Учреждение образования
«Белорусский государственный университет культуры и искусств»

Факультет культурологии и социокультурной деятельности
Кафедра информационных технологий в культуре

СОГЛАСОВАНО
Заведующий кафедрой

_____ Т.С. Жилинская
«___»____-_____ 2021 г.

СОГЛАСОВАНО
Декан факультета

_____ Н.Е. Шелупенко
«___»_____ 2021 г.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС
ПО УЧЕБНОЙ ДИСЦИПЛИНЕ

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ
Раздел 4. АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ

для специальности 1–21 04 01 Культурология,
направление специальности 1–21 04 01–02 Культурология (прикладная),
специализации 1–21 04 01–02 04 Информационные системы в культуре

Составитель:
П.В. Гляков, профессор кафедры
информационных технологий в культуре

Рассмотрен и утвержден
на заседании Совета университета 18.05.2021
протокол № 9

Составитель:

Гляков Петр Владимирович, профессоркафедры информационных технологий в культуре учреждения образования «Белорусский государственный университет культуры и искусств», кандидат физико-математических наук, доцент

Рецензенты:

В.В. Казаченок, заведующий кафедрой компьютерных технологий и системБелорусского государственного университета, докторпедагогическихнаук, профессор;

В.С. Якимович, доцент кафедры высшей математики учреждения образования «Белорусский национальный технический университет»,кандидат педагогических наук

Рассмотрен и рекомендован к утверждению:
Кафедрой информационных технологий в культуре
(протокол от .03.2021 г.,№);

Советом факультета культурологии и социокультурной деятельности
(протокол от .03.2021 г., №)

Оглавление

1 ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Дисциплина «Алгоритмы обработки данных» является составной частью интегрированной дисциплины «Теоретические основы информационных технологий», в которую кроме нее входит дисциплины «Прикладная математика» и «Системный анализ и моделирование информационных процессов». В соответствии с учебными планами она предназначена для изучения студентами высших учебных заведений по специальности 1-2104 01 Культурология (по направлениям) направления специальности 1-21 04 01-02 Культурология (прикладная)специализации 1-21 04 01-02 04 Информационные системы в культуре.

Цель изучения дисциплины «Алгоритмы обработки данных» – формирование знаний и умений в области алгоритмизации, технологий проектирования алгоритмов, методов построения и разработки программ.

Основными *задачами* дисциплины являются:

– знакомство с основами теории сложности алгоритмов;

– изучение технологий проектирования и разработки программ;

– изучение языков программирования JavaScriptи VisualBasic;

– приобретение умений разрабатывать алгоритмы и программы.

В результате изучения дисциплины студенты должны *знать*:

– способы описания и представления алгоритмов;

– этапы подготовки и выполнения программ на компьютере;

– основные структуры данных;

– технологии проектирования алгоритмов;

– эффективные средства языков программирования JavaScriptи VisualBasicдля написания программ.

Студенты должны *уметь*:

– использовать современные подходы к проектированию и программированию.

– использовать средства автоматизации для разработки программ;

– разрабатывать стандартные алгоритмы и программы.

Методы обучения. Учебный материал излагается на основе современных методических требований с учетом педагогических целей на уровнях представления, понимания, знания, применения и творчества. При чтении лекций особое внимание уделяется рассмотрениюпримеров, иллюстрирующих то или иное понятие, приводятся различные способы интерпретации понятий.

При обучении применяется такой метод, когда сначала выделяются классы задач и внутри этих классов задач рассматриваются типичные методы их решения. При этом сначала строятся схемы решения таких классов задач на

основе классических структур управления, а потом обсуждается их программирование, когда свойства объявляются при помощи разнообразных структур данных.

Обучение ведется на основе:

– выделения элементарных операций при построении типичныхалгоритмов обработки простых данных, структурированных статичных идинамических данных;

– одинаковой формы записи алгоритма для решения задач содинаковой структурой исходных данных;

– выделения вспомогательных алгоритмов, которые потом оформляютсяподпрограммами языка и могут объединяться в модули.

Лабораторные занятия направлены на формирование умений практического использования полученных знаний при разработке алгоритмов и программ для решения конкретных задач. Методика их проведения содействует развитию творческих способностей каждого студента и приобретению навыков самостоятельной работы. Используются такие новые формы активизации учебного процесса, как игры, викторины и т.п.

Самостоятельная работа студентов ориентирована на изучение отдельных вспомогательных тем дисциплины, решение дополнительных рекомендованных задач и подбор практических примеров, иллюстрирующих теоретические основы алгоритмизации и программирования. Результаты самостоятельной работы выявляются как при ответах на теоретические вопросы, так и при разработке алгоритмов и программ для решения задач.

Учебным планом на изучение дисциплины «Алгоритмы обработки данных» предусмотрено 112 часов всего, из них 44 часа – аудиторные занятия. Примерное распределение аудиторных часов по видам занятий: лекции – 14 часов, лабораторные занятия –18 часов, семинарские и практические занятия – 12 часов.

Дисциплина рассчитана на один семестр. Текущий контроль осуществляется при выполнении и сдаче лабораторных работ.Форма контроля – экзамен.

2 ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

2.1 Тематика лекционных занятий

Раздел 1. Основы алгоритмизации
Тема 1. Понятие алгоритма и способы представления.
Лекция 1.Понятие алгоритма и способы представления.
Тема 4. Методы обработки информации.
Лекция 2. Методы обработки информации.
Тема 5. Введение в теорию сложности.
Лекция 3. Введение в теорию сложности.
Тема 7.Алгоритмы в криптологии.
Лекция 4.Алгоритмы в криптологии.
Тема 8. Алгоритмы в базах данных.
Лекция 5. Алгоритмы в базах данных.
Раздел 2. Язык программирования JavaScript
Тема 9. Основы языка JavaScript.
Лекция 6.Основы языка JavaScript.
Раздел 3. Язык программирования VisualBasic
Тема 12. Работа в средеVisualBasic.
Лекция 7. Работа в среде VisualBasic.

2.2 Конспект лекций

Лекция 1

Понятие алгоритма и способы представления

Основные вопросы

1. Понятие алгоритма.
2. Эмпирические свойства алгоритма.
3. Способы представления и разработки алгоритмов.
4. Типы структур в алгоритмах.
5. Основная теорема структурного программирования.

*Цель.*Ознакомление с понятием алгоритма и его свойствами. Изучение способов представления алгоритмов, алгоритмических структур и основной теоремы структурного программирования.

Понятие алгоритма

Важным этапом при решении задачи на компьютере является этап разработки алгоритма. В нашем курсе мы будем знакомиться только с понятием алгоритма, потому чтосуществующее точное математическое определение алгоритма требует глубоких математических знаний в области дискретной математики. Начнем с формулировки понятия алгоритма, которую дал академик А.П. Ершов в учебнике «Основы информатики и вычислительной технике: «Под алгоритмом понимают понятное и точное предписание (указание) исполнителю осуществить последовательность действий, направленных на достижение указанной цели или на решение предлагаемой задачи.

Приведем еще две формулировки понятия алгоритма, которые тоже будут рабочими в нашем курсе:

1. «Алгоритм – это точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату».

2. «Под алгоритмом будем понимать конечную совокупность указаний, руководствуясь которыми можно выполнить любое сложное задание или решить задачу».

Свойства алгоритма

Алгоритм должен обладать следующими основными свойствами:

1. Понятность– исполнитель алгоритма должен понимать, как его выполнять. Иными словами, имея алгоритм и произвольный вариант исходных данных, исполнитель должен знать, как надо действовать для выполнения этого алгоритма.

2. Детерминированность (определенность) –каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвола.

Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче. При заданных исходных данных обеспечивается однозначность искомого результата;

3. Дискретность (прерывность, раздельность) – алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов (этапов, действий).

4. Результативность (или конечность) состоит в том, что за конечное число шагов алгоритм либо должен приводить к решению задачи, либо после конечного числа шагов останавливаться из-за невозможности получить решение с выдачей соответствующего сообщения, либо продолжаться в течение времени, отведенного для исполнения алгоритма, с выдачей промежуточных результатов.

5. Массовость означает, что алгоритм решения задачи разрабатывается в общем виде, т.е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется областью применимости алгоритма.

Формы представления алгоритмов

Одним из самых трудоемких этапов решения задачи на компьютере является разработка алгоритма. Человечество разработало эффективный алгоритм завязывания шнурков на ботинках. Многие дети с пятилетнего возраста могут это делать. Но дать чисто словесное описание этого алгоритма без картинок и демонстрации – очень трудно.

При разработке алгоритмов чаще всего используют следующие способы их описания: словесный, графический и с помощью алгоритмического языка (псевдокода).

На практике наиболее распространены следующие формы представления алгоритмов:

- словесная(запись на естественном языке);
- графическая(изображения из графических символов);
- псевдокоды (полуформализованные описания алгоритмов на условном алгоритмическом языке, включающие в себя как элементы языка программирования, так и фразы естественного языка, общепринятые математические обозначения и др. Русскоязычным представителем псевдокодов является алгоритмический язык);
- программная(тексты на языках программирования).

Словесный способ записи алгоритмов

Словесный способзаписи алгоритмов представляет собой описание последовательных этапов обработки данных. Алгоритм задается в произвольном изложении на естественном языке.

Пример.Напишите алгоритм нахождениянаибольшего общего делителя (НОД)двух натуральных чисел (алгоритм Эвклида).

Алгоритм может быть следующим:

1. Задать два числа;
2. Если числа равны, то взять любое из них в качестве ответа и остановиться, в противном случае продолжить выполнение алгоритма;
3. Определить большее из чисел;
4. Заменить большее из чисел разностью большего и меньшего из чисел;
5. Повторить алгоритм с шага 2.

Описанный алгоритм применим к любым натуральным числам и должен приводить к решению поставленной задачи. Убедитесь в этом самостоятельно, определив с помощью этого алгоритма наибольший общий делитель чисел 102 и 24.

Словесныеописания алгоритмов не имеет широкого распространения, так как они обладают следующими существенными недостатками:

- строго не формализуемы;
- страдают многословностью записей;
- допускают неоднозначность толкования отдельных предписаний.

Графический способ записи алгоритмов

Графический способ представления алгоритмов является более компактным и наглядным по сравнению со словесным.

При графическом представлении алгоритм изображается в виде последовательностисвязанных между собой функциональных блоков, каждый из которых соответствуетвыполнению одного или нескольких действий.

Такое графическое представление называется схемой алгоритма илиблок-схемой. В блок-схеме каждому типу действий (вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий, окончанию обработки и т.п.) соответствует геометрическая фигура, представленная в видеблочного символа. Блочные символы соединяютсялиниями переходов, определяющими очередность выполнения действий.

Графический способ записи алгоритмов является наиболее наглядным и распространенным. Он основан на использовании геометрических фигур(блоков), каждая из которых отображает конкретный этап процесса обработки данных, соединяемых между собой прямыми линиями, называемыми линиями потока.

Псевдокоды

Псевдокод представляет собой систему обозначений и правил, предназначенную для единообразной записи алгоритмов.

Псевдокод занимает промежуточное место между естественным и формальным языками. С одной стороны, он близок к обычному естественному языку, поэтому алгоритмы могут на нем записываться и читаться как обычный текст. С другой строны, в псевдокоде используются некоторые формальные конструкции и математическая символика, что приближает запись алгоритма к общепринятой математической записи.

В псевдокоде не приняты строгие синтаксические правила для записи команд, присущие формальным языкам, что облегчает запись алгоритма на стадии его проектирования и дает возможность использовать более широкий набор команд, рассчитанный на абстрактного исполнителя.

Однако в псевдокоде обычно имеются некоторые конструкции, присущие формальным языкам, что облегчает переход от записи на псевдокоде к записи алгоритма на формальном языке. В частности, в псевдокоде, так же, как и в формальных языках, есть служебные слова, смысл которых определен раз и навсегда. Они выделяются в печатном тексте жирным шрифтом, а в рукописном тексте подчеркиваются.

Единого или формального определения псевдокода не существует, поэтому возможны различные псевдокоды, отличающиеся набором служебных слов и основных (базовых) конструкций.

Примером псевдокода является школьный алгоритмический язык в русской нотации, описанный в учебнике А.Г. Кушниренко и др. "Основы информатики и вычислительной техники", 1991. Этот язык в дальнейшем мы будем называть просто "алгоритмический язык".

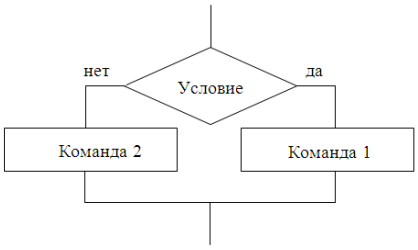
Основные типы структур в структурном программировании

В структурном программировании для записи алгоритмов используют три базовых структуры: следование, разветвление и цикл. Представим эти структуры в виде блок-схем.

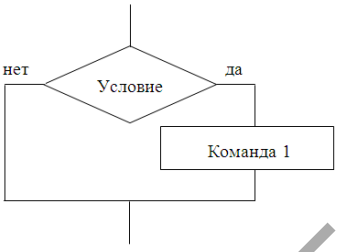
Характерной особенностью базовых структур является наличие в них одного входа и одного выхода. Стрелка сверху базовой структуры указывает на вход, а стрелка снизу – на выход.

1. Базовая структура "следование".Образуется последовательностью действий, следующих одно за другим.Она отображает естественный (сверху-вниз) порядок выполнения действий». Поэтому нет необходимости указывать в ней стрелки.

2. Базовая структура “разветвление”. В этой алгоритмической конструкции в зависимости от условия выполняется та или иная последовательность действий. Когда условие является истинным, выполняется команда 1, в противном случае выполняется команда 2.

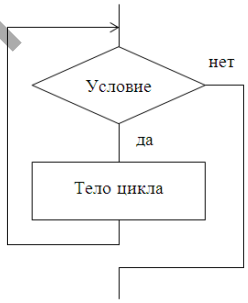


В сокращенной форме эта конструкция может быть представлена следующим образом.



Команда 1 выполняется тогда, когда условие является истинным.Если при выполнении или невыполнении условия требуется выполнить не одно, анесколько действий, то следует применить составной оператор.

3. Базовая структура «цикл».Она используется для представления последовательности действий, выполняемых многократно. Последовательность действий, повторяющаяся в процессе выполнения цикла, называется телом цикла.



При выполнении команды цикла сначала проверяется условие.Если условие истинно, то выполняются действия тела цикла и снова осуществляетсяпереход к проверке условия. Если условие окажется ложным, то осуществится выход из структуры цикла.

Основная теорема структурного программирования

Основная теорема структурного программирования утверждает, что логическая структура любого алгоритма может быть представлена комбинацией трех базовых структур: следования, разветвления и цикла.

Лекция 2

Методы обработки информации

Основные вопросы

1. Общий вид алгоритма на алгоритмическом языке.
2. Запись списка параметров.
3. Описание типов данных.
4. Запись строк с аргументами и результатами.
5. Описание вспомогательных переменных.
6. Оператор присваивания.
7. Оператор разветвления.
8. Оператор цикла с предусловием.
9. Линейные, разветвляющиеся и циклические алгоритмы.
10. Исполнение алгоритма.
11. Правильность алгоритмов.
12. Подбор тестов для проверки алгоритма.
13. Тестирования и отладка алгоритмов.

Цель. Изучение способа записи алгоритма на алгоритмическом языке, описания параметров, вспомогательных переменных. Изучение основных типов алгоритмов, формирование умений исполнять алгоритмы, выполнять тестирование и отладку алгоритмов, разрабатывать алгоритмы с использованием разветвлений и циклов.

Общий вид алгоритма на алгоритмическом языке

Алгоритмический язык– формальный язык, используемый для записи, реализации или изученияалгоритмов.В отличие от большинства языков программирования он не привязан к архитектурекомпьютера, не содержит деталей, связанных с устройством машины. Алголоподобный алгоритмический язык с русским синтаксисом был введён в употребление академикомА.П. Ершовымв середине 1980-х годов в качестве основы для «безмашинного» курса информатики. Впервые он был опубликован в учебнике «Основы информатики и вычислительной техники» в 1985 г.

Алгоритм на русском алгоритмическом языке в общем виде записывается следующим образом:

```
алг название алгоритма (список параметров с описанием типа)
аргсписок аргументов
резсписок результатов
начсписок вспомогательных переменных с описанием типа
последовательность команд
кон
```

Часть алгоритма от словаалгдо слованачназываетсязаголовком, а часть, заключенная между словаминачиикон, – телом алгоритма.

В предложенииалгпосле названия алгоритма в круглых скобках перечисляются параметры алгоритма, являющиеся аргументами и результатами, и тип значения (цел,вещ,сим,литилилог) всех входных (аргументы) и выходных (результаты) переменных. При описании массивов (таблиц) используется служебное словотаб, дополненное граничными парами по каждому индексу элементов массива.

В записи алгоритма ключевые слова обычно подчёркиваются либо выделяются полужирным шрифтом. Для выделения логических блоков применяются отступы, а парные слова начала и конца находятся на одной вертикали.

Рассмотрим на примере вычисления функции n! (n!=1*2*3* … *n), как выглядит алгоритм на алгоритмическом языке. В этом примере предполагается, что n является натуральным числом.

```
алгФакториал (натn, вещf)
аргn
резf
начцелi
f:=1, i:=2
нцпока i ≤ n
f:=f*I, i:=i+1
кц
кон
```

В таблице приведены основные служебные слова алгоритмического языка.

Описаниеалгоритма	Типы данных	Обозначениеусловий	Обозначение цикла	Логические функции
алг– алгоритм	цел – целый	если	нц – начало цикла	и
арг –аргумент	вещ – вещественный	то	кц– конец цикла	или
рез – результат	сим – символьный	иначе	пока	не
нач – началоалгоритма	лит – строка	все	для	
кон– конец алгоритма	лог – логический	выбор	от	
	таб – массив	при	до	
	длин – длина		шаг	

Основные команды алгоритмического языка

Команда присваивания служит для вычисления выражений и присваивания их значений переменным. Общий вид команды присваивания:

A := B,

где знак ":=" означает команду, которая будетменять прежнее значение переменной, стоящей в левой части,на вычисленное значение выражения, стоящего в правой части.

Пример 1. Разработать алгоритм, который по заданному радиусу rбудет находить длину окружности С.

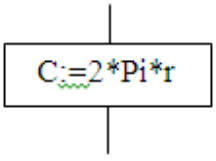
На алгоритмическом языке алгоритм будет выглядеть следующим образом

```
алгДлина окружности(вещr, С)
   аргr
   резС
нач
   С:=2
   С:=С*Pi
   С:=С*r
кон
```

В этом примере мы специально подробно расписали все действия для вычисления длины окружности. Далее мы это делать не будем. Мы должны понимать, что алгоритм – это не программа, а разработка алгоритма – это лишь один из этапов решения задачи на компьютере (хотя и очень важный). Поэтому алгоритм решения этой задачи можно было сразу записать так

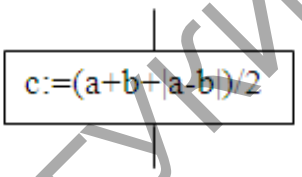
```
алг Длина окружности(вещr, С)
   аргr
   резС
нач
   С:=2*Pi*r
кон
```

Блок-схема данного алгоритма выглядит следующим образом



Пример 2. Разработать алгоритм, который найдет большее значение переменных аиb и присвоит это значение переменной с.

Внимательно рассмотрите приведенный ниже алгоритм и выясните, будет ли он решать поставленную задачу. Использованная в нем запись |a-b| означает модуль числа a-b.

Алгоритмический язык	Блок-схема
алгМаксимум(веща,b,c) арга,b резс нач c:=(a+b+|a-b|)/2 кон	

Оператор разветвления

Оператор разветвления если в алгоритмическом языке имеет две формы: полную и сокращенную.

Полная форма	Краткая форма
если условие то операторы 1 иначе операторы 2 все	если условие то операторы все

При выполнении оператора разветвления еслив полной форме сначала проверяется условие. Если оно истинно, то выполняются операторы 1 и завершается выполнение оператора если. В противном случае выполняются операторы 2и на этом прекращается выполнениеоператора если.

В случае сокращенной формы оператора еслиоператоры, после служебного слова то, выполняются только тогда, еслиусловие истинно.

Пример. Разработать алгоритм, который найдет большее значение трех переменных a,b и си присвоит это значение переменной d.

Для решения этой задачи рассмотрим два способа. Один способ содержит вложенные друг в друга операторы если; другой – использует ранее полученные знания для нахождения максимума из двух переменных.

1 способ	2 способ
алгМаксимум 1(веща,b,c,d) арга,b,c резd нач	алгМаксимум 2(веща,b,c,d) арга,b,c резd нач

```
   еслиа>b
      тоеслиа>c
            тоd:=a
            иначеd:=c
         все
      иначееслиb>c
               тоd:=b
               иначеd:=c
            все
   все
кон
```

```
   еслиа>b
      тоd:=a
      иначеd:=b
   все
   еслис>d
      тоd:=c
   все
кон
```

Выясните преимущества и недостатки каждого способа.

Оператор цикла с предусловием

Оператор цикла с предусловиемиспользуется, когда число повторений операторов тела цикла заранее неизвестно. В алгоритмическом языке он имеет следующий вид:

```
нцпокаусловие
     операторы
кц
```

При выполнении оператора цикла с предусловиемвначале проверяется условие. Если условие является истинным, то выполняются операторы тела цикла и передача управления происходит на проверку условия. Так будет происходить до тех пор, пока условие не станет ложным. Когда условие станет ложным, завершится выполнение оператора цикла.

Пример. Разработать алгоритм, который будет находить n-е число Фибоначчи. Числа Фибоначчи определяются следующим образом:

$F_1=0$, $F_2=1$, $F_n=F_{n-1}+F_{n-2}$для n>2, где n – натуральное число.

```
алг Число Фибоначчи(целn,F)
    аргn
    резF
начцелF1,F2,i
    еслиn=1
        тоF:=0
        иначееслиn=2
                тоF:=1
                иначеF1:=0; F2:=1; i:=2
                   нцпокаi<n
                         F:=F1+F2; F1:=F2; F2:=F; i:=i+1
                   кц
            все
```

```
    все
кон
```

В этом алгоритме для организации цикла использован оператор цикла с предусловием пока. Если аргумент n равен 3, то условие циклаi<nобеспечивает однократное выполнение операторов тела циклаF:=F1+F2; F1:=F2; F2:=F; i:=i+1. При их выполнении переменнаяiстанетравной3, что сделаетусловие цикла ложным. Поэтому завершится выполнение цикла и самого алгоритма.На выходе из цикла мы получим значение переменнойF, равное 2, что на самом деле соответствует истине: третье число Фибоначчи равно 2. Самостоятельно исполните алгоритм приn, равном 5.

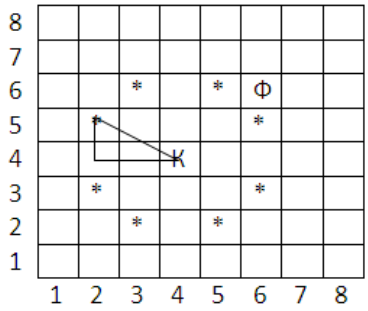
В соответствии с тремя базовыми структурами раличают три типа алгоритмов: линейные, разветвляющиеся и циклические. При выполнении линейного алгоритма исполнитель выполняет одну команду за другой в порядке их следования. В разветвляющихся алгоритмах действия исполнителя определяются результатами проверки некоторых условий. Структура раветвления может быть в полной и неполной форме. При исполнении циклического алгоритма отдельные команды или группы команд повторяются многократно.

При решении задач на компьютере, которые требуют только последовательные действия, отсутствует необходимость использовать этап алгоритмизации. Для их решения сразу записывают команды на языках программирования. Поэтому мы сразу перейдем к рассмотрению алгоритмов с разветвлениями, циклами и их комбинациями.

Алгоритмы с разветвлениями

Пример 1. На шахматной доске дан конь с координатами (x_k;$y_{k)}$и фигура с координатами ($x_ф$;$y_{ф)}$. Разработать алгоритм, который определяет, находится ли фигура под боем коня. Если фигура находится под боем коня, то переменной с присвоить значение «бьет», в противном случае присвоить значение «не бьет».

Представим шахматную доску как координатную плоскость. Вместо буквенной нижней нумерации будем использовать цифровую. Обозначим коня буквой k, а фигуру – буквой. Отметим на координатной плоскости звездочкой * те клетки, которые находятся под боем коня.



Если внимательно посмотреть на полученный рисунок, то можно заметить, что все клетки со звездочками находятся на одинаковом расстоянии от коня. Доказать это можно следующим образом. Соединим клетки с координатами (2;5) и (4;4), (2;4) и (4;4) и, наконец, (2;5) и (2,4). Мы получили прямоугольный треугольник, у которого длина одного катета равна 1, а другого равна 2. Построим таких же еще 7 треугольников, у которых гипотенузасоединяет клетку со звездочкой и клетку с конем. Полученные треугольники будут равны, так как у них равны две стороны и угол. Поэтому гипотенузы у них также будут равны. Геометрическое место точек, равноудаленных от одной точки, называемой центром, является окружностью. В нашем случае центром окружности является клетка с конем, а клетки со звездочками – точками окружности. Вычислим радиус окружности.Он определяется следующим образом:

$$r = \sqrt{1^2 + 2^2} = \sqrt{5}$$

Таким образом, чтобы фигура былапод боем коня, она должна стоять на клетке, находящейся на расстоянии $\sqrt{5}$ от клетки коня.

```
алгБой конем(целXk,Yk,Xf,Yf, лит с)
   аргXk,Yk,Xf,Yf
   резс
нач
   если(Xk-Xf)*(Xk-Xf)+(Yk-Yf)*(Yk-Yf)=5
      тос:='бьет'
      иначес:='не бьет'
   все
кон
```


Правильность алгоритмов

После построения алгоритма решения задачи, точнее, класса однотипных задач, всегда возникают вопросы: правильно ли построен алгоритм, верно ли решает он задачу. Для их решения следует проверить правильность алгоритма. Естественная и наиболее простая проверка – тестирование, которое и применяется на практике. Суть тестирования заключается в том, что, имея некоторую совокупность правильно решенных задач, решают их затем с

помощью алгоритма и убеждаются в его истинности. Если имеются расхождения, то алгоритм соответственно исправляют.

Лекция 3

Введение в теорию сложности алгоритмов

Основные вопросы

1. Размерность задачи.
2. Характеристики времени и памяти алгоритмов.
3. Вычислительная машина.
4. Единицы измерения времени и памяти алгоритма.
5. Оценка алгоритма в лучшем, среднем и худшем случаях.

*Цель.*Изучение характеристик времени и памяти алгоритмов. Формирование умений оценивать алгоритм в лучшем, среднем и худшем случаях.

Размерность задачи

Вычислительная сложность– понятие винформатикеитеории алгоритмов, обозначающее функцию зависимости объёма работы, которая выполняется некоторым алгоритмом, от размера входных данных. Раздел, изучающий вычислительную сложность, называетсятеорией сложности вычислений.

Объём работы обычно измеряетсяпонятиями времени и пространства, называемыми вычислительными ресурсами. Время определяется количеством элементарных шагов, необходимых для решения задачи, тогда как пространство определяется объёмом памяти или места наносителе данных. Таким образом, в этой области предпринимается попытка ответить на центральный вопрос разработки алгоритмов: «как изменится время исполнения и объём занятой памяти в зависимости от размера входа?». Здесь под размером входа понимается длина описания данных задачи вбитах(например, взадаче поиска элемента в массиве, содержащего n элементов, длина входа почти пропорциональна количеству элементов). Поэтому в качестве размера этой задачи часто берут величину n).

Стеоретической информатикойтесно связаны такие области как алгоритмический анализитеория вычислимости. Связующим звеном между теоретической информатикой и алгоритмическим анализом является тот факт, что их формирование посвящено анализу необходимого количества ресурсов определённых алгоритмов решения задач, тогда как более общим вопросом является возможность использования алгоритмов для подобных задач. Конкретизируясь, попытаемся классифицировать проблемы, которые могут или не могут быть решены при помощи ограниченных ресурсов. Отличие теории вычислительной сложности от вычислительной теории заключается в сильном ограничении доступных ресурсов. Вычислительная теория отвечает на вопрос, какие задачи, в принципе, могут быть решены алгоритмически.

Теория сложности вычислений возникла из потребности сравнивать быстродействие алгоритмов, чётко описывать их поведение (время исполнения и объём необходимой памяти) в зависимости от размера входа.

Количество элементарных операций, затраченных алгоритмом для решения конкретного экземпляра задачи, зависит не только от размера входных данных, но и от самих данных. Например, количество операций алгоритма сортировки методом пузырькаказначительно меньше в случае, если входные данные уже отсортированы. Чтобы избежать подобных трудностей, рассматривают понятие временной сложности алгоритмав худшем случае.

Временная сложность алгоритма(в худшем случае) – это функция от размера входных данных, равная максимальному количеству элементарных операций, проделываемых алгоритмом для решения экземпляра задачи указанного размера.

Аналогично понятиювременной сложности в худшем случаеопределяется понятиевременная сложность алгоритма в наилучшем случае. Также рассматривают понятиесреднее время работы алгоритма, то есть математическое ожиданиевремени работы алгоритма. Иногда говорят просто: «Временная сложность алгоритма» или «Время работы алгоритма», имея в виду временную сложность алгоритма в худшем, наилучшем или среднем случае (в зависимости от контекста).

По аналогии с временной сложностью, определяютпространственную сложность алгоритма, только здесь говорят не о количестве элементарных операций, а об объёме используемой памяти.

Вычислительная машина

Для того чтобы приступить к вычислению характеристик времени и памяти алгоритма, надо выбрать виртуальную вычислительнуюмашину или устройство, которое будет понятно исполнителю. В нашем случае проще всего в качестве такого устройства выбрать алгоритмический язык.

Единицы измерения времени и памяти алгоритма

За единицу времени мы будем принимать время, требуемое для выполнения одного оператора присваивания и время выполнения проверки условия. За единицу памяти возьмем память, занимаемую одной переменной или одним элементом массива. Для некоторых задач можно взять память, занимаемую одним символом. Несмотря на кажущуюся грубость выбранных нами оценок, они позволяют сравнивать между собой алгоритмы решения задачи и выбирать лучший.

Оценка алгоритма

Для оценки сформулируем условие задачи, допускающей достаточную вариативность решения. Удобно для этой цели взять задачу последовательного поиска.

Условие. Дан массив целых чисел a размерности n. Требуется определить, есть ли в нем элемент a_i, равный некоторому целому числу b. Если такой элемент есть, то строковой переменной c присвоить значение “да”, в противном случае присвоить значение “нет”.

Обычно алгоритмы, которые представляют студенты, на алгоритмическом языке выглядят следующим образом. С правой стороны мы указываем, сколько времени в худшем случае затрачивают операторы в указанных строках.

Алгоритм Поиск1

```
алг Поиск1(целтаб a[1:n], целn,b, литc)
    арг a, n, b
    рез c
начцелi
    i:=1, c:= “нет”2
    нцпокаi≤ nn+1
        если a[i]=bn
            то c:= “да”, i:=n+10
            иначе i:=i+1n
        все
    кц
кон
```

Идея алгоритма Поиск1 заключается в том, что последовательно просматриваются элементы массива a и сравниваются с величиной b. Как только будет обнаружен элемент, равный b, цикл завершает свою работу, поскольку переменной i, используемой для организации цикла, присваивается значениеi:=n+1, которое делает условие цикла ложным.

Алгоритм Поиск2

```
алг Поиск2(цел таб a[1:n], цел b, n, лит c)
     арг a, n, b
     рез c
нач цел i
     i:=1, c:= “нет”              2
     нцпока i≤ n и c:= “нет”   2(n+1)
          если a[i]=b             n
          то c:= “да”             0
          иначе i:=i+1            n
       все
     кц
кон
```

В алгоритме Поиск2 условие цикла является составным. В этом условии проверяется не только переменная i, которая используется для последовательного просмотра элементов массива a, но и переменная c, которой предварительно установлено значение “нет”. Как только будет найден искомый элемент, этой переменной будет присвоено значение “да” и цикл завершит свою работу.

Алгоритм Поиск3

```
алг Поиск3(цел таб a[1:n], нат n, цел b, лит c)
      арг a, n, b
      рез c
нач цел i
      i:=1, c:= “нет”              2
      пока i≤ nn+1
      нц
         если a[i]=b               n
            то c:= “да”            0
         все
         i:=i+1  n
      кц
кон
```

Алгоритм Поиск3 не завершает свою работу, как только найдет искомый элемент. Он будет осуществлять просмотр всех элементов массива a и каждый раз, когда будет встречаться элемент a_i, равный b, переменной c будет присваиваться значение “да”. Интуитивно становится понятным, что в среднем случае этот алгоритм будет работать дольше, чем алгоритмы Поиск1 и Поиск4.

Алгоритм Поиск4

```
      алг Поиск4(цел таб a[1:n], нат n, цел b, лит c)
         арг a, n, b
         рез c
      нач цел i
         i:=1                      1
         пока i≤ n                 n+1
         нц
            если a[i]=b            n
```

```
        тоi:=n+1          0
      все
      i:=i+1              n
   кц
   еслиi=n+2              1
      то c:= “да”         0
      иначе c:= “нет”     1
   все
кон
```

Цикл в алгоритме `Поиск4` завершает свою работу сразу же после обнаружения искомого элемента массива a. Для этой цели также, как и в алгоритме Поиск1, переменной i, используемой в условии цикла, присваивается значение, которое делает условие цикла ложным. Отличие от алгоритма Поиск1 состоит в том, что значение переменной c, в которой формируется результат выполнения алгоритма, присваивается после завершения работы цикла.

Характеристики алгоритмов

Вычислим характеристики времени T(n) и памяти M(n) для худшего случая, т. е. такого случая, когда алгоритм будет требовать при выполнении наибольшего количества времени. Такой случай соответствует ситуации, в которой массив a не содержит элемента a_i, равного числу b.

С правой стороны строк приведенных алгоритмов Поиск1, Поиск2, Поиск3, Поиск4 мы указали, сколько раз выполняется либо проверка условия цикла и разветвления, либо оператор присваивания. Команды это должны сделать сами. Просуммировав эти значения, мы получаем соответственно следующие характеристики времени в худшем случае для приведенных алгоритмов: $T_1(n)=3n+3$, $T_2(n)=4n+4$, $T_3(n)=3n+3$, $T_4(n)=3n+4$. Для всех четырех алгоритмов память, требуемая для выполнения алгоритма в худшем случае, будет одинаковой: $M(n)=n+4$. В эту величину входит память, требуемая для размещения исходных данных, результата и вспомогательной переменной **i**.

Из вычисленных характеристик времени выполнения алгоритма в худшем случае видно, что при достаточно большом размере задачи **n** алгоритм `Поиск2` будет выполняться дольше остальных в 4/3 раза:

$$\lim_{n \to \infty} \frac{4n+4}{3n+4} = \frac{4}{3}.$$

Как правило, командам не удается найти более быстрый алгоритм для решения задачи последовательного поиска. Поэтому преподаватель сам

знакомит студентов с идеей алгоритма для выполнения быстрого последовательного поиска (БПП) и сам записывает на доске алгоритм БПП, который приведен ниже.

Алгоритм БПП

```
алг БПП(цел таб a[1:n+1], нат n, цел b, лит c)
      арг a, n, b
      рез c
нач цел i
      i:=1,    a[1:n+1]:= b        2
      пока a[i]≠b                  n+1
      нц
        i:=i+1  n
      кц
      еслиi≤n  1
        то c:= “да”                0
        иначе c:= “нет”            1
      все
кон
```

Характеристики времени и памяти в худшем случае для этого алгоритма будут равны соответственно следующим величинам:

$$T_{БПП}(n)= 2n+5, \qquad M_{БПП}(n) = n+5.$$

При сравнении характеристики времени алгоритма БПП с характеристикой времени алгоритма Поиск1 (который является наиболее быстрымиз приведенных) при достаточно большом значении **n** получаем следующее:

$$\lim_{n \to \infty} \frac{3n+3}{2n+5} = \frac{3}{2}.$$

Это означает, что алгоритм БПП выполняется в 1,5 раза быстрее, чем алгоритм Поиск1, говоря другими словами, его использование вместо алгоритма Поиск1 позволяет экономить электроэнергию в 1,5 раза.

Лекция 4

Алгоритмы в криптологии

Основные вопросы

1. Обобщенный алгоритм электронной цифровой подписи.
2. Шифры простой подстановки.
3. Полиграммное шифрование.
4. Подстановочное шифрование.
5. Шифр Вернама.

Цель. Изучение основных алгоритмовшифрования информации.

Обобщенный алгоритм электронной цифровой подписи

Пусть A и B – некоторые пользователи, обменивающиеся информацией по открытому каналу связи. Пусть X – совокупность всевозможных сообщений, Y – некоторое множество "подписей". Пусть $F_k : Y \to X$ – функция, зависящая от параметра $k \in K$, называемого **ключом**. Будем считать, что ключ k состоит из двух частей: k_S и k_O, где k_S — секретная составляющая, известная только A, и k_O – открытая составляющая, известная "всем" (не держится в секрете). Пусть F_k является сюръекцией, т. е. для любого $x \in X$ существует прообраз $y = F_k^{-1}(x)$. Функцию F также считаем общеизвестной.

Предположим, что выполняются следующие свойства:

1) зная k_O, функцию $F_k(y)$ можно вычислить по алгоритму полиномиальной сложности;

2) зная k_S, функцию $F_k(y)$ можно инвертировать по алгоритму полиномиальной сложности;

3) зная k_O, но не зная k_S, функцию $F_k(y)$ сложно инвертировать, то есть не известен или не существует полиномиальный алгоритм нахождения $F_k^{-1}(x)$.

Прообраз $y = F_k^{-1}(x)$некоторого сообщения xназывается ***подписью этого сообщения***. Пара (x, y)называется ***подписанным сообщением.***

В силу первого свойства всегда легко проверить, соответствует ли подпись сообщению, а в силу третьего свойства подделать подпись при достаточно большом ключе практически невозможно. Доказательство этого свойства позволило бы придать подписанным сообщениям юридическую силу.

Секретный и открытый ключи находятся во взаимно однозначном соответствии и в силу третьего требования не существует полиномиального

алгоритма вычисления секретной компоненты по открытой компоненте. Таким образом, в общем виде алгоритм ЭЦП выглядит так.

1. Для передаваемого сообщения *x*отправитель *A* находит $y = F_k^{-1}(x)$. Знание секретного ключа k_Sпозволяет ему сделать это за приемлемое время.

2. Далее *A* передаёт *B* по какому-либо каналу связи пару (*x*,*y*), где *x* – сообщение, *y* – подпись.

3. Получив подписанное сообщение (*x*,*y*), *B* находит $x' = F_k(y)$. Знание открытого ключаk_Oпозволяет сделать это за приемлемое время.

4. Получатель *B* сверяет *x*и *x′*. Если они совпадают, то полученное сообщение считаем подлинным. В противном случае либо сообщение *x*изменено (фальшивое), либо подпись *y*неверная (поддельная).

Указанную модель можно дополнить предварительным шифрованием пересылаемого сообщения и итоговой расшифровкой. Роль функции F_k иногда выполняет некоторая схема шифрования с открытым ключом. В силу этого многие вопросы (стойкость, выбор ключей и др.) равносильны для схем ЭЦП и соответствующих криптосистем.

Шифры простой подстановки

Одним из самых старых шифров является шифр Юлия Цезаря. Алгоритм этого шифра состоит в следующем. Каждая буква латинского алфавита сдвигается циклически вправо на $k = 3$ позиций. Таким образом, имеем подстановку (замену):

A	B	C	D	E	F	G	H	…	T	U	V	W	X	Y	Z
D	E	F	G	H	I	J	K	…	W	X	Y	Z	B	B	C

Шифрование осуществляется в соответствии с этой подстановкой. Например, E_3(SUN)= VXQ. Понятно, что выбор $k = 3$ не является единственно возможным. При других ключах *k* имеем

$$E_{25}(\text{IBM}) = \text{HAL},\ E_6(\text{IDMUP}) = \text{OJSAV}.$$

Нетрудно показать, что $D_k = E_{26-k}$, $D_k E_k = E_0 = D_0$, а ключ *k* определен по модулю 26.

Шифр Цезаря является примером *шифра подстановки,* или *замены.* Его еще называют *шифром простой подстановки.* Криптоанализ такого шифра очень прост. Дело в том, что для любого современного языка вычислены частотные характеристики букв, т.е. относительные частоты их появления в «нормальных» текстах. Приведем эти частоты в процентах (с упорядочением) для английского языка.

Высокий		Средний		Низкий	
E	12.31	L	4.03	B	1.62
T	9.59	D	3.65	G	1.61
A	8.05	C	3.20	V	0.93
O	7.94	U	3.10	K	0.52
N	7.19	P	2.29	Q	0.20
I	7.18	F	2.28	X	0.20
S	6.59	M	2.25	J	0.10
R	6.03	W	2.03	Z	0.09
H	5.14	У	1.88		

Можно сказать, что при шифровании простой заменой буквы текста заменяются буквами этого или другого алфавита в соответствии с некоторой подстановкой.

Еще одним примером шифра простой замены является *модулярный шифр.* Выберем число *a,* взаимно простое с модулем $m = 26$. Пусть p – буква английского алфавита, отождествленная со своим порядковым номером (0, 1, ..., 25). Тогда $E_a(p) = (ap + k)(\mathrm{mod} m)$, где k – фиксировано. В этом случае ключом является пара чисел (a, k). Условие взаимной простоты необходимо для обратимости шифра. Конечно, буквы можно заменять и какими-то другими символами. К семейству шифров замены относятся гомофонические, полиграммные и многоалфавитные шифры.

Гомофоническое шифрование – один из способов защиты от частотной криптоатаки. Каждая буква текста шифруется несколькими символами этого или другого алфавита. Число этих символов пропорционально частотной характеристике шифруемой буквы. Ключом в этом случае является таблица с гомофонией, например:

Буква	Гомофония							
H	17	19	34	41	56			
I	08	22	53	65	88	90	83	
M	03	44						
N	02	09	15	27	32	40	59	
O	01	11	23	42	54	70	80	67
P	33	91						
C	05	10	20					

Одним из возможных вариантов зашифровать текст HOMOPHONIC является следующий: 17 01 44 23 91 41 11 15 88 20.

Полиграммное шифрование

При *полиграммном шифровании* заменяются не буквы текста, а их комбинации. Если заменяются пары букв, то имеем *биграммное шифрование.* Примером биграммного шифрования является шифр ***Плейфера***. Образуем из английского алфавита какой-нибудь квадрат 5 х 5 и будем хранить его, как всякий ключ, в секрете.

Например:

H	A	R	P	S
I	C	O	D	B
E	F	G	K	L
M	N	Q	T	U
V	W	X	Y	Z

Здесь буква J не употребляется или отождествляется с буквой I.

Замена биграмм проводится по правилам:

1) если m_1и m_2находятся в одной строке, то биграмма m_1, m_2шифруется диграммойc_1, c_2, где буквы c_1, и c_2 являются правыми соседями букв m_1, и m_2соответственно; если правого соседа нет, то берется первая буква строки;

2) если m_1, и m_2 – в одном столбце, то берутся нижние соседи с аналогичной оговоркой;

3) если $m_1 = m_2$, то в незашифрованном тексте между ними вставляется незначащая буква (например, X);

4) при нечетном количестве букв в незашифрованном тексте к нему дописывается незначащая буква;

5) в наиболее вероятном случае, когда m_1 и m_2 расположены в разных столбцах и строках, c_1 и c_2 выбираются, как показано на схеме:

$$\begin{array}{cccccccccccc} m_1 & \dots & c_1 & c_2 & \dots & m_2 & c_1 & \dots & m_1 & m_2 & \dots & c_2 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ c_2 & \dots & m_2 & m_1 & \dots & c_1 & m_2 & \dots & c_2 & c_1 & \dots & m_1 \end{array}$$

Покажем это на примере:

m	=	RE	NA	IS	SA	NC	EX
E(m)	=	HG	WC	BH	HR	WF	GV

Еще одна биграммнаякриптосхема, принадлежащая ***Хиллу***, основана на линейной алгебре. Осуществим цифровую кодировку букв английского алфа-

вита: A = 0, B = 1, C = 2, ..., Z = 25. Выберем какую-нибудь обратимую по модулю 26 квадратную матрицу M порядка 2. Это – ключ. Пусть, например,

$$M = \begin{pmatrix} 2 & 5 \\ 3 & 3 \end{pmatrix}, \quad M^{-1} = \begin{pmatrix} 17 & 15 \\ 9 & 20 \end{pmatrix}.$$

Биграммы будем записывать в виде матриц-столбцов. Например:

$$P_1 = \begin{pmatrix} H \\ E \end{pmatrix} = \begin{pmatrix} 7 \\ 4 \end{pmatrix}, \quad P_2 = \begin{pmatrix} L \\ P \end{pmatrix} = \begin{pmatrix} 11 \\ 15 \end{pmatrix}.$$

Шифрование биграмм определим формулой $C = MP$. Зашифруем, для примера, слово $m = P_1P_2$: m = HELP, c = IHTA.

К биграммному шифрованию, так же, как и к шифру простой подстановки, применима частотная криптоатака. Приведем в связи с этим наиболее распространенные английские биграммы с указанием процентов их встречаемости:

TH	6,3	AK	2,0	HA	1,7
IN	3,1	EN	2,0	OU	1,4
ER	2,7	TI	2,0	IT	1,4
RE	2,5	TE	1,9	ES	1,4
AN	2,2	AT	1,8	ST	1,4
HE	2,2	ON	1,7	OR	1,4

Подстановочное шифрование

При *многоалфавитном подстановочном шифровании* задается d шифров простой подстановки, определяемых функциями $f_1, f_2, ..., f_d$, а сообщение

$$m = m_1, m_2, ..., m_d, m_{d+1}, ..., m_{2d}, ...$$

шифруется по правилу

$$E_k(m) = f_1(m_1), f_2(m_2), ..., f_d(m_d), f_1(m_{d+1}), ..., f_d(m_{2d}), ...$$

К таким шифрам относится шифр ***Виженера***. Ключ образуется последовательностью букв $k_1, k_2, ..., k_d$, при этом для буквы a i-го алфавита функция выглядит следующим образом: $f_i(a) = (a+k_i)(\mathrm{mod} m)$.

Пример приведен ниже.

m	=	MULT	IALP	HABE	TENC	RYPT	ION
k	=	KEYS	KEYS	KEYS	KEYS	KEYS	KEY
$E_k(m)$	=	WZJL	SEJH	REZW	DILU	BCNL	SSL

Наряду с подстановочными шифрами известны так называемые *перестановочные (транспозиционные)* шифры. При этом буквы сообщения остаются прежними, но меняют свое расположение в тексте. Приведем два примера.

Сообщение можно разбить на группы букв, скажем, по три буквы, а затем в каждой группе сделать одну и ту же перестановку. Например:

TEACHENCRYPTION → EATHECCRNPTYONI.

То же сообщение можно записать в прямоугольнике 3 x 5:

T	E	A	C	H
E	N	C	R	Y
P	T	I	O	N

Затем можно переписать его по столбцам: TEPENTACICROHYN, что и будет криптограммой (зашифрованным текстом).

Шифр Вернама

В заключение отметим еще один шифр, предложенный ***Вернамом***. Сообщение *m* обычно записывают в виде последовательности нулей и единиц. Длина ключа *k* равна длине сообщения. Шифрование состоит в применении к *m* и *k операции* XOR(исключающее ИЛИ, ранее мы ее называли логическим сложением или сложением по модулю 2), $E_k(m) = m \oplus k$. Очевидно, $D_k = E_k$, так как $(m \oplus k) \oplus k = m \oplus (k \oplus k) = m$. Шифр Вернама считается практически нераскрываемым, так как данное сообщение с помощью подбора ключа, к сожалению, слишком большого, можно преобразовать в любое другое. Основная проблема состоит в хранении и передаче ключа.

В современной компьютерной криптографии используются многие перечисленные шифры как составные части сложных криптосистем.

Лекция 5

Алгоритмы в базах данных

Основные вопросы

1. Общий алгоритм преобразования ключа в адрес.
2. Метод средних квадратов.
3. Деление ключа.
4. Сдвиг разрядов.
5. Метод складывания.
6. Преобразование системы счисления.
7. Метод Лина.
8. Выбор алгоритма преобразования ключа.

Цель. Изучение алгоритмов преобразования ключей записей в адрес памяти базы данных.

Общий алгоритм преобразования ключа в адрес

Общий алгоритм преобразования ключа записи в адрес памяти в базах данных выполняется в три этапа:

1. Если ключ не цифровой, он преобразуется в соответствующее цифровое представление таким образом, чтобы исключить потерю информации, содержащуюся в ключе. Например, буквенные знаки должны переводиться в цифровой код; допускается также представление символьного ключа в виде строки битов.

2. Ключи (в цифровом или битовом представлении) затем преобразуются в совокупность произвольно распределенных чисел, значения которых имеют тот же порядок, что и значения адресов основной области памяти. Набор ключей должен быть распределен по возможности равномерно в диапазоне допустимых адресов.

3. Полученные числа умножаются на константу, что позволяет разместить их строго в диапазоне значений адресов основной области. Например, пусть в результате выполнения этапа 2 мы получаем четырехзначные числа, а в основной области имеется 7000 пакетов. Тогда четырехзначные числа следует умножать 0,7, что позволит распределить получаемые адреса в интервале от 0 до 6999. Этот относительный номер пакета преобразуется в машинный адрес пакета.

Метод средних квадратов

Ключ возводится в квадрат. После этого из полученного результата выделяется число, состоящее из центральных цифр. Например, если ключ

записи равен 172 148, то его квадрат 029634933904. Четыре центральные цифры составляют число 3493.

Деление ключа

В основе данного метода лежит обычное деление чисел. Он дает лучшие результаты, чем метод средних квадратов. Ключ делится на число, равное числу пакетов в основной области или близкое к нему. Делитель должен быть простым числом или числом, которое не содержит небольших сомножителей. Остаток от деления и дает относительный адрес пакета.

Например, если число пакетов равно 10000, то в качестве делителя можно использовать число 9991 (у этого числа один большой делитель 97). Пусть ключ записи равен 172 148. Остаток от деления 172 148 на 9991, равный 2301, будет взят в качестве относительного адреса пакета, в который направляется запись с ключом 172 148.

Сдвиг разрядов

Все разряды числа, являющегося ключом, разбиваются на две части: старшие и младшие разряды. Обе эти части сдвигаются по направлению друг к другу так, чтобы число перекрывающихся разрядов соответствовало длине адреса (рис. 1). Цифры, содержащиеся в перекрывающихся разрядах, суммируются.

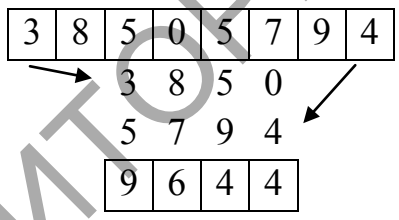


Рис. 1. Метод сдвига разрядов

Метод складывания

Ключ разбивается на части, средняя из которых равна длине адреса (рис. 2). Первая и третья налагаются на вторую подобно тому, как складывается втрое лист бумаги. Цифры затем суммируются. Это метод наиболее удобен для преобразования больших ключей.

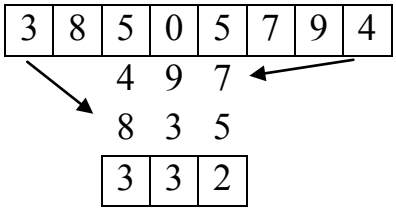


Рис. 2. Складывание частей ключа

Анализ отдельных разрядов ключа

Для достижения равномерного распределения адресов пакетов можно использовать анализ распределения значений чисел и символов в каждом ключе. Выбросив из ключа разряды, имеющие распределение, сильно отличающееся от равномерного, мы можем достичь более равномерного распределения адресов.

Преобразование системы счисления

В основе этого алгоритма лежит преобразование основания системы счисления ключа (например, можно использовать основание 11). После преобразования выделяются младшие разряды.

Пример**.** Пусть ключ равен 172 148. Тогда

$$1\cdot11^5+ 7\cdot11^4+ 2\cdot11^3+ 1\cdot11^2+ 4\cdot11^1 + 8 = 266\ 373.$$

Число 6373, образованное младшими разрядами, будет являться адресом.

При использовании данного метода необходимые расчеты выполняются на компьютере более быстро, чем при использовании методов складывания и сдвига разрядов.

Метод Лина

В данном методе осуществляется представление ключа в системе счисления с основанием *p,* после чего осуществляется деление результата на q^m. Остаток будет искомым адресом. Здесь *p* и *q* – простые числа (или числа, не содержащие небольших множителей), а *m* – целое положительное число.

Рассмотрим пример. Ключ 172148 поразрядно преобразуется в двоичную строку: 0001 0111 0010 0001 0100 1000. После этого осуществляется перегруппировка строки по три разряда: 000 101 110 010 000 101 001 000 = 05620510.

Затем осуществляется деление числа 05 620 510 на величину $q^m= 97^2$ по правилам деления десятичных чисел. Остаток от деления 3 337 есть номер пакета.

Выбор алгоритма преобразования ключа

Наиболее приемлемый алгоритм преобразования ключавыбирают следующим образом: берут достаточно представительный набор ключей файла, применяют к нему все возможные алгоритмы хеширования и определяют число записей, направленных в каждый пакет основной области, и число записей, направленных в область переполнения.

Исследования подобного рода выполнялись неоднократно. Они показали, что лучшие результаты получаются, когда применяется метод деления. Метод средних квадратов дает результаты, близкие к теоретическим, полученным в

случае использования случайного преобразования ключей. Результаты использования сложных методов (метода преобразования основания системы счисления, метода Лина) близки к тем, которые получаются при преобразовании ключей с помощью генератора случайных чисел.

Лучшим является не тот метод, который обеспечивает именно случайное распределение записей, а тот, который обеспечивает равномерное заполнение всего адресного пространства памяти.

Лекция 6

Основы языка JavaScript

Основные вопросы

1. Отладка в браузере Chrome.
2. Редакторы кода JavaScript.
3. Консоль разработчика.
4. Консоль браузера.
5. Объекты.
6. Литералы и свойства.
7. Инструкции в JavaScript.
8. Ввод-вывод. Метод document.write(). Метод prompt(). Метод alert().
9. Управляющие инструкции.

*Цель:*изучение основ программирования на языкеJavaScript.

Отладка в браузере Chrome

Изначально *JavaScript* был создан, чтобы *«сделать веб-страницы живыми»*.

Программы на этом языке называются *скриптами*. Они могут встраиваться в HTML и выполняться автоматически при загрузке веб-страницы.

Скрипты распространяются и выполняются, как простой текст. Им не нужна специальная подготовка или компиляция для запуска.

Когда JavaScript создавался, у него было другое имя – «LiveScript». Однако, язык Java был очень популярен в то время, и было решено, что позиционирование JavaScript как «младшего брата» Java будет полезно. Со временем JavaScript стал полностью независимым языком со своей собственной спецификацией, называющейсяECMAScript, и сейчас не имеет никакого отношения к Java.

Сегодня JavaScript может выполняться не только в браузере, но и на сервере или на любом другом устройстве, которое имеет специальную программу, называющуюся«движком» JavaScript.

У браузера есть собственный движок, который иногда называют «виртуальная машина JavaScript».

Разные движки имеют разные «кодовые имена». Например: V8– в Chrome и Opera.

Движки работают следующим образом:

Движок (встроенный, если это браузер) читает («парсит») текст скрипта.

Например, в браузере JavaScript может:

– Добавлять новый HTML-код на страницу, изменять существующее содержимое, модифицировать стили.

– Реагировать на действия пользователя, щелчки мыши, перемещения указателя, нажатия клавиш.

– Отправлять сетевые запросы на удалённые сервера, скачивать и загружать файлы.

– Задавать вопросы посетителю, показывать сообщения.

– Запоминать данные на стороне клиента («localstorage»).

Как минимум, *три*сильные стороны JavaScript:

– Полная интеграция с HTML/CSS.

– Простые вещи делаются просто.

– Поддерживается всеми основными браузерами и включён по умолчанию.

JavaScript – это единственная браузерная технология, сочетающая в себе все эти три вещи.

Затем он преобразует («компилирует») скрипт в машинный язык.

После этого машинный код запускается и работает достаточно быстро.

Движок применяет оптимизации на каждом этапе. Он даже просматривает скомпилированный скрипт во время его работы, анализируя проходящие через него данные, и применяет оптимизации к машинному коду, полагаясь на полученные знания. В результате скрипты работают очень быстро.

В браузере для JavaScript доступно всё, что связано с манипулированием веб-страницами, взаимодействием с пользователем и веб-сервером.

Редакторы кода

Большую часть своего рабочего времени программисты проводят в редакторах кода.

Есть два основных типа редакторов: IDE и «лёгкие» редакторы. Многие используют по одному инструменту каждого типа.

Термином IDE (IntegratedDevelopmentEnvironment, «интегрированная среда разработки») называют мощные редакторы с множеством функций, которые работают в рамках целого проекта. Как видно из названия, это не просто редактор, а нечто большее.

IDE загружает проект (который может состоять из множества файлов), позволяет переключаться между файлами, предлагает автодополнение по коду всего проекта (а не только открытого файла), также она интегрирована с системой контроля версий, средой для тестирования и другими инструментами на уровне всего проекта.

Можно выбрать бесплатную среду разработки VisualStudioCode. Она – кроссплатформенная.

ДляWindowsестьещёVisualStudio (непутатьсVisualStudioCode). VisualStudio – это платная мощная среда разработки, которая работает только на Windows. Она хорошо подходит для .NET платформы. У неё есть бесплатная версия, которая называется VisualStudioCommunity.

Многие IDE платные, но у них есть пробный период. Их цена обычно незначительна по сравнению с зарплатой квалифицированного разработчика, так что пробуйте и выбирайте ту, что вам подходит лучше других.

«Лёгкие» редакторы менее мощные, чем IDE, но они отличаются скоростью, удобным интерфейсом и простотой. В основном их используют для того, чтобы быстро открыть и отредактировать нужный файл.

Главное отличие между «лёгким» редактором и IDE состоит в том, что IDE работает на уровне целого проекта, поэтому она загружает больше данных при запуске, анализирует структуру проекта, если это необходимо, и так далее. Если надо работать только с одним файлом, то гораздо быстрее открыть его в «лёгком» редакторе.

На практике «лёгкие» редакторы могут иметь множество плагинов, включая автодополнение и анализаторы синтаксиса на уровне директории, поэтому границы между IDE и «лёгкими» редакторами размыты.

Следующие редакторы заслуживают внимания:

- Atom (кроссплатформенный, бесплатный).
- SublimeText (кроссплатформенный, условно-бесплатный).
- Notepad++ (Windows, бесплатный).

Консоль разработчика

Код уязвим для ошибок. Но по умолчанию в браузере ошибки не видны. То есть, если что-то пойдёт не так, мы не увидим, что именно сломалось, и не сможем это починить.

Для решения задач такого рода в браузер встроены так называемые «Инструменты разработки» (Developertools или сокращённо –devtools).

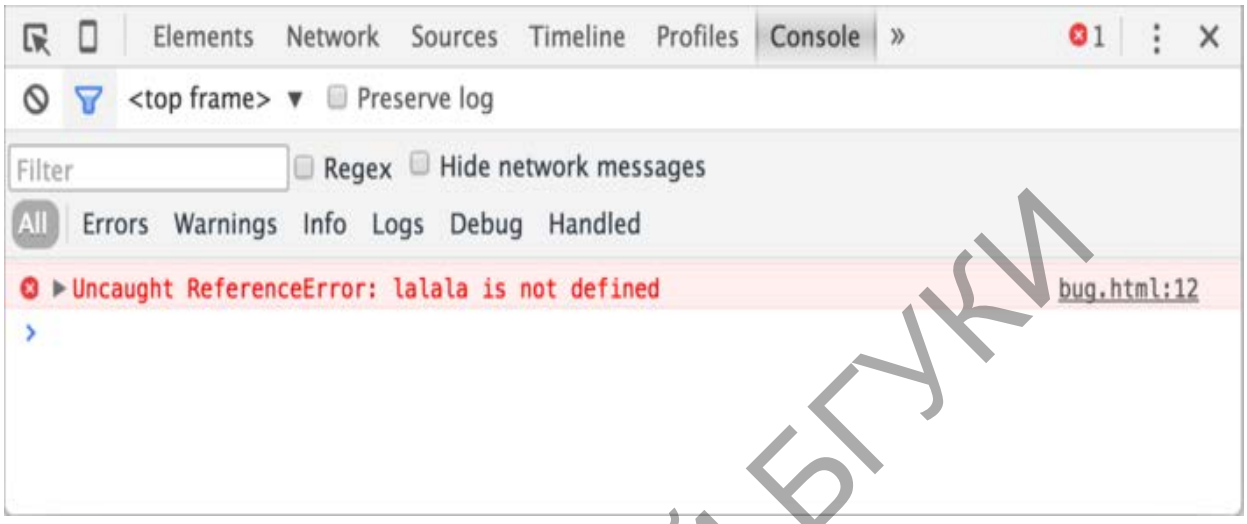
Chrome и Firefox снискали любовь подавляющего большинства программистов во многом благодаря своим отменным инструментам разработчика. Остальные браузеры, хотя и оснащены подобными инструментами, но все же зачастую находятся в роли догоняющих и по качеству, и по количеству свойств и особенностей. В общем, почти у всех программистов есть свой «любимый» браузер. Другие используются только для отлова и исправления специфичных «браузерозависимых» ошибок.

Для начала знакомства с этими мощными инструментами давайте выясним, как их открывать, смотреть ошибки и запускать команды JavaScript.

Откройте страницу bug.html.

В её JavaScript-код закралась ошибка. Она не видна обычному посетителю, поэтому давайте найдём её при помощи инструментов разработки.

Нажмите F12. По умолчанию в инструментах разработчика откроется вкладка Console (консоль).



Точный внешний вид инструментов разработки зависит от используемой версии Chrome. Время от времени некоторые детали изменяются, но в целом внешний вид остаётся примерно похожим на предыдущие версии.

В консоли мы можем увидеть сообщение об ошибке, отрисованное красным цветом. В нашем случае скрипт содержит неизвестную команду «lalala».

Справа присутствует ссылка на исходный код bug.html:12 с номером строки кода, в которой эта ошибка и произошла.

Под сообщением об ошибке находится синий символ>. Он обозначает командную строку, в ней мы можем редактировать и запускать JavaScript-команды. Для их запуска нажмитеEnter.

Обычно при нажатииEnterведённая строка кода сразу выполняется.

Чтобы перенести строку, нажмитеShift+Enter. Так можно вводить более длинный JS-код.

Программы на JavaScript могут быть вставлены в любое место HTML-документа с помощью тега<script>.

Пример

```
<!DOCTYPE HTML>
<html>
<body>
<p>Передскриптом...</p>
<script>
```

```
alert( 'Привет, мир!' );
</script>
<p>...После скрипта.</p>
</body>
</html>
```

Вы можете запустить пример, нажав на кнопку «Play» в правом верхнем углу блока с кодом выше.

Тег <script> содержит JavaScript-код, который автоматически выполнится, когда браузер его обработает.

Если имеется много JavaScript-кода, то можно поместить его в отдельный файл.

Файл скрипта можно подключить к HTML с помощью атрибута src:

```
<script src="/path/to/script.js"></script>
```

Здесь /path/to/script.js – это абсолютный путь до скрипта от корня сайта. Также можно указать относительный путь от текущей страницы. Например, src="script.js" будет означать, что файл "script.js" находится в текущей папке.

Можно указать и полный URL-адрес. Например:

```
<scriptsrc="https://cdnjs.cloudflare.com/ajax/libs/lodash.js/3.2.0/lodash.j
s"></script>
```

Для подключения нескольких скриптов используйте несколько тегов:

```
<script src="/js/script1.js"></script>
<script src="/js/script2.js"></script>
…
```

Как правило, только простейшие скрипты помещаются в HTML. Более сложные выделяются в отдельные файлы.

Польза от отдельных файлов в том, что браузер загрузит скрипт отдельно и сможет хранить его вкеше. Другие страницы, которые подключают тот же скрипт, смогут брать его из кеша вместо повторной загрузки из сети. И таким образом файл будет загружаться с сервера только один раз.

Это сокращает расход трафика и ускоряет загрузку страниц.

Если атрибутsrcустановлен, содержимое тегаscriptбудет игнорироваться.

В одном теге<script>нельзя использовать одновременно атрибутsrcи код внутри.

Нижеприведённый пример не работает:

```
<scriptsrc="file.js">
alert(1); // содержимое игнорируется, так как есть атрибут src
</script>
```

Нужно выбрать: либо внешний скрипт <scriptsrc="…">, либо обычный код внутри тега <script>.

Вышеприведённый пример можно разделить на два скрипта:

```
<script src="file.js"></script>
<script>
alert(1);
</script>
```

На протяжении долгого времени JavaScript развивался без проблем с обратной совместимостью. Новые функции добавлялись в язык, в то время как старая функциональность не менялась.

Преимуществом данного подхода было то, что существующий код продолжал работать. А недостатком – что любая ошибка или несовершенное решение, принятое создателями JavaScript, застревали в языке навсегда.

Так было до 2009 года, когда появился ECMAScript 5 (ES5). Он добавил новые возможности в язык и изменил некоторые из существующих. Чтобы устаревший код работал, как и раньше, по умолчанию подобные изменения не применяются. Поэтому нам нужно явно их активировать с помощью специальной директивы: "usestrict".

Директива выглядит как строка: "usestrict" или 'usestrict'. Когда она находится в начале скрипта, весь сценарий работает в «современном» режиме.

Пример:

```
"usestrict";
// этот код работает в современном режиме
```

Отметим, что вместо всего скрипта"usestrict"можно поставить в начале большинства видов функций. Это позволяет включить строгий режим только в конкретной функции. Но обычно люди используют его для всего файла.

Убедитесь, что «usestrict» находится в начале.

Проверьте, что "usestrict" находится в первой исполняемой строке скрипта, иначе строгий режим может не включиться.

Здесь строгий режим не включён:

```
alert("somecode");
// "usestrict" ниже игнорируется - он должен быть в первой строке
"usestrict";
// строгий режим не активирован
```

Над "usestrict" могут быть записаны только комментарии.

Нет никакого способа отменить usestrict.

Нет директивы типа "nousestrict", которая возвращала бы движок к старому поведению.

Как только мы входим в строгий режим, отменить это невозможно.

Консоль браузера

При использовании консоли браузера для тестирования функций, надо обратить внимание, чтоusestrictпо умолчанию в ней выключен.

Иногда, когда usestrictимеет значение, вы можете получить неправильные результаты.

Можно использоватьShift+Enterдля ввода нескольких строк и написать в верхнейстрокеusestrict:

```
'usestrict'; <Shift+Enter для перехода на новую строку>
//  ...ваш код...
<Enter для запуска>
```

В большинстве браузеров, включая Chrome и Firefox, это работает.

В старых браузерах консоль не учитывает такойusestrict, там можно «оборачивать» код в функцию, вот так:

```
(function() {
  'use strict';
  // ...вашкод...
})()
```

Объекты

В JavaScript существует семь типов данных. Шесть из них называются «примитивными», так как содержат только одно значение (будь то строка, число или что-то другое).

Объекты же используются для хранения коллекций различных значений и более сложных сущностей. В JavaScript объекты используются очень часто, это одна из основ языка. Объект может быть создан с помощью фигурных скобок{…}с необязательным списком*свойств*. Свойство – это пара «ключ: значение», где ключ – это строка (также называемая «именем свойства»), а значение может быть, чем угодно.

Можно представить объект в виде ящика с подписанными папками. Каждый элемент данных хранится в своей папке, на которой написан ключ. По ключу папку легко найти, удалить или добавить в неё что-либо.

Пустой объект («пустой ящик») можно создать, используя один из двух вариантов синтаксиса:

```
letuser = newObject(); // синтаксис "конструктор объекта"
letuser = {};  // синтаксис "литерал объекта"
```

Обычно используют вариант с фигурными скобками {...}. Такое объявление называют *литералом объекта* или *литеральной нотацией*.

Литералы и свойства

При использовании литерального синтаксиса {...} мы сразу можем поместить в объект несколько свойств в виде пар «ключ: значение»:

```
letuser = {     // объект
name: "John",  // под ключом "name" хранится значение "John"
age: 30        // под ключом "age" хранится значение 30
};
```

Свойства объекта также иногда называют *полями объекта*.

У каждого свойства есть ключ (также называемый «имя» или «идентификатор»). После имени свойства следует двоеточие ":", и затем указывается значение свойства. Если в объекте несколько свойств, то они перечисляются через запятую.

В объектеusersейчас находятся два свойства:

Первое свойство с именем "name" и значением"John".

– Второе свойство с именем"age"и значением30.

Можно сказать, что наш объект user – это ящик с двумя папками, подписанными «name» и «age».

Мы можем в любой момент добавить в него новые папки, удалить папки или прочитать содержимое любой папки.

Для обращения к свойствам используется запись «через точку»:

```
// получаем свойства объекта:
alert( user.name ); // John
alert( user.age ); // 30
```

Значение может быть любого типа. Давайте добавим свойство с логическим значением:

```
user.isAdmin = true;
```

Для удаления свойства мы можем использовать оператор delete:

```
deleteuser.age;
```

Имя свойства может состоять из нескольких слов, но тогда оно должно быть заключено в кавычки:

```
let user = {
  name: "John",
  age: 30,
"likesbirds": true  // имя свойства из нескольких слов должно быть в кавычках
};
```

Последнее свойство объекта может заканчиваться запятой:

```
let user = {
  name: "John",
```

```
  age: 30,
}
```

Это называется «висячая запятая». Такой подход упрощает добавление, удаление и перемещение свойств, так как все строки объекта становятся одинаковыми.

Для свойств, имена которых состоят из нескольких слов, доступ к значению «через точку» не работает:

```
// это вызовет синтаксическую ошибку
user.likesbirds = true
```

JavaScript видит, что мы обращаемся к свойству user.likes, а затем идёт непонятное слово birds. В итоге синтаксическая ошибка.

Точка требует, чтобы ключ был именован по правилам именования переменных. То есть не имел пробелов, не начинался с цифры и не содержал специальные символы, кроме $и_.

Для таких случаев существует альтернативный способ доступа к свойствам через квадратные скобки. Такой способ сработает с любым именем свойства:

```
letuser = {};
// присваивание значения свойству
user["likesbirds"] = true;
// получение значения свойства
alert(user["likes birds"]); // true
// удалениесвойства
deleteuser["likesbirds"];
```

Сейчас всё в порядке.

Обратите внимание, что строка в квадратных скобках закавычена (подойдёт любой тип кавычек).

Квадратные скобки также позволяют обратиться к свойству, имя которого может быть результатом выражения. Например, имя свойства может храниться в переменной:

```
letkey = "likesbirds";
// тожесамое, чтоиuser["likesbirds"] = true;
user[key] = true;
```

Здесь переменная key может быть вычислена во время выполнения кода или зависеть от пользовательского ввода. После этого мы используем её для доступа к свойству. Это даёт нам большую гибкость.

Пример:

```
let user = {
  name: "John",
```

```
  age: 30
};
letkey = prompt("Что вы хотите узнать о пользователе?", "name");
// доступ к свойству через переменную
alert( user[key] ); // John (если ввели "name")
```

Запись «через точку» такого не позволяет:

```
let user = {
  name: "John",
  age: 30
};
let key = "name";
alert( user.key ); // undefined
```

Здесь переменная key может быть вычислена во время выполнения кода или зависеть от пользовательского ввода. После этого мы используем её для доступа к свойству. Это даёт нам большую гибкость.

Пример:

```
let user = {
  name: "John",
  age: 30
};
letkey = prompt("Что вы хотите узнать о пользователе?", "name");
// доступ к свойству через переменную
alert( user[key] ); // John (если ввели "name")
```

Запись «через точку» такого не позволяет:

```
let user = {
  name: "John",
  age: 30
};
let key = "name";
alert( user.key ); // undefined
```


Инструкции в JavaScript

Скрипты на JavaScript практически полностью состоят из инструкций. В JavaScript есть три типа инструкций: условные, циклы, переходы. Условные инструкции, например,ifиswitchзаставляют интерпретатор выполнить или пропустить фрагмент кода.Циклы, например, foriwhileзацикливают фрагмент кода, то есть повторяют его определенное количество раз. Инструкции

переходов, например, break и return, которые заставляют JavaScript перейти на другой фрагмент кода (часть программы).

Простые инструкции - это выражения, также инструкция присваивания, инкремент, декремент, оператор delete и остальные.

Чтобы объединить несколько инструкций в одну (блок инструкций), нужно поместить их в фигурные скобки. Пример блока инструкций:

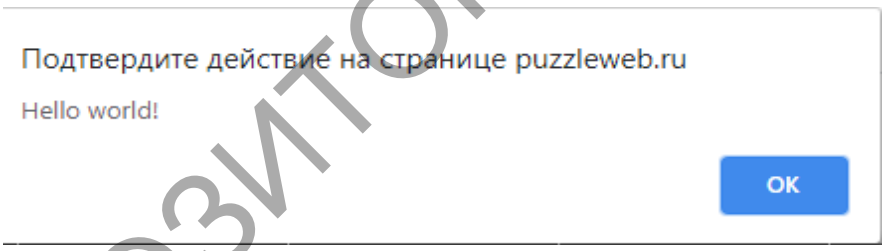
```
{
a *= b;
c = a + b;
document.write(c);
}
```

Методы ввода-вывода на экран

Методalert()выводит на экран модальное окно с сообщением. Модальное окно означает, что выполнение сценария и дальнейшее взаимодействие со страницей приостанавливается до тех пор, пока не закроется данное окно, в данном случае, пока не будет нажата кнопка OKдля продолжения работы. Например, следующий Script

```
alert("Helloworld!");
```

выведет модальное окно:



Метод prompt()

Методprompt()выводит на экран модальное окно приглашения на ввод данных пользователем.

Синтаксис метода:

varимя_переменной = prompt(msg, defaultText);

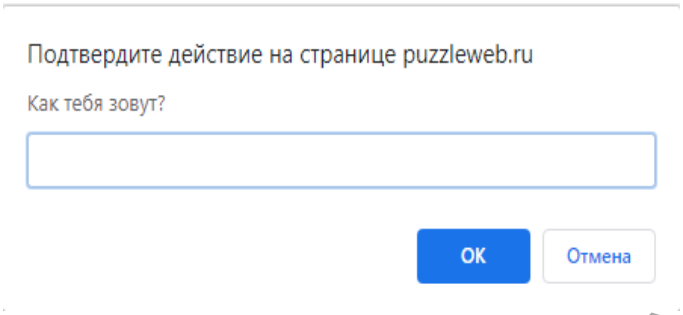
Имя_переменной нужно заменить именем используемой переменной, которой будет присвоено значение возвращаемое методом prompt(), msg - сообщение, которое будет показано пользователю (обычно это вопрос), defaultText - строка, которая отображается. Пример скрипта:

```
<script>
  varmyName = prompt("Как тебя зовут?","");
      alert("Привет " + myName + "!");
```

</script>

Пользователь должен, что-нибудь ввести и нажать OK, или отменить ввод нажав на CANCEL.

Метод prompt() возвращает, то что ввел пользователь – строку или специальное значение null, если ввод был отменен.



Метод document.write()

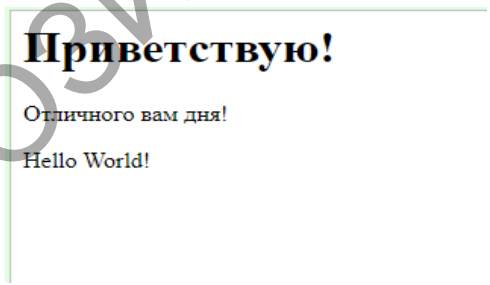
Методdocument.write()выводит на страницу переданные ему аргументы.

Синтаксис метода:

document.write(arg1,arg2,arg3,...);

Аргументов может быть любое количество, и они могут быть любых типов, при выводе они преобразуются в строки. Пример:

```
document.write("<h1>Приветствую!</h1><p>Отличного вам дня!</p>");
document.write("HelloWorld!");
```



Метод document.write()работает только на этапе загрузки страницы. Если document.write()вызвать после того, как страница загрузилась, результатом будет - перезаписанная страница, с текстом, который был добавлен с помощью document.write(). Documentявляется одним из предопределенных объектов JavaScript, а write() – это предопределенный метод объекта document. Точкаобъединяет объекти метод, показывая, что данный метод принадлежит объектуdocument.

Условные инструкции

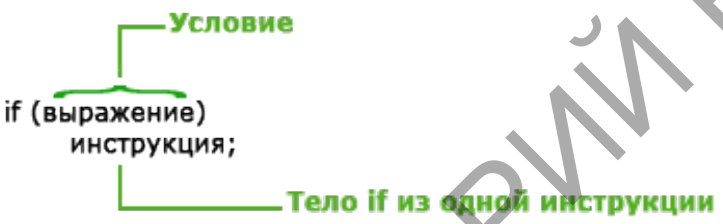
Управляющие инструкции – это инструкции, которые позволяют управлять выполнением программного кода. Обычно выполняемый код в управляющей инструкции называют телом этой инструкции.

Управляющие инструкции могут быть вложенными, а также использоваться внутри других управляющих инструкций.

По умолчанию интерпретатор JavaScript выполняет инструкции одну за другой в порядке их следования в исходном коде. В тех случаях, когда выполнение или невыполнение некоторых инструкций должно зависеть от выполнения или невыполнения некоторого условия, используются условные инструкции.

Инструкция if

Инструкция if имеет две формы. Синтаксис первой формы:



Выражение в круглых скобках называется условием выполнения инструкции ifили краткоусловием. Сначала вычисляется значение выражения. Полученное значение, если необходимо, неявно преобразуется к булеву типу. Если результатом вычисления выражения является значениеtrue, то инструкция выполняется. Если выражение возвращаетfalse, то инструкция не выполняется.

```
if (true)
  alert("Выполнено!");
 if (false)
  alert("Не выполнится!");
```

Синтаксис ifпозволяет выполнить только одну инструкцию, однако если требуется выполнить более одной инструкции нужно использовать составную инструкцию:

```
if (true) {
  var str = "Hello!";
  alert(str);
}
```

Рекомендуется всегда использовать составную инструкцию, даже если нужно выполнить всего одну строку кода.

Синтаксис второй формы if

```
if (выражение)
  инструкция;
else
  инструкция;
```

Ключевое слово else позволяет добавить инструкцию, выполняемую в том случае, если условие имеет ложное значение:

```
if (false)
  alert("Не выполнится");
else
  alert("Выполнится");
```

Управляющие инструкции могут быть вложенными, что позволяет создавать следующие конструкции:

```
var num = 2;
if (num == 1) {
  alert("значение num: " + num);
} else if (num == 2) {
  alert("значение num: " + num);
} else {
  alert("Не знаю такого числа!");
}
```

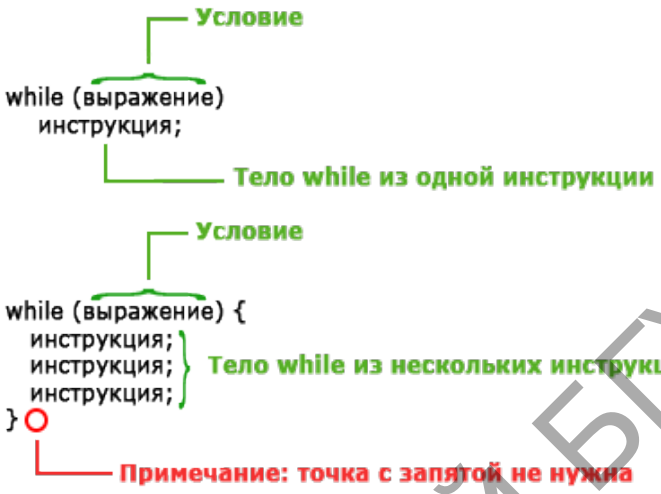
В этом коде нет ничего особенного. Это просто последовательность инструкций, где каждая инструкция if является частью else предыдущей инструкции if. Такая форма записи на первый взгляд может показаться не совсем понятной, поэтому рассмотрим синтаксически эквивалентную форму, показывающую вложенность инструкций if.

```
var num = 2;
if (num == 1) {
  alert("значение num: " + num);
}
else {
  if (num == 2) {
   alert("значение num: " + num);
  }
  else {
    alert("Не знаю такого числа!");
  }
}
```

Циклы JavaScript

Цикл – это управляющая инструкция, позволяющая повторять выполнение программного кода определённое количество раз. Каждое отдельное исполнение инструкций в теле цикла называется итерацией.

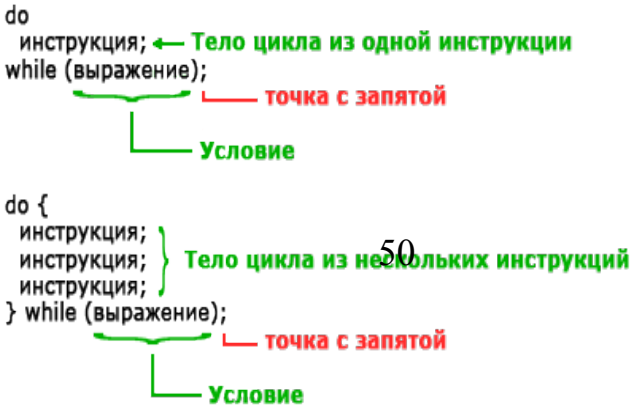
Синтаксис циклаwhile:



Выражение в круглых скобках называется условием выполнения цикла или кратко условием. Сначала вычисляется значение выражения. Полученное значение, если необходимо, неявно преобразуется к булеву типу. Если результатом вычисления выражения является значение true, то инструкция, расположенная в теле цикла, выполняется, затем управление передаётся в начало цикла и условие вычисляется снова. Если результатом вычисления выражения является значение false, интерпретатор завершает работу цикла и переходит к выполнению инструкции следующей за циклом. Таким образом, интерпретатор снова и снова выполняет код расположенный в теле цикла, пока условие остаётся истинным:

```
vari = 0;
while (i< 3) {   // Выполнять код, пока значение переменной i меньше 3
 alert("i: " + i);
  i++;           // Увеличиваем значение переменной i
}
```

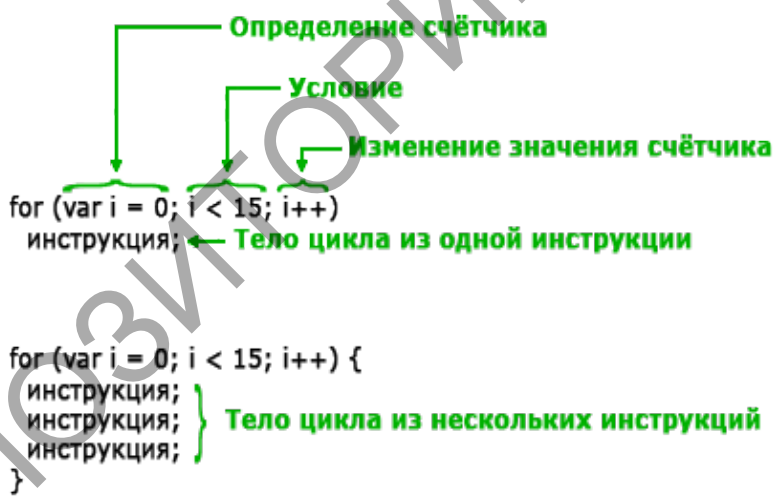
Цикл do-while



Циклdo-whileпохож на циклwhile, за исключением того, что проверка условия выполнения цикла производится после первой итерации, а не перед ней, и завершается цикл точкой с запятой. Так как условие проверяется после итерации, код в теле цикла do-while всегда выполняется минимум один раз:

```
<html>
<body>
<script>
  var count = 0;
  do {
document.write(count + " ");
    count++;
  } while(count < 5);
</script>
</body>
</html>
```

Цикл for



В цикле for располагаются три выражения, разделяемые точкой с запятой. Эти три выражения имеют следующий порядок выполнения:

Первое выражение всегда вычисляется только один раз – перед первой итерацией. Поэтому обычно в качестве первого выражения выступает определение переменной, которая используется в условии выполнения цикла в качестве счётчика.

Второе выражение определяет условие выполнения цикла. Оно вычисляется перед каждой итерацией и определяет, будет ли выполняться тело цикла. Если результатом вычисления выражения является истинное значение, программный код в теле цикла выполняется. Если возвращается ложное значение, выполнение цикла завершается и управление переходит к следующей

после цикла инструкции. Если при первой проверке условия, оно оказывается ложным, код в теле цикла не выполнится ни разу.

После каждой итерации вычисляется третье выражение. Обычно его используют для изменения значения переменной, которая используется в проверке условия выполнения цикла.

Примерциклаfor:

```
<html>
<body>
<script>
   for (var i = 1, j = 5; i<= 5; i++, j--)
document.write(i + " " + j +"<br>");
</script>
</body>
</html>
```

Лекция7

Работа в среде VisualBasic

Основные вопросы

1. Запуск среды программированияVisualBasic.
2. Интерфейс среды программирования.
3. Окно формы.
4. Типы данных и операции над ними.
5. Работа с элементами среды программирования.
6. Основные принципы объектно-ориентированного программирования

Цель. Изучение основных приемов работы в среде программирования VisualBasic.

Запуск среды программированияVisualBasic

VisualBasic.Netявляетсячастьюплатформы.NetFramework, котораявключаетвсебяобщуюсредувыполненияприложенийCLR (CommonLanguageRunTime) инаборбиблиотекклассовBCL (BaseClassLibrary).

Общая среда выполнения приложений позволяет использовать коды программ, написанные на разных языках программирования совместно. Таких .Net-языков около двух десятков (VisualBasic.Net, VisualC++.Net, VisualC#.Netи др.). Подобная языковая совместимость достигается за счет компиляции программ, написанных на различных языках, в общий промежуточный язык IL (IntermediateLanguage). При этом среда CLRпонимает только язык IL.

Библиотека классов .NetFrameworkимеет иерархическую структуру и состоит из пространства имен, которое позволяет избежать использования одинаковых имен классов. В структуре классов могут встречаться классы с одинаковыми именами, но существовать они будут порознь, каждый в своем пространстве. Доступ к существующим пространствам имен осуществляется через директиву Imports, которая имеет следующий синтаксис:

Importsимя_пространства_имен

Пространства имен, поставляемые фирмой Microsoft, начинаются либо с ключевого слова System, либо с Microsoft. Для наглядности желательно указывать пространства имен. В случае работы с графикой это System.Drawing.Graphics, для работы с базами данных –System.Data.ADO. По умолчанию имя пространства имен для проектов в VisualBasic.Netсовпадает с именем проекта.

Ниже приведем минимальные теоретические сведения по VisualBasic.Net, необходимые для понимания текстов программ.

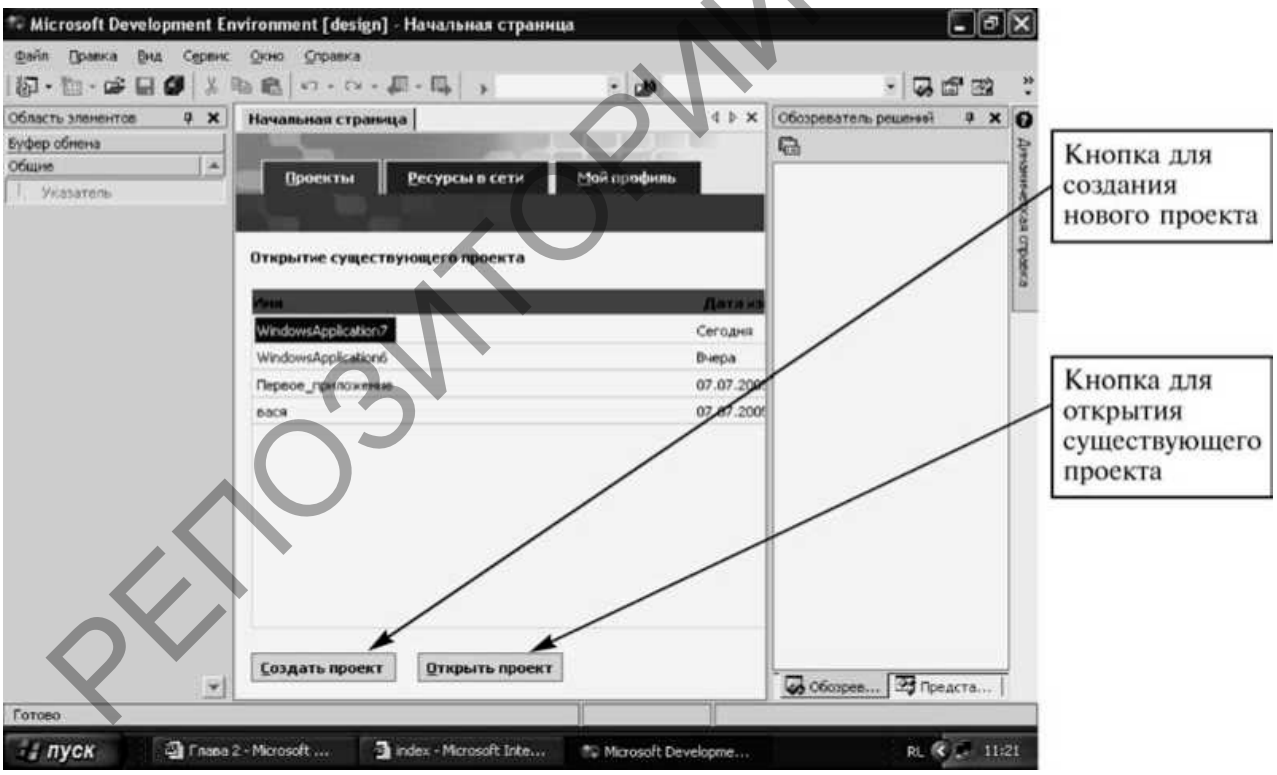
Обратите внимание, что в текстах программ строка заголовка процедуры в некоторых примерах превосходит ширину страницы и не помещается на ней целиком, поэтому часть ее была перенесена на следующую строку. Например,

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal eAs
System.EventArgs) Handles MyBase.Load
```

Такая запись не должна вводить в заблуждение при наборе текста кода, потому что при создании процедуры вы выбираете объект и событие, а остальную строку заголовка среда программирования дописывает сама. Однако, для удобства чтения листингов программ, строка заголовка на последующих страницах будет выделена полужирным шрифтом.

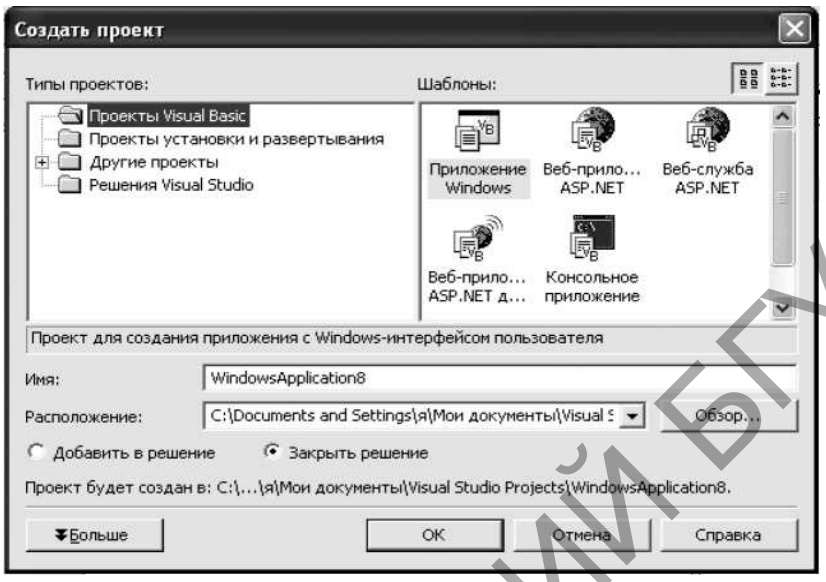
Интерфейс среды программирования

После того как вы откроете VisualBasic.Net, на экран монитора будет выведена страница VisualStudioStartupPage, с помощью которой можно создавать новые проекты и открывать ранее созданные.

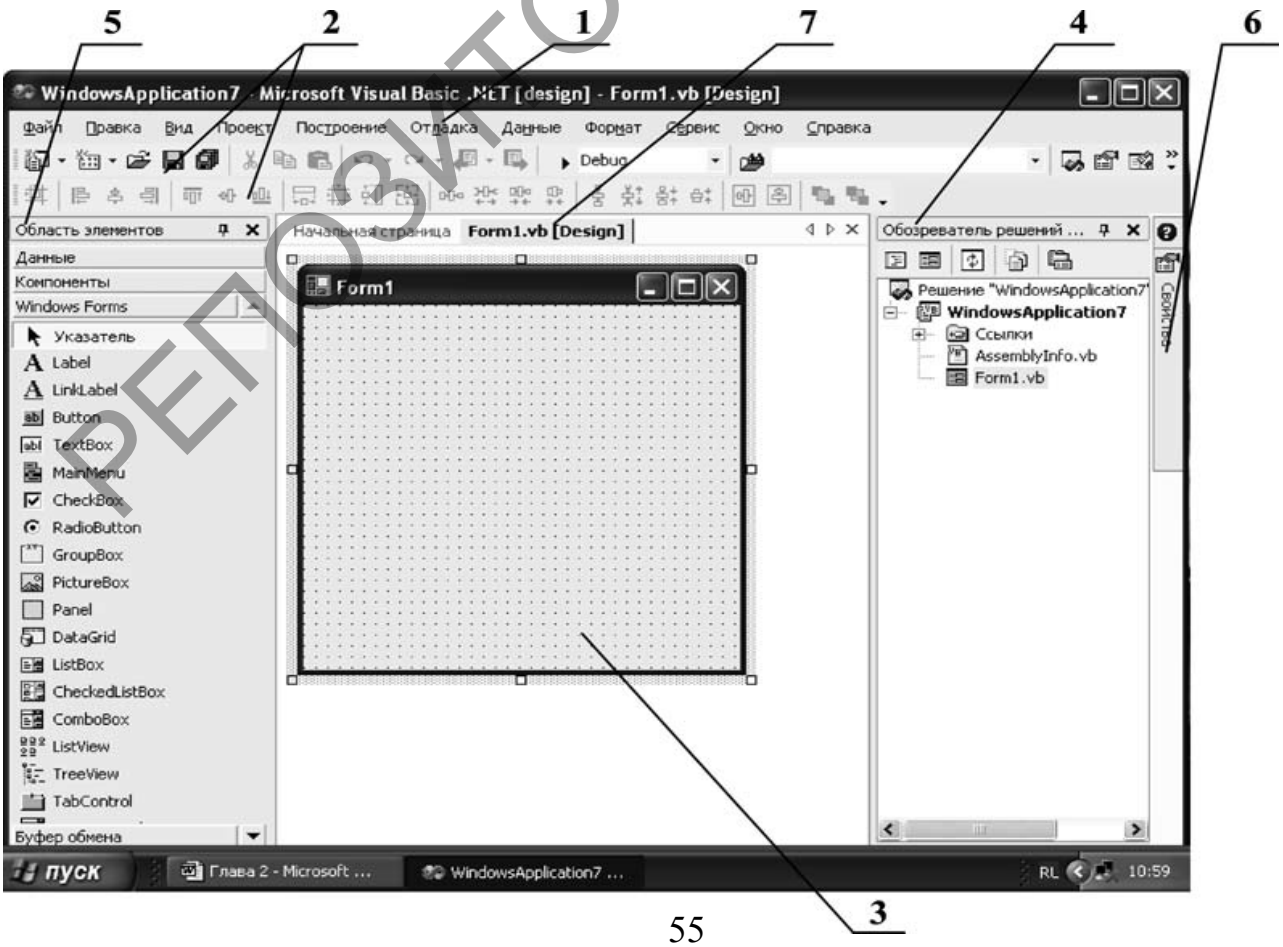


Если щелкнуть мышью на кнопке Создать проект. Появится одноименное диалоговое окно для выбора типа проекