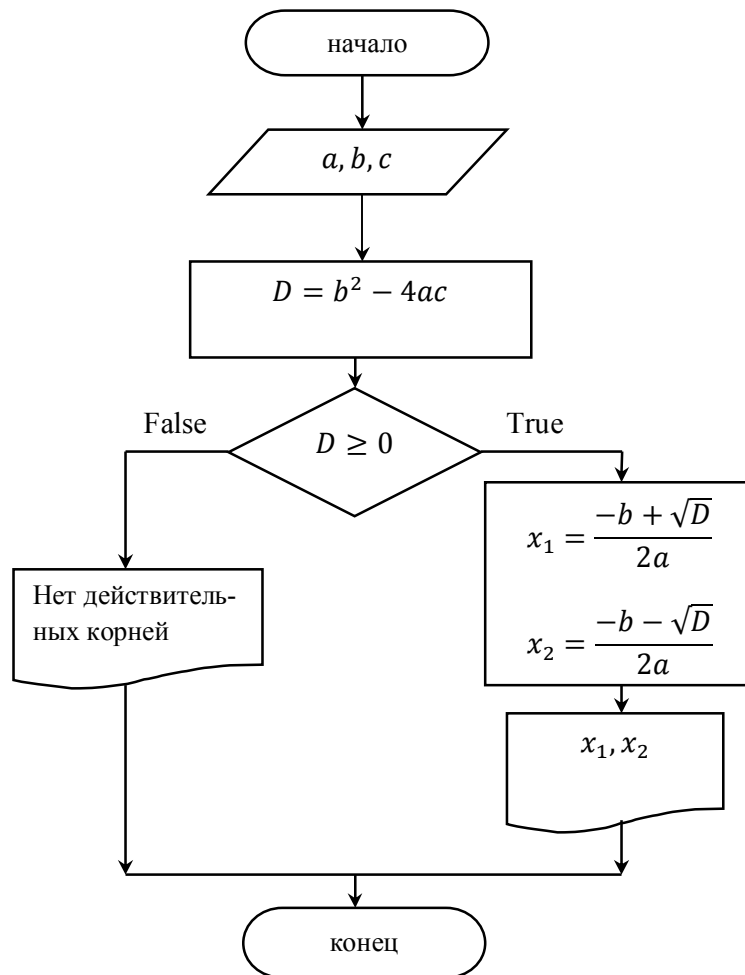


Рисунок 5 – Нахождение корней квадратного уравнения $ax^2 + bx + c = 0$

Контрольные вопросы

1. В каком виде можно представить алгоритм?
2. Какие элементы блок-схем вы знаете? Объясните назначение этих элементов.
3. Назовите три базовые алгоритмические структуры. Приведите примеры.
4. В каких случаях используется алгоритмическая структура следование?
5. В каких случаях используется алгоритмическая ветвление? Приведите примеры.
6. Какие виды ветвлений существуют? Приведите примеры.

7. В каких случаях используется алгоритмическая цикл? Приведите примеры.
8. Какие виды циклов существуют?
9. Как будет выглядеть блок-схема из рисунка 5, если нахождение корней уравнения реализовано через подпрограмму?
10. Что такое псевдокод? Чем отличается псевдокод от программы, написанной на языке программирования?
11. Составьте словесное описание алгоритма для блок-схемы из рисунка 5.
12. В чем заключается преимущество использования блок-схемы перед словесным описанием алгоритма?

Лабораторная работа №3

Знакомство с языком Си. Компиляция программ

Цель работы: изучение основных терминов и определений в области создания исходного кода на языке Си и компиляции программ.

Программа

Программа - это набор команд, последовательно выполняя которые, ЭВМ решает некоторую задачу: нахождение корней уравнений, анализ текста, предсказание погоды и т. п. Итак, *ЭВМ является исполнителем строго определенного набора команд*. Это значит, что она может выполнить лишь те команды, которые ей понятны и которые она умеет выполнять. *Все остальные команды для ЭВМ не имеют никакого смысла.*

В настоящее время для написания программ для ЭВМ обычно используются высокоуровневые языки программирования. Программы, написанные на них, непонятны непосредственно для ЭВМ, но работа с ними гораздо проще работы с низкоуровневыми инструкциями конкретной ЭВМ. После написания программы на языке высокого уровня для ее выполнения необходимо ее «перевести» на понятный конкретной ЭВМ машинный язык (язык машинных кодов). Об этом подробнее будет рассказано далее.

Одним из самых простых интерфейсов взаимодействия ЭВМ и человека является текстовый: пользователь набирает команды на клавиатуре, а ЭВМ выдает текстовые сообщения на экран. Поэтому первая программа, которую мы напишем - это программа для вывода на экран строки «Hello, world!». На языке Си исходный код данной программы имеет вид:

```
/*  
    My first C-program  
*/  
#include <stdio.h>  
int main()  
{  
    printf ("Hello, world!\n");  
    return 0;  
}
```

Этот текст можно набрать в любом текстовом редакторе и сохранить в файле с расширением «.c», например, hello.c. Теперь некоторые пояснения к самой программе.

Комментарии

Для улучшения читаемости программы код программы может содержать комментарии. Комментарии – это строки программы, которые поясняют код и при этом компилятор их игнорирует. Комментарии могут появляться в любом месте программы. В языке Си существует два типа комментариев: строчные и блочные. Строчные комментарии появляются после символов `//`, а блочные комментарии отделяются от текста программы символами `/* */`. В тексте программы комментарии будут выглядеть следующим образом:

```
/*  
    Block comment  
*/  
int main() // Line comment
```

Заголовочные файлы

В программировании заголовочный файл или подключаемый файл – это файл, содержимое которого автоматически добавляется препроцессором в исходный текст в том месте, где располагается некоторая директива. Заголовочные файлы (библиотеки или пользовательские файлы) могут содержать типы данных, структуры, прототипы функций, перечислимые типы и макросы, хранящиеся в других файлах. Строка

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

позволяет использовать в теле основной программы функцию вывода строки на экран `printf`.

Функция main

Каждая Си-программа должна содержать функцию с именем `main()`. Функция `main()` является основной исполняемой функцией программы, написанной на языке Си. Для выполнения определенных действий функция `main` обычно обращается к другим функциям, часть из которых находится в той же самой программе, а часть – в библиотеках, содержащих ранее написанные функции. Одним из способов обмена данными между функциями является передача аргументов. Круглые скобки, следующие за именем функции, заключают в себе список аргументов. В нашем случае `main` - функция без аргументов, что указывается как `()`. Операторы, составляющие функцию, заключаются в фигурные скобки `{ }`. Фигурные скобки в языке Си представляют собой составной оператор, позволяющий группировать операторы языка вместе. Функция `main` возвращает значение типа `int`! Если вы присвоите функции `main` тип отличный от `int`, то в компиляторах предшествующих стандарту C99 вы получите неопределенное поведение своей программы, а в компиляторе с поддержкой стандарта C99 вы получите неспецифицированное поведение. Возвращение значения из функции происходит с по-

мощью ключевого слова `return`. Тип возвращаемого значения должен полностью совпадать с задекларированным типом функции.

Вызов других функций

Обращение к функции осуществляется указанием ее имени, за которым следует заключенный в круглые скобки список аргументов. Круглые скобки должны присутствовать и в том случае, когда функция вообще не имеет аргументов. Функция `printf` является библиотечной⁶ функцией, которая выводит строку "Hello, world!\n" на экран. Строка

```
printf("Hello, world!\n");
```

является обращением к функции `printf`, аргументом которой является строка "Hello, world!\n". Последовательность из любого количества символов, заключенных в двойные кавычки "...", называется «символьной строкой» или «строчной константой».

Символы «\n» называются эскейп-последовательностью и служат указанием для перехода на экране к новой строке. Если вы не включите \n, то обнаружите, что ваша выдача не закончится переходом курсора на новую строку. Использование последовательности функции \n - единственный способ введения символа новой строки в аргумент `printf`. В таблице 2 представлены некоторые из наиболее употребительных эскейп-последовательностей.

Таблица 2 – Наиболее употребительные эскейп-последовательности

<i>Последовательность</i>	<i>Описание</i>
\n	Перевод каретки на новую строку.
\t	Табуляция (смещение каретки по горизонтали на 8 знакомест ⁷).
\b	Возвращение на одну позицию назад (если в этой позиции имеется символ – он стирается).
\"	Вывод двойных кавычек на экран.
\v	Вертикальная табуляция.
\\	Вывод на экран обратной косой черты.

Многократный вызов функции `printf` можно использовать для поэтапной сборки выходной строки. Наша первая программа с точно таким же успехом могла бы быть записана следующим образом:

```
int main()
{
    printf("Hello, ");
```

⁶ Прототип функции `printf` находится в заголовочном файле стандартной библиотеки ввода-вывода `stdio.h`.

⁷ Согласно кодировке ASCII.

```

printf("world!");
printf("\n");
return 0;
}

```

Оформление программы в целом

В настоящее время существует несколько наиболее распространенных стилей расстановки фигурных скобок и оформления программы в целом. Правильное оформление программы делает код легко читаемым и удобным для редактирования и модификации. С подробным описанием существующих стилей вы можете ознакомиться в справочнике [8]. Мы настоятельно рекомендуем на начальном этапе знакомства с программированием использовать при оформлении кода программ стиль Алмена (Eric Allman). Этот стиль предполагает использование фигурных скобок друг под другом, и отступ в виде табуляции для всех операторов размещающихся внутри фигурных скобок. Именно этот стиль использован для оформления примеров в настоящем пособии.

Компиляция

Как мы уже отмечали выше, ЭВМ не понимает исходный код высокоуровневых языков. Для перевода исходного текста программы, написанной на высокоуровневом языке программирования в машинные коды⁸ используют специальную программу – компилятор. Здесь и далее слово «программа» будем понимать в двух смыслах:

1. *Исходный текст программы* - последовательность операторов, записанная на языке программирования, которую необходимо выполнить для решения какой-либо задачи.
2. *Машинный код* - понятная конкретной ЭВМ последовательность команд.

Компиляция программы на языке Си происходит в несколько этапов: препроцессирование, трансляция и компоновка. При запуске программы-компилятора сначала исходная программа передается *препроцессору*⁹, который выполняет *директивы*, содержащиеся в ее тексте. Препроцессор выполняет также некоторую начальную подготовку исходного текста, соединяя длинные строки разделенные символами «\» и отбрасывая строки комментариев. Получившийся полный текст программы поступает на вход *транслятора*, который выделяет лексемы¹⁰, а затем на основе грамматики языка распознает выражения и операторы, построенные из этих лексем. При этом транслятор выявляет синтаксические ошибки и в случае их отсутствия создает (зачастую с промежуточной фазой генерации ассемблерного кода и обра-

⁸ ЭВМ понимает и может непосредственно исполнять только машинные коды.

⁹ В примере такой директивой является `#include <stdio.h>`, т.е. подключение заголовочного файла стандартной библиотеки ввода-вывода.

¹⁰ Последовательность допустимых символов языка программирования.

ботки его ассемблером) *объектный модуль (объектный код)*. *Объектный код* – это результат перевода (трансляции) исходного текста программы на язык машинных кодов. Затем *компоновщик (линковщик)*, или редактор связей, формирует *исполняемый модуль* программы, подключая к полученному ранее объектному модулю другие объектные модули, в том числе содержащие функции библиотек, обращение к которым содержится почти в любой программе (например, для осуществления вывода на экран). Если программа состоит из нескольких исходных файлов, они компилируются по отдельности и объединяются на этапе линковки.

Для компиляции программы необходимо в командной строке выполнить команду:
`cc hello.c`

После компиляции в текущем каталоге будет создан исполняемый файл с именем `a.out`¹¹. Запуск его по команде `./a.out` приведет к выводу на экран строки «Hello, world!».

Контрольные вопросы

1. В каких смыслах можно понимать слово «программа»?
2. Перечислите и опишите основные компоненты Си-программы. Приведите пример.
3. Дайте определение понятию «компилятор».
4. Из каких стадий состоит процесс компиляции Си-программы? Охарактеризуйте эти стадии.
5. Какие действия выполняет линковщик?
6. Как можно откомпилировать программу? Продемонстрируйте на примере.
7. Что такое символьная строка? Как вывести строку на экран?
8. Для чего используются эскейп-последовательности?
9. Что такое комментарии? Для чего нужно использовать в программе комментарии?
10. Какой стандарт языка Си является действующим на данный момент?
11. Может ли функция `main` иметь тип `void` (не иметь тип)? Обоснуйте ответ.

¹¹ В различных операционных системах и средах разработки компиляция и запуск программы могут осуществляться различными способами. Здесь мы рассмотрели пример для UNIX-подобной операционной системы.

Лабораторная работа №4

Переменные. Типы данных. Основные операции

Цель работы: изучение основных типов данных языка Си и операций над ними.

Типы данных

В таблице 3 представлены основные типы данных, используемые языком Си, а также характерный размер памяти, отводимый под хранение переменных разного типа на 64-разрядной архитектуре x86-64. Обратите внимание, что стандарт языка Си не регламентирует, сколько байт отводится в памяти для переменной того или иного типа. Поэтому на разных системах (32-разрядных, 64-разрядных) объем памяти, занимаемый, например, переменной типа `long int`, может быть различным (исключением является только тип данных `char`: переменная этого типа всегда занимает в памяти 1 байт).

Таблица 3 – Типы данных

Тип данных	Описание	Размер, занимаемый в памяти
<code>short int</code>	Короткое целое	2 байта
<code>int</code>	Целое со знаком	4 байта
<code>long int</code>	Длинное целое	8 байт
<code>unsigned int</code>	Целые без знака	4 байта
<code>float</code>	Вещественное число с плавающей точкой одинарной точности	4 байта
<code>double</code>	Вещественное число с плавающей точкой двойной точности	8 байт
<code>long double</code>	Длинное вещественное число с плавающей точкой двойной точности	16 байт
<code>char</code>	Символьная переменная	1 байт

Переменные целых типов могут быть знаковыми (поддерживается хранение отрицательных значений) и беззнаковыми (хранение отрицательных значений не поддерживается). Для указания того, к какому виду относится переменная, используют ключевые слова `signed` и `unsigned`.

Оператор `sizeof`

Этот оператор возвращает количество байт, которое на данной системе выделяется переменной того или иного типа данных. Его синтаксис:

```
z = sizeof(short);
```

где переменная `z` имеет целый тип.

Объявление переменных

Имя переменной – это последовательность латинских букв, цифр и символов подчеркивания «_». Регистр букв является значимым, поэтому переменные `arg` и `Arg` считаются разными. Первым символом имени переменной не может быть цифра. Следует придумывать имена, отражающие смысл применяемых переменных. Правильный подбор имен и объявление переменных в начале программы улучшает читаемость кода и делает структуру программы более прозрачной. Хорошим тоном при написании программ является объявление всех переменных в начале программы.

В общем случае объявление переменных можно представить следующим образом¹²:

```
int A, B;  
float c = 0.0F;  
double x, y = 1.0;  
char z = 'x';
```

Таким образом, для объявления переменной необходимо указать тип переменной и имя переменной, на этом этапе также можно инициализировать значение переменной некоторой константой.

Константы

При использовании целочисленных констант необходимо просто указать цифрами нужное число, например, `123456`. Если необходимо задать отрицательное значение перед числом ставим знак «минус»: `-123456`. Если необходимо уточнить, что константа имеет тип `long int`, то после цифровой записи числа добавляем суффикс `L` или `l`, например: `123456L`.

Вещественные константы имеют две формы записи. Первая форма записи использует символ точки для разделения целой и дробной части числа: `123.456`. По умолчанию считается, что вещественные константы имеют тип `double`. Если необходимо явно указать, что вещественная константа имеет тип `float`, то после числовой записи числа добавляется суффикс `F` или `f`: `123.456F`. Вторая форма записи числа называется экспоненциальной (научной). Она особенно удобна при работе с очень маленькими или очень большими числами. Например, число $1,234 \cdot 10^{-5}$ может быть записано в экспоненциальной форме следующим образом: `1.234E-5`. Символ `E` (или `e`) заменяет собой `10`. Обе формы записи эквивалентны.

Символьные константы записываются в виде символа, заключенного в одинарные кавычки, например: `'a'`.

¹² Следует помнить, что при объявлении переменных без инициализации: `int A, B;` компилятор просто выделяет в оперативной памяти место для хранения переменных, не обращая внимания на содержимое данной области памяти.

Математические функции

Для использования в программе математических функций необходимо подключить заголовочный файл математической библиотеки `math.h`¹³. Список математических функций представлен в Приложении Б.

Практически все математические функции работают с аргументами типа `double` и рассчитывают значения типа `double`, за исключением функции `abs`. Если в расчетах необходимо использовать числа с плавающей точкой типа `float`, то необходимо к используемой функции добавить букву “f”. Например, функция `sin` для чисел `x` и `y` типа `float` будет иметь вид `y=sinf(x)`.

При использовании функции `pow(x,y)` следует понимать, что данная функция при расчете использует разложение в ряд. При этом погрешность вычисления (отличие от истинного значения) может быть достаточно большой. В тех случаях, где происходит возведение в небольшую целую степень, использования данной функции следует избегать!

Основные операции

В таблице 4 приведены основные операции языка Си.

Таблица 4 – Основные операции языка Си

Оператор	Запись операции	Описание
<i>Арифметические операции</i>		
=	$x = y$	Бинарный арифметический оператор присваивания.
+	$x + y$	Бинарный арифметический оператор сложения.
–	$x - y$	Бинарный арифметический оператор вычитания.
/	x / y	Бинарный арифметический оператор деления.
*	$x * y$	Бинарный арифметический оператор умножения.
%	$x \% y$	Бинарный арифметический оператор получения остатка от деления. Возвращает 0, если x делится нацело на y , и остаток от деления в противном случае. (Данный оператор не может быть применен к переменным типа <code>float</code> или <code>double</code>).
<i>Инкремент и декремент</i>		
++	$++x$ $x++$	Унарный оператор увеличения значения переменной x на 1 в префиксной и постфиксной форме записи.
--	$--x$ $x--$	Унарный оператор уменьшения значения переменной x на 1 в префиксной и постфиксной форме записи.

¹³Для корректной компиляции файла, использующего функции из математической библиотеки, в командной строке необходимо набрать следующую строку `cc -lm program.c`. Флаг `-lm` подсказывает линковщику, что необходимо подключить математическую библиотеку.

<i>Побитовые операции</i>		
&	$x \& y$	Бинарный оператор, проводящий побитовую операцию И (AND) над числами x и y .
	$x y$	Бинарный оператор, проводящий побитовую операцию ИЛИ (OR) над числами x и y .
^	$x \wedge y$	Бинарный оператор, проводящий побитовую операцию ИСКЛЮЧАЮЩЕГО ИЛИ (XOR) над числами x и y .
<<	$x \ll y$	Бинарный оператор побитового сдвига на y бит влево числа x .
>>	$x \gg y$	Бинарный оператор побитового сдвига на y бит вправо числа x .
~	$\sim x$	Унарный оператор побитового НЕ.
<i>Составные операторы</i>		
+=	$x += y$	Данная запись эквивалентна $x = x + y$
--	$x -= y$	Данная запись эквивалентна $x = x - y$
*=	$x *= y$	Данная запись эквивалентна $x = x * y$
/=	$x /= y$	Данная запись эквивалентна $x = x / y$
% =	$x \% = y$	Данная запись эквивалентна $x = x \% y$
^ =	$x \wedge = y$	Данная запись эквивалентна $x = x \wedge y$
<<=	$x \ll = y$	Данная запись эквивалентна $x = x \ll y$
>>=	$x \gg = y$	Данная запись эквивалентна $x = x \gg y$
<i>Адреса и указатели</i>		
&	$\&x$	Унарная операция получения адреса переменной x .
*	$*x$	Унарная операция объявления указателя. Указатель – это переменная, которая содержит адрес другой переменной.

Функции вывода-вывода данных в текстовом режиме

Для вывода данных на консоль в языке Си используется функция форматного вывода строки `printf`. Для того, чтобы вывести на консоль значения переменных необходимо в строке, выводимой функцией `printf` на экран использовать специальные флаги (спецификаторы ввода-вывода).

Таблица 5 – Основные спецификаторы ввода/вывод на консоль

<i>Флаг</i>	<i>Назначение</i>
%d	Переменная типа <code>int</code>
%o	Переменная типа <code>int</code> в восьмеричном представлении
%u	Переменная типа <code>unsigned int</code>
%x	Переменная типа <code>int</code> шестнадцатеричном представлении
%f	Переменная вещественного типа (<code>float</code>)
%lf	Переменная вещественного типа (<code>double</code>)

%e	Переменная вещественного типа в экспоненциальном формате
%a	Переменная вещественного типа в шестнадцатеричном виде
%c	Переменная символьного типа
%s	Строка (последовательность символов)

Следующий пример демонстрирует вывод переменных разных типов на консоль:

```
#include <stdio.h>
int main()
{
    int a = 1;
    float b = 1.0F;
    char c = 'a';
    printf("\na=%d\nb=%f\tb=%e\nc=%c\n", a, b, b, c);
    return 0;
}
```

Логика работы функции `printf` такова. Анализируется первый аргумент (строка в двойных кавычках) и, пока в ней не встретится спецификатор вывода, очередной символ строки выводится на экран (помните, что эскейп-последовательности интерпретируются как символы). Если очередным символом строки является спецификатор вывода, то вместо него на экран будет выведено значение соответствующей ему переменной. При этом порядок спецификаторов вывода в форматной строке соответствует порядку переменных, указываемых после нее. Таким образом, с помощью `printf` можно вывести на экран значение одной или нескольких переменных, или даже значение одной и той же переменной в различных форматах. Для ввода данных с клавиатуры в языке C обычно используется функция `scanf`. Для вывода переменных используются те же спецификаторы, что и для функции `printf`. Пример модификации предыдущего примера с вводом переменных с консоли:

```
#include <stdio.h>
int main()
{
    int a;
    float b;
    char c;
    printf("\nInput integer, float and character:");
    scanf("%d%f%c", &a, &b, &c);
    printf("\na=%d\nb=%f\tb=%e\nc=%c\n", a, b, b, c);
}
```

```
    return 0;
}
```

Здесь первый аргумент функции `scanf` (то, что указано до запятой в круглых скобках) - спецификатор ввода, а второй аргумент - указание на то, в какую переменную поместить введенное пользователем значение. Функция `scanf` может быть использована для ввода значений нескольких переменных сразу. Обратите внимание, что в форматной строке нет пробелов¹⁴ - спецификаторы написаны сразу друг за другом. Отметим, что хорошим тоном при программировании с использованием функции `scanf` считается предварительный вывод сообщения для пользователя о том, что конкретно он должен ввести с консоли.

Контрольные вопросы

1. Какие типы данных существуют в языке Си? Что можно сказать об их размерах? Приведите примеры объявления и инициализации переменных различных типов.
2. Как записываются константы различных типов в языке Си? Приведите примеры.
3. Какие арифметические операции существуют в языке Си? Каков порядок (ранг) выполнения этих операций в алгебраических выражениях. Приведите примеры.
4. Какие побитовые операции существуют в языке Си? Напишите программу, реализующие побитовые операции.
5. Какие спецификаторы ввода-вывода Вы знаете? Как работают функции `printf` и `scanf` с переменными?
6. Какие типы комментариев существуют в языке Си? Для чего используются комментарии в программе.
7. Определите с помощью оператора `sizeof` размер памяти, отводимый под хранение переменных разного типа на вашем компьютере. Определите диапазон допустимых значений для каждого типа данных.
8. Объясните, почему перед функцией `scanf` следует использовать функцию `printf` с описанием того, что требуется от пользователя?
9. Допустимо ли в строковой части функции `scanf` использование разнообразных символов и эскейп-последовательностей? Объясните почему.
10. Какими правилами следует руководствоваться при выборе имен переменных в языке Си?

¹⁴ В отличие от функции `printf`, в функции `scanf` в символьной строке **крайне нежелательно** использование любых символов, отличных от спецификаторов ввода! Если же Вы их все-таки используете, то должны четко понимать: что, почему и зачем именно так Вы делаете.

Лабораторная работа №5

Условные операторы

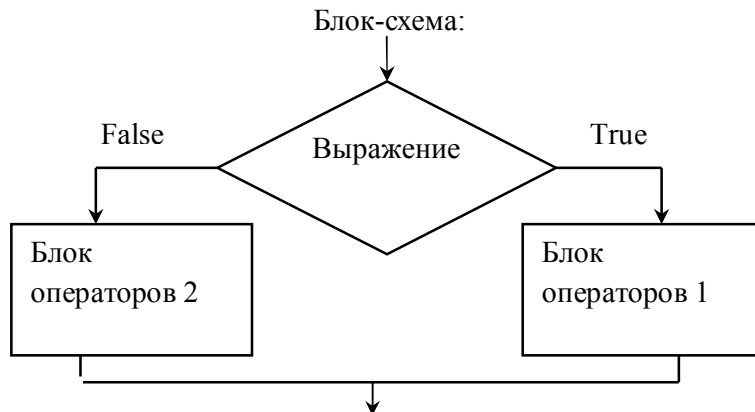
Цель работы: изучение условных конструкций языка Си и приобретение практических навыков работы с ними.

Оператор *if*

Конструкция `if-else` используется при использовании условий. Синтаксис в общем случае (полное ветвление) выглядит следующим образом:

Синтаксис языка Си:

```
if (Выражение)
    Блок операторов 1;
else
    Блок операторов 2;
```



При вычислении выражение может принимать значения `True` (истина, или $\neq 0$) и `False` (ложь, или 0). Пример выражения: $x > y$. В случае, если значение выражения `True` - выполняется блок операторов 1, в противном случае выполняется блок операторов 2. Таким образом, в случае проверки на неравенство нулю численных значений конструкцию оператора ветвления можно упростить:

```
y=x+1+x*x; // Вычисляем значение выражения y
if (y)
{
    блок операторов 1; // выполняется, если y отличен от 0
}
else
{
    блок операторов 2; // выполняется, если y равен 0
}
```

Ниже приведена таблица 6 с указанием операций сравнения целых и вещественных чисел, используемых при составлении логических выражений:

Таблица 6 - Операторы сравнения

Название	Синтаксис	Примечание
Равенство	==	A == B
Неравенство	!=	A != B
Больше	>	A > B
Меньше	<	A < B
Больше или равно	>=	A >= B
Меньше или равно	<=	A <= B

Помимо указанных в таблице 6 операций сравнения на практике часто возникает необходимость одновременного выполнения нескольких условий. Например, если в задаче используется функция $f(x) = \frac{1}{x-1} + \frac{1}{x+4}$, в области $x \in (-4; 1)$ или $-4 < x < 1$. При написании программы такие условия должны быть переформулированы с использованием логических операторов НЕ, ИЛИ, И (Таблица 7).

Таблица 7 - Логические операции

Название	Синтаксис	Примечание
НЕ	!	!A
ИЛИ		A B
И	&&	A && B

В таблице 7 в качестве операндов A и B выступают логические переменные (т.е. A и B могут принимать два значения: True и False), в отличие от таблицы 1, где A и B – это переменные вещественного или целого типов. Таким образом, условие $-4 < x < 1$ в программе должно быть записано в следующем виде: $(x > -4) \&\& (x < 1)$.

Рассмотрим работу условного оператора if-else на примере вычисления корней квадратного уравнения $ax^2 + bx + c = 0$. Программа, вычисляющая действительные корни уравнения, должна вычислять дискриминант по формуле $D = b^2 - 4ac$. В случае, когда $D < 0$, пользователь должен получить сообщение об отсутствии действительных корней уравнения «There are no real roots of equation!».

```

/*****
Calculation of roots of equation ax^2+bx+c=0
*****/

#include <stdio.h>
#include <math.h>

```

```

int main()
{
    double a,b,c,x,D,x1,x2;
    a=0.5;
    b=1.0;
    c=1.1;
    D=b*b-4.0*a*c; // Discriminant calculation
    if (D == 0.0)
    {
        printf("\nThere are no real roots of equation!\n");
        return 0;
    }
    else // if (D != 0.0)
    {
        x1=(-b+sqrt(D))/(2.0*a);
        x2=(-b-sqrt(D))/(2.0*a);
        printf("\nThe roots are x1=%.4f\tx2=%.4f\n",x1,x2);
    }
    return 0;
}

```

Здесь следует отметить появление функции вычисления корня `sqrt(D)`, для вычисления которой необходимо подключить заголовочный файл математической библиотеки `math.h`.

Следует отметить, что оператор `else` является опциональным, и его можно опускать, когда в случае невыполнения выражения не нужно выполнять никаких действий.

Вложенные операторы

Операторы условия могут быть вложенными друг в друга в соответствии с условиями задачи, которая решается программистом. В общем случае степень вложенности операторов может быть произвольной. В качестве иллюстрации приведем код программы, которая ищет минимальное из трех чисел a, b, c .

```

/*****
    Searching for minimum among a, b and c
    *****/
#include <stdio.h>

```

```

int main()
{
    int a,b,c;
    printf("\nInsert three integer numbers a, b and c:");
    scanf("%d%d%d", &a, &b, &c);
    if (a<b) // Первый уровень
    {
        if(a<c) // Второй уровень
            printf ("\nMinimum is a=%d\n", a);
        else
            printf ("\nMinimum is c=%d\n", c);
    }
    else
    {
        if (b<c) // Второй уровень
            printf ("\nMinimum is b=%d\n", b);
        else
            printf ("\nMinimum is c=%d\n", c);
    }
    return 0;
}

```

Отметим также важность использования фигурных скобок на первом уровне вложения (проверьте работу программы при отсутствии данных скобок). В случае отсутствия данных скобок мы получим, что первый же `else` будет относиться уже к первому уровню вложенности, а не ко второму, последующие же два `else` будут генерировать сообщение компилятора об ошибке. Также важным моментом является возможность неиспользования фигурных скобок для второго уровня вложенности, поскольку используется только одна функция вывода на печать.

Оператор switch

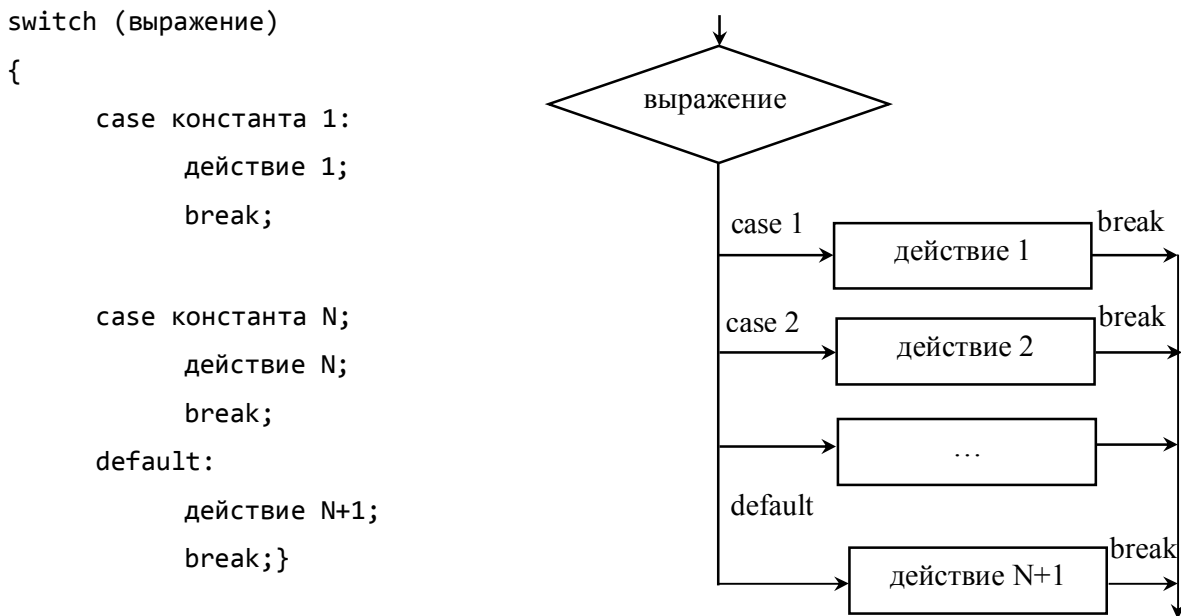
Оператор `switch` используется, когда предполагается выбор одного из нескольких возможных путей выполнения программы. Отметим, что данный оператор работает только с выражениями, результатом которых является значение целого или символьного типа. Выражения, которые следуют за ключевым словом `case`, должны быть константными выражениями целого (для переменных целого типа) либо символьного (для переменных символьного типа) типа.

Поясним работу оператора выбора switch:

1. Вычисляется значение выражения;
2. Если значение выражения совпадает со значением одной из констант после ключевого слова `case`, тогда выполняются соответствующие действия, предусмотренные данной ветвью, а затем оператор `break` прерывает работу оператора `switch`.
3. Если же значение переменной не совпало ни с одним из константных значений, то в этом случае выполняются действия, предусмотренные в блоке `default` (по умолчанию).

Отметим, что использование сигнификатора `default` необязательно.

В общем случае синтаксис оператора `switch` следующий:



Приведем пример использования конструкции `switch` для реализации простейшего калькулятора:

```
/******
Simplest calculator
******/
#include <stdio.h>
int main()
{
    char symbol;
    int a,b;
    printf ("\nInput integer numbers a & b\n");
    printf ("Use following syntax a operator b\n");
```

```

printf ("You can use following operators: + - / *\n");
scanf ("%d%c%d",&a, &symbol, &b);
switch (symbol)
{
    case '+':
        printf("\n%d + %d = %d\n", a, b, a+b);
        break;
    case '-':
        printf("\n%d - %d = %d\n", a, b, a-b);
        break;
    case '/':
        printf("\n%d / %d = %d\n", a, b, a/b);
        break;
    case '*':
        printf("\n%d * %d = %d\n", a, b, a*b);
        break;
    default:
        printf("\nCannot resolve operator\n");
}
return 0;
}

```

Как же реализовать задачу множественного выбора для чисел с плавающей точкой? В данном случае необходимо использовать конструкции типа `if-else if`. В общем случае конструкция будет выглядеть следующим образом:

```

if (выражение 1)
    блок операторов 1;
else if (выражение 2)
    блок операторов 2;
else if (выражение 3)
    блок операторов 3;
...
else if (выражение N)
    блок операторов N;
else
    блок операторов N+1;

```

Контрольные вопросы

1. Поясните на примере выполнение полного оператора ветвления `if-else`.
2. Поясните на примере выполнение неполного оператора ветвления `if-else`.
3. Поясните на примере выполнение оператора множественного выбора `switch case`.
4. Как организовать в языке Си конструкцию множественного выбора для чисел с плавающей точкой?
5. Как будет работать оператор `switch` если из тела оператора убрать все операторы `break`?
6. Можно ли не использовать метку `default` при работе с оператором `switch` и почему?
7. Что Вы знаете о математических функциях в языке Си?
8. Как рассчитать с помощью Си-программы значения тригонометрических функций угла $\varphi = 125^\circ$?
9. При работе программы, рассчитывающей арктангенс x , получено значение `0.785398`. Поясните, что это означает?
10. Как будет выглядеть программа простейшего калькулятора, приведенная в тексте лабораторной работы, если использовать оператор `if-else`?
11. Запишите пример программы нахождения минимума из трех чисел с помощью оператора `if-else if`.
12. Запустите программы, приведенные в примерах к данной лабораторной работе, и объясните, какие действия они выполняют.

Лабораторная работа №6

Циклы

Цель работы: знакомство с операторами циклов языка Си и приобретение практических навыков по работе с ними.

Циклы позволяют осуществлять многократное выполнение каких-либо действий. Каждое повторение цикла называется итерацией. Цикл всегда имеет точку входа и проверочное условие. Для завершения цикла необходимо также наличие точки выхода, в противном случае циклы называют бесконечными. Для бесконечных циклов проверочное условие всегда истинно.

Цикл со счетчиком for

Синтаксис цикла с заранее известным числом повторений имеет следующий вид:

```
for (выражение_1; выражение_2; выражение_3)
{
    Блок операторов;
}
```

Выражение_1 чаще всего служит в качестве инициализации какой-нибудь переменной, выполняющей роль счетчика итераций.

Выражение_2 используется как проверочное условие выхода из цикла и на практике обычно содержит выражение, содержащее операторы сравнения. Предполагается, что *выражение_2* выбрано таким образом, чтобы цикл выполнялся хотя бы один раз.

Выражение_3 служит чаще всего для приращения значения счетчика циклов, либо влияет на изменение значения счетчика итераций и выполнение (или невыполнение) *выражения_2*.

Все три выражения не обязательно должны присутствовать в конструкции, однако правильный синтаксис обязательно предполагает наличие оператора точка с запятой (;). Простейший пример бесконечного цикла:

```
for (;;)
    printf("\nInfinite cycle!");
```

Данный цикл будет выполняться до принудительного завершения программы.

Если в цикле должны одновременно изменяться несколько переменных, которые зависят от переменной цикла, вычисление их значений можно поместить в оператор `for`, воспользовавшись оператором запятой (,).

Приведем пример цикла, позволяющего вычислить сумму ряда

$$S = \sum_{i=1}^{10} i^2 = 1 + 2^2 + 3^2 + \dots + 10^2.$$

```
#include <stdio.h>
int main()
{
    int i, S=0;
    for(i=1; i<=10; i++)
        S+=i*i;
    printf("\nSum of the sequence S=%d\n", S);
    return 0;
}
```

Следующий пример иллюстрирует одновременное изменение в цикле двух переменных:

```
#include<stdio.h>
int main()
{
    int i, j;
    const int N=10;
    for(i=1,j=1; i<=N;i++,j+=4)
        printf("\n%d\t%d\n", i,j);
    return 0;
}
```

Циклы с условием - while и do-while

Оператор цикла while выполняет блок операторов до тех пор, пока проверочное условие (выражение) остается истинным. Цикл while реализует цикл «пока» с предусловием.

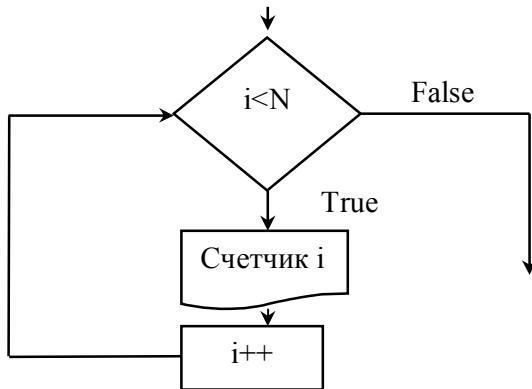
Общий вид цикла с предусловием:

```
while(выражение)
{
    Блок операторов;
}
```

Общий вид цикла с постусловием:

```
do
{
    Блок операторов;
} while(выражение);
```

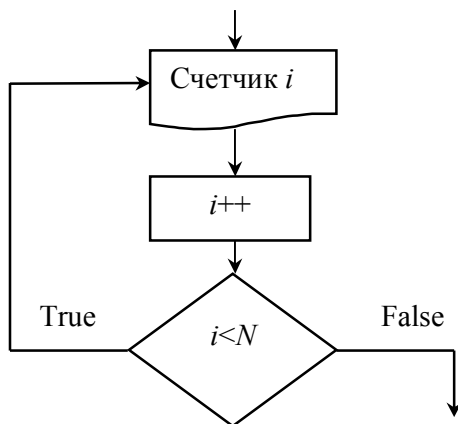

Пример использования цикла do-while:



```
#include<stdio.h>

int main()
{
    int i=1, N=10;
    while (i<N)
    {
        printf("\nCounter %d",i);
        i++;
    }
    return 0;
}
```

В отличие от цикла while, цикл do-while сначала выполняет тело цикла (блок операторов), а затем проверяет истинность выражения. Такая конструкция цикла do-while гарантирует, что тело цикла выполнится хотя бы один раз.



```
#include<stdio.h>

int main()
{
    int i=1, N=10;
    do
    {
        printf("\nCounter %d",i);
        i++;
    }while(i<N);
    return 0;
}
```

Приведем в качестве еще одного примера вычисление суммы бесконечного ряда $S = \sum_{n=1}^{\infty} \frac{1}{n^2}$

/******

Программа вычисления суммы бесконечного ряда с заданной точностью

*****/

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

```
int main()
```

```
{
```

```
    int n=1;           // Начальное значение счетчика
```

```
    float S=0.0,eps,t; // Вводим необходимые переменные для хранения суммы ряда S,
```

```
                      // точности, с которой выполняется расчет суммы eps
```

```
                      // и текущего слагаемого ряда t
```

```

printf("\nInput calculation precision eps=");
scanf("%f",&eps);
do
{
    t=1.0/(float)(n*n); // Здесь важным является приведение переменной n к
                        // типу float
    S+=t;               // Вычисление суммы
    n++;               // Изменение счетчика цикла
}while(t>eps); // Проверка условия: текущий член ряда больше заданной точности
printf("\nThe sum of sequence S=%f\n\n",S);
return 0;
}

```

Если выражение представляет собой константу с истинным значением (True или $\neq 0$), то тело цикла будет выполняться всегда, в этом случае мы будем иметь дело с бесконечным циклом. Выполнение цикла завершается, как только выражение становится ложным (False или 0). Цикл также окажется бесконечным, когда условие, определяемое выражением изначально истинно и далее нигде в цикле не выполняются действия, направленные на изменение значения выражения. Пример бесконечного цикла `while`:

```

while (1)
{
    Блок операторов;
}

```

Естественным вопросом в таком случае будет, как избежать выполнения бесконечного цикла, предусмотрев выход из цикла в нескольких местах? В этом случае эффективным будет использование оператора `break`. В теле любого цикла также можно использовать оператор `continue`. Оператор `break` позволяет выйти из цикла, передавая управление следующему за циклом оператору. Оператор `continue` позволяет пропустить часть операторов тела цикла и начать новую итерацию. Оператор `continue` удобен, когда от итерации к итерации изменяется последовательность выполняемых операторов тела цикла, то есть когда тело цикла содержит ветвления.

Вывести числа от 0 до 99 исключая числа, оканчивающиеся на 5 или 8:	Вывести числа от 0 до 99 ниже главной диагонали:
<pre>#include <stdio.h> int main() { unsigned int i,j; for(i=0; i<10; i++) { for(j=0; j<10; j++) { if((j!=8) && (j!=5)) continue; printf("%2d ",i*10+j); } printf("\n"); } return 0; }</pre>	<pre>#include <stdio.h> int main() { unsigned int i,j; for(i=0; i<10; i++) { for(j=0; j<10; j++) { if(j>i) break; printf("%2d ",i*10+j); } printf("\n"); } return 0; }</pre>

Вложенные циклы

Помимо описанных выше конструкций для циклов также возможно вложение циклов друг в друга.

Следующий пример показывает расчет суммы ряда (двойной суммы):

$$\sum_{i=1}^{20} \sum_{j=1}^{10} \frac{1}{i^2 + j}.$$

```
#include <stdio.h>
int main()
{
    const int M=10, N=20;
    float s=0.0;
    int i,j;
    printf("\nПрограмма вычисления двойной суммы\n");
    for(i=1;i<=N;i++)
    {
        for(j=1;j<=M;j++)
        {
            s+=1.0/(i*i+j);
        }
    }
    printf("\n Sum=%f \n", s);
}
```

Контрольные вопросы

1. Опишите синтаксис цикла `while`, приведите примеры.
2. Опишите синтаксис цикла `do-while`, приведите примеры.
3. Составьте блок-схему, описывающую работу программы из примера расчета суммы бесконечного ряда.
4. Опишите выполнение оператора цикла `for` и приведите пример работы данного цикла.
5. Как работает оператор `break`? Приведите пример.
6. Как работает оператор `continue`? Приведите пример.
7. Как изменится работа программы из последнего примера для вычисления двойной суммы, если в строке суммирования записать $s+=1/(i*j);$?
8. Что такое бесконечный цикл? Как происходит выход из бесконечного цикла?
9. Запустите программы, приведенные в примерах к данной лабораторной работе, и объясните, какие действия они выполняют.
10. Подумайте, в каких случаях в циклах следует применять `break`. Является ли хорошим стилем программирования использование данного оператора в готовом программном продукте. Дайте развернутый ответ.

Приложение А

Варианты заданий

1. Даны действительные числа x и y . Меньшее из этих двух чисел заменить их полусуммой, а большее – их удвоенным произведением.
2. Даны действительные числа x и y . Если x и y отрицательны, то каждое из этих чисел заменить его модулем, если отрицательно только одно из этих чисел, то оба числа умножить на 0.5; если оба значения неотрицательны, и ни одно из них не принадлежит отрезку $[0.5; 2.0]$, то оба значения уменьшить в 10 раз; во всех остальных случаях x и y оставить без изменения.
3. Вычислить площадь треугольника со сторонами a , b и c по формуле Герона, предварительно проверив условие корректности исходных данных (длины всех сторон положительны, сумма длин любых двух сторон больше длины третьей).
4. Определить время падения камня на поверхность Земли с высоты h . При вводе данных проверять условие корректности исходных данных.
5. Вычислить площадь треугольника по его стороне a и высоте h . При вводе данных проверять условие корректности исходных данных.
6. Вычислить площадь окружности по заданному радиусу. При вводе данных проверять условие корректности исходных данных.
7. Даны два числа a и b , найти их среднее арифметическое и среднее геометрическое.
8. Вычислить высоту треугольника, зная две стороны треугольника и угол между ними. При вводе данных проверять условие корректности исходных данных.
9. Вычислить радиус вписанной в квадрат со стороной a окружности. При вводе данных проверять условие корректности исходных данных.
10. Вычислить углы треугольника, зная длины его сторон. При вводе данных проверять условие корректности исходных данных.
11. Вычислить площадь трапеции. При вводе данных проверять условие корректности исходных данных.
12. Вычислить длину гипотенузы прямоугольного треугольника, зная длины его катетов. При вводе данных проверять условие корректности исходных данных.
13. Вычислить объем цилиндра, зная его радиус R и высоту h . При вводе данных проверять условие корректности исходных данных.
14. Вычислить объем конуса. При вводе данных проверять условие корректности исходных данных.
15. Вычислить сторону треугольника, зная две другие стороны и угол между ними. При

вводе данных проверять условие корректности исходных данных.

16. Вычислить площадь ромба, зная длину стороны и угол. При вводе данных проверять условие корректности исходных данных.
17. Вычислить площадь треугольника, зная длины его сторон и радиус описанной окружности. При вводе данных проверять условие корректности исходных данных.
18. Вычислить высоту равностороннего треугольника, зная длину всех его сторон. При вводе данных проверять условие корректности исходных данных.
19. Вычислить длину отрезка, зная координаты его концов. При вводе данных проверять условие корректности исходных данных.
20. Вычислить среднее геометрическое пяти вводимых чисел. При вводе данных проверять условие корректности исходных данных.
21. Даны действительные числа $a_1, b_1, c_1, a_2, b_2, c_2$. Выяснить верно ли, что $|a_1b_2 - a_2b_1| \geq 0.0001$, и если верно, найти решение системы

$$\begin{aligned} a_1x + b_1y + c_1 &= 0 \\ a_2x + b_2y + c_2 &= 0 \end{aligned}$$

22. Определить полярные координаты точки (r, φ) , которые можно вычислить, зная декартовы координаты точки (x, y) , по формулам $\varphi = \arctg(y/x)$, $r = \sqrt{x^2 + y^2}$. При вычислении угла φ учесть, что значение r может быть равно 0, а угол может находиться в разных четвертях.
23. Напишите программу, вычисляющую значения функции. Спецификация должна содержать указание области допустимых значений функции.

а) $y = \arccos(1 - 2x)$

б) $y = \arcsin(\sqrt{1 - 4x})$

в) $y = \ln(\sqrt{x} - \sqrt{x-1})$

г) $y = \frac{1}{x + \sqrt{x^2 - 1}}$

д) $y = \arcsin(e^{4x})$

е) $y = \arctg^2\left(\frac{1}{x}\right)$

ж) $y = \sqrt[5]{\ln \sin \frac{x+3}{4}}$

з) $y = \ln^5 \sin x$

и) $y = \arccos\left(e^{-\frac{x^2}{2}}\right)$

к) $y = \ln \arccos(2x)$

л) $y = \arctg\left(\frac{x+3}{x-3}\right)$

м) $y = \arctg \ln(5x + 3)$

н) $y = \frac{2}{3} \sqrt{(1 + \ln x)^3}$

п) $y = \sqrt{1 - \arccos^2 x}$

р) $y = \sqrt[5]{\arctg(e^{5x})}$

с) $y = e^{\sqrt{1 + \ln x}}$

т) $y = x^2 \arcsin(\sqrt{1 - x^2})$

у) $y = \sqrt{3} \left(\arctg \frac{x}{\sqrt{3}} + \ln \frac{x - \sqrt{3}}{x + \sqrt{3}} \right)$

ф) $y = \ln \sqrt{\frac{e^{4x}}{e^{4x} + 1}}$

х) $y = \log_5 \cos(7x)$

ц) $y = \log_7 \cos(\sqrt{1 + x})$

ч) $y = \frac{\ln \sin x}{x + \pi}$

ш) $y = \arcsin \sqrt{x}$

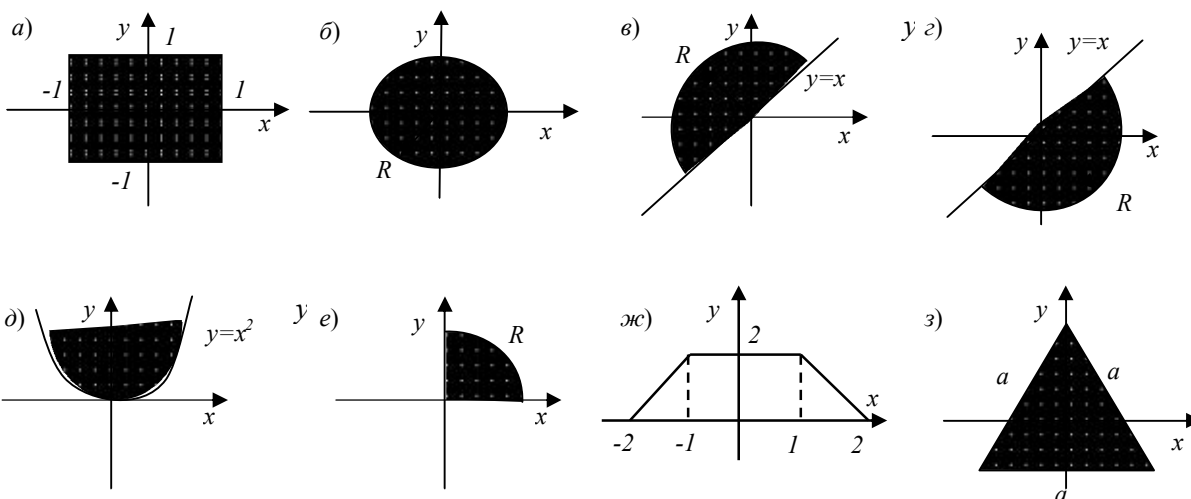
щ) $y = \arcsin(\sqrt{\sin x})$

ы) $y = \ln \sqrt{\frac{1 + e^{2x}}{e^{2x} - 1}}$

$$o) y = \arctg(\sqrt{6x-1})$$

$$э) y = \sqrt[5]{\ln \sin \frac{x+3}{4}}$$

24. Напишите программу, которая моделирует стрельбу по мишени: пользователь задает с консоли два числа x и y . Программа выдает ответ, попала ли точка в заштрихованную область.



25. Напишите программу рассчитывающую значение функции для любого x , задаваемого пользователем с консоли¹⁵.

$$a) f = \begin{cases} ax^2 + b & \text{при } x < 0 \text{ и } b \neq 0 \\ \frac{x-a}{x-b} & \text{при } x > 0 \text{ и } b = 0. \\ \frac{x}{c} & \text{в других случаях} \end{cases}$$

$$e) f = \begin{cases} -x^2 & \text{при } x < 0 \text{ и } b \neq 0 \\ \frac{a-x}{cx} & \text{при } x > 0 \text{ и } b = 0. \\ \frac{x}{c} & \text{в других случаях} \end{cases}$$

$$б) f = \begin{cases} -\frac{2x-c}{cx-a} & \text{при } x < 0 \text{ и } b \neq 0 \\ \frac{x-a}{x-c} & \text{при } x > 0 \text{ и } b = 0. \\ -\frac{x}{c} - \frac{c}{2x} & \text{в других случаях} \end{cases}$$

$$ж) f = \begin{cases} -x^2 - b & \text{при } x < 0 \text{ и } b \neq 0 \\ \frac{a-x}{x} & \text{при } x > 0 \text{ и } b = 0. \\ -\frac{x}{c} & \text{в других случаях} \end{cases}$$

$$в) f = \begin{cases} dx + bx^2 & \text{при } x < 0 \text{ и } b \neq 0 \\ \frac{x-a}{x-c} & \text{при } x > 0 \text{ и } b = 0. \\ \frac{x+5}{c(x-10)} & \text{в других случаях} \end{cases}$$

$$з) f = \begin{cases} x^2, & \text{при } -2 < x \leq 2, \\ 4, & \text{в других случаях.} \end{cases}$$

$$г) f = \begin{cases} a(x+c)^2 & \text{при } x < 0 \text{ и } b \neq 0 \\ \frac{x-a}{-c} & \text{при } x > 0 \text{ и } b = 0. \\ a - \frac{x}{c} + b & \text{в других случаях} \end{cases}$$

$$и) f = \begin{cases} 0, & \text{при } x < 0 \\ x^2 - x & \text{при } 0 < x \leq 1 \\ x - \sin(\pi x^2) & \text{в других случаях} \end{cases}$$

$$д) f = \begin{cases} ax^2 + b^2x & \text{при } x < 0 \text{ и } b \neq 0 \\ x - \frac{a}{x-c} & \text{при } x > 0 \text{ и } b = 0. \\ 1 + \frac{x}{c} & \text{в других случаях} \end{cases} f =$$

$$к) f = \begin{cases} \sqrt{a-x} & \text{при } a < x \\ \sqrt{|a-x|} & \text{при } a > x. \\ 0 & \text{в других случаях} \end{cases}$$

$$л) f = \begin{cases} -ax^3 - b & \text{при } x < 0 \text{ и } b \neq 0 \\ \frac{x-a}{x-c} & \text{при } x > 0 \text{ и } b = 0. \\ \frac{x}{c} + \frac{c}{x} & \text{в других случаях} \end{cases}$$

¹⁵ Для лабораторной работы 6: вычислить и вывести на экран в виде таблицы значения функции $f(x)$ на интервале $x \in [a, b]$ с шагом dx . Значения переменных a, b, c, d, e, dx пользователь вводит с клавиатуры.

$$\begin{cases} 0, & \text{при } x \leq -3 \\ -x-2, & \text{при } -3 \leq x \leq -1 \\ x, & \text{при } -1 \leq x < 1 \\ -x+2, & \text{при } 1 \leq x < 2 \\ 0, & \text{при } x \geq 2 \end{cases} \quad \text{м) } f = \begin{cases} ax^2 + \frac{b}{c} & \text{при } x < 0 \text{ и } b \neq 0 \\ \frac{x-a}{(x-c)^2} & \text{при } x > 0 \text{ и } b = 0. \\ \frac{x^2}{c^2} & \text{в других случаях} \end{cases}$$

26. Напишите программу извлечения корня с любым натуральным показателем k из положительного числа X с заданной точностью $1 > \varepsilon > 0$. Для вычисления воспользуйтесь формулой:

$$x_{i+1} = \frac{X + (k-1)x_i^k}{kx_i^{k-1}}$$

27. Вычислите число π с заданной точностью $1 > \varepsilon > 0$. Для вычисления используйте периметр 2^{n+1} -угольника, вписанного в ту же окружность:

$$P_{n+1} = 2^{n+1}R \sqrt{2 - \sqrt{4 - \frac{P_n^2}{2^{2n}R^2}}}$$

учитывая что $P \rightarrow 2\pi R$ при $n \rightarrow \infty$. Периметр вписанного в данную окружность квадрата вычисляется по формуле $P_2 = 4\sqrt{2}R$.

28. Вычислите число π с заданной точностью $1 > \varepsilon > 0$. Для расчета воспользуйтесь следующими соображениями: если $r_2 = \frac{\sqrt{2}}{2}$, $l_2 = \frac{1}{4}$ и $l_{n+1} = \frac{r_n + l_n}{2}$, $r_n = \sqrt{r_n l_{n+1}}$, то $\lim_{n \rightarrow \infty} l_n = \frac{1}{\pi}$. Расчет продолжать до тех пор, пока значения (полученные в двух последующих итерациях) не совпадут в пределах заданной точности.

29. Вычислите e^x с заданной точностью $1 > \varepsilon > 0$, используя формулу разложения в ряд Тейлора

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \dots + \frac{x^n}{n!} + \dots$$

Сравните полученный результат с вычислением данной функции с помощью математической библиотеки.

30. Вычислите $\ln\left(\frac{x+1}{x-1}\right)$ с заданной точностью $1 > \varepsilon > 0$, используя формулу разложения в ряд Тейлора ($|x| > 1$)

$$\ln\left(\frac{x+1}{x-1}\right) = 2 \sum_{n=0}^{\infty} \frac{1}{(2n+1)x^{2n+1}} = 2\left(\frac{1}{x} + \frac{1}{3x^3} + \frac{1}{5x^5} + \dots\right).$$

Сравните полученный результат с вычислением данной функции с помощью математической библиотеки.

31. Вычислите $\ln\left(\frac{1+x}{1-x}\right)$ с заданной точностью $1 > \varepsilon > 0$, используя формулу разложения в ряд Тейлора ($|x| < 1$)

$$\ln\left(\frac{x+1}{x-1}\right) = 2 \sum_{n=0}^{\infty} \frac{x^{2n+1}}{2n+1} = 2\left(x + \frac{x^3}{3} + \frac{x^5}{5} + \dots\right).$$

Сравните полученный результат с вычислением данной функции с помощью математической библиотеки.

32. Вычислите e^x с заданной точностью $1 > \varepsilon > 0$, используя формулу разложения в ряд Тейлора ($|x| < \infty$)

$$e^{-x} = \sum_{n=0}^{\infty} \frac{(-1)^n x^n}{n!} = 1 - x + \frac{x^2}{2!} - \dots + \frac{(-1)^n x^n}{n!} + \dots.$$

Сравните полученный результат с вычислением данной функции с помощью математической библиотеки.

33. Вычислите $\cos(x)$ с заданной точностью $1 > \varepsilon > 0$, используя формулу разложения в ряд Тейлора ($|x| < \infty$)

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!} = 1 - \frac{x^2}{2!} + \dots + (-1)^n \frac{x^{2n}}{(2n)!} + \dots.$$

Сравните полученный результат с вычислением данной функции с помощью математической библиотеки.

34. Вычислите $\sin(x)$ с заданной точностью $1 > \varepsilon > 0$, используя формулу разложения в ряд Тейлора

$$\sin(x) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} x^{2n-1}}{(2n-1)!} = x - \frac{x^3}{3!} + \dots + (-1)^{n-1} \frac{x^{2n-1}}{(2n-1)!} + \dots.$$

Сравните полученный результат с вычислением данной функции с помощью математической библиотеки.

35. Вычислите $\ln(1+x)$ с заданной точностью $1 > \varepsilon > 0$, используя формулу разложения в ряд Тейлора ($|x| < 1$)

$$\ln(x+1) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1} x^n}{n!} = x - \frac{x^2}{2!} + \dots + \frac{(-1)^{n-1} x^n}{n!} + \dots.$$

Сравните полученный результат с вычислением данной функции с помощью математической библиотеки.

36. Вычислите $\ln(1-x)$ с заданной точностью $1 > \varepsilon > 0$, используя формулу разложения в ряд Тейлора ($|x| < 1$)

$$\ln(1-x) = -\sum_{n=1}^{\infty} \frac{x^n}{n!} = -\left(x + \frac{x^2}{2!} + \dots + \frac{x^n}{n!} + \dots\right).$$

Сравните полученный результат с вычислением данной функции с помощью математической библиотеки.

37. Вычислите $\arctg(x)$ с заданной точностью $1 > \varepsilon > 0$, используя формулу разложения в ряд Тейлора ($x > 1$)

$$\arctg(x) = \frac{\pi}{2} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1}}{(2n+1)x^{2n+1}} = \frac{\pi}{2} - \frac{1}{x} + \frac{1}{3x^3} - \frac{1}{5x^5} + \dots$$

Сравните полученный результат с вычислением данной функции с помощью математической библиотеки.

38. Вычислите $\arctg(x)$ с заданной точностью $1 > \varepsilon > 0$, используя формулу разложения в ряд Тейлора ($|x| \leq 1$)

$$\arctg(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots$$

Сравните полученный результат с вычислением данной функции с помощью математической библиотеки.

39. Вычислите e^{-x^2} с заданной точностью $1 > \varepsilon > 0$, используя формулу разложения в ряд Тейлора ($|x| < \infty$)

$$e^{-x^2} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{n!} = 1 - x^2 + \frac{x^4}{2!} - \frac{x^6}{3!} + \frac{x^8}{4!} - \dots$$

Сравните полученный результат с вычислением данной функции с помощью математической библиотеки.

40. Вычислите $\sin(x)/x$ с заданной точностью $1 > \varepsilon > 0$, используя формулу разложения в ряд Тейлора ($|x| < \infty$)

$$\frac{\sin(x)}{x} = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n+1)!} = 1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!} + \dots$$

Сравните полученный результат с вычислением данной функции с помощью математической библиотеки.

41. Вычислите $\ln(x)$ с заданной точностью $\varepsilon > 0$, используя формулу разложения в ряд Тейлора ($x > 1/2$)

$$\ln(x) = \sum_{n=0}^{\infty} \frac{(x-1)^{n+1}}{(n+1)(x+1)^{n+1}} = \frac{x-1}{x} - \frac{(x-1)^2}{2(x+1)^2} + \dots$$

Сравните полученный результат с вычислением данной функции с помощью

математической библиотеки.

42. Найти сумму ряда $\sum_{n=0}^{\infty} \frac{x^n}{2^n}$, если $|x| \leq 1$, с заданной точностью $1 > \varepsilon > 0$. Результат сравните с точным решением.

43. Для заданных действительных чисел $x \neq 0, 1 > \varepsilon > 0$ найти сумму ряда $\sum_{n=0}^{\infty} \frac{x^{2n}}{2^n n!}$ с заданной точностью ε .

44. Для заданных действительных чисел $x \neq 0, 1 > \varepsilon > 0$ найти сумму ряда $\sum_{n=0}^{\infty} \frac{(-x)^{2n+1}}{2n!}$ с заданной точностью ε . Найдите количество учтенных слагаемых.

45. Для заданных действительных чисел $x \neq 0, 1 > \varepsilon > 0$ найти сумму ряда $\sum_{n=0}^{\infty} \frac{(-1)^n x^{n+1}}{(n+1)(2+n)!}$ с заданной точностью ε . Найдите количество учтенных слагаемых.

46. Для заданных действительных чисел $x \neq 0, 1 > \varepsilon > 0$ найти сумму ряда $\sum_{n=0}^{\infty} \frac{\cos^3(3^{n-1})}{3^n}$, учитывая только те слагаемые, в которых множитель $1/3^n$ имеет величину не меньшую ε . Найдите количество учтенных слагаемых.

47. Вычислите сумму членов ряда

$$z = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \dots + \frac{x^{20}}{20!} = 1 + \sum_{i=1}^{10} \frac{x^{2i}}{(2i)!},$$

для вычислений используйте рекуррентную формулу

$$z_n = z_{n-1} \frac{x^2}{2n(2n-1)}.$$

48. Вычислите сумму членов ряда

$$z = 1 - x + \frac{x^3}{3!} - \frac{x^5}{5!} + \dots + \frac{x^{15}}{15!} = 1 + \sum_{i=1}^{10} (-1)^n \frac{x^{2i-1}}{(2i-1)!},$$

для вычислений используйте рекуррентную формулу

$$z_n = z_{n-1} \frac{-x^2}{(2n-2)(2n-1)}.$$

Замечание: за начальные значения для рекуррентной формулы принять: $z_0 = -x, z = 1$.

49. Вычислить число сочетаний C_n^m из n элементов по m :

$$C_n^m = \frac{n!}{m!(n-m)!} = \prod_{i=0}^{n-m} \frac{m+i}{i}.$$

50. Вычислить размещение A_n^m из n элементов по m :

$$A_n^m = n(n-1)(n-2) \dots (n-(m-1)).$$

51. Вычислить сумму ряда

$$\sum_{i=1}^n \frac{x^i}{i!}$$

для заданных пользователем натурального числа n и действительного числа x .

52. Вычислить сумму ряда

$$\sum_{i=1}^n \frac{x + \cos(ix)}{2^i}$$

для заданных пользователем натурального числа n и действительного числа x .

53. Вычислить сумму ряда

$$\sum_{i=1}^n \left(\frac{1}{i!} + \sqrt{|x|} \right)$$

для заданных пользователем натурального числа n и действительного числа x .

54. Вычислить произведение

$$\prod_{i=1}^n \left(\frac{i}{i+1} - \cos^i(|x|) \right)$$

для заданных пользователем натурального числа n и действительного числа x .

55. Вычислить произведение

$$\prod_{i=1}^n \left(1 + \frac{\sin(ix)}{i!} \right)$$

для заданных пользователем натурального числа n и действительного числа x .

56. Вычислить произведение

$$\prod_{i=1}^n \left(\frac{(1-x)^{i+1} + 1}{((i-1)! + 1)^2} \right)$$

для заданных пользователем натурального числа n и действительного числа x .

57. Напишите программу, которая рассчитывает произведение двух комплексных чисел

$$z_1 = x_1 + iy_1, \quad z_2 = x_2 + iy_2.$$

58. Напишите программу, которая рассчитывает частное двух комплексных чисел $z_1 = x_1 +$

$$iy_1, \quad z_2 = x_2 + iy_2.$$

59. Напишите программу, которая рассчитывает сумму (разность) двух комплексных чисел

$$z_1 = x_1 + iy_1, \quad z_2 = x_2 + iy_2.$$

60. Напишите программу, которая рассчитывает аргумент комплексного числа $z_1 = x_1 + iy_1$.

61. Напишите программу, которая рассчитывает модуль комплексного числа $z_1 = x_1 + iy_1$.

Приложение Б

Математические функции

Таблица 8 – Основные математические функции математической библиотеки
(заголовочный файл math.h)

<i>Функция</i>	<i>Описание</i>	<i>Аргументы</i>
<code>abs(x)</code>	Модуль числа x : $ x $. Данная функция имеет целый тип.	x – целое число (тип <code>int</code>).
<code>fabs(x)</code>	Модуль числа x : $ x $.	x – вещественное число.
<code>ceil(x)</code>	Округление вверх до ближайшего целого $[x]$.	x – вещественное число.
<code>floor(x)</code>	Округление вниз до ближайшего целого $[x]$.	x – вещественное число.
<code>cos(x)</code>	Косинус.	x – вещественное число. Значение x вводится в радианах.
<code>sin(x)</code>	Синус.	x – вещественное число. Значение x вводится в радианах.
<code>tan(x)</code>	Тангенс.	x – вещественное число. Значение x вводится в радианах.
<code>acos(x)</code>	Арккосинус. Значение угла рассчитывается в радианах.	x – вещественное число.
<code>asin(x)</code>	Арсинус. Значение угла рассчитывается в радианах.	x – вещественное число.
<code>atan(x)</code>	Арктангенс. Значение угла рассчитывается в радианах.	x – вещественное число.
<code>log(x)</code>	Натуральный логарифм $\ln(x)$.	x – вещественное число.
<code>log10(x)</code>	Десятичный логарифм $\lg(x)$.	x – вещественное число.
<code>pow(x, y)</code>	Возведение в степень x^y .	x, y – вещественные числа.
<code>exp(x)</code>	Экспонента e^x .	x – вещественное число.
<code>sqrt(x)</code>	Квадратный корень \sqrt{x} .	x – вещественное число.

Список литературы

1. **Harbison III S. P. and Steele Jr. G. R.** C A reference manual [Book]. - New Jersey : Prentice Hall, 2002. - p. 552.
2. **Holub A.I.** Enough rope to shoot yourself in the foot: Rools for C and C++ programming [Book]. - [s.l.] : McGraw-Hill, 1995. - p. 186.
3. **Kernighan B. W. and Ritchie D. M.** The C programming language [Book]. - New Jersey : Prentice Hall PTR, Englewood Cliffs, 1988. - Second Edition : p. 282.
4. **Plauger P. J.** The standard C library [Book]. - New Jersey : Prentice Hall PTR, 1991. - p. 512.
5. **Shaw Z. A.** Learn C The Hard Way: A Clear and Direct Introduction To Modern C Programming [Книга]. - Indianapolis : Addison-Wesley Professional, 2015. - 1 edition : стр. 384.
6. **Бартенев О. В.** Современный Fortran [Book]. - Москва : ДИАЛОГ МИФИ, 2000. - p. 448.
7. **Павловская Т.А. Щупак Ю.А.** C/C++ Программирование на языке высокого уровня. Структурное программирование. Практикум [Book]. - СПб : Питер, 2002. - p. 240.
8. **Подбельский В. В. and Фомин С. С.** Программирование на языке Си [Book]. - Москва : Финансы и статистика, 2004. - p. 600.
9. **Прата С.** Язык программирования C (C11). Лекции и упражнения [Книга]. - М. : Вильямс, 2015. - стр. 928.
10. **Хэзфилд Р. Кирби Л., и др.** Искусство программирования на C. Фундаментальные алгоритмы, структуры данных и примеры приложений. Энциклопедия программиста: Пер. с англ [Book]. - Киев : Издательство "Диасофт", 2001. - p. 736.
11. **Шилдт Г.** Полный справочник по C [Book]. - Москва : Вильямс, 2005. - 4-е издание : p. 704.

Содержание

Введение	3
Лабораторная работа №1 Введение в программирование на языках высокого уровня..	4
Лабораторная работа №2 Алгоритмы. Графическое представление алгоритмов.....	7
Лабораторная работа №3 Знакомство с языком Си. Компиляция программ.....	13
Лабораторная работа №4 Переменные. Типы данных. Основные операции.....	18
Лабораторная работа №5 Условные операторы.....	24
Лабораторная работа №6 Циклы.....	31
Приложение А Варианты заданий	37
Приложение Б Математические функции	45
Список литературы.....	46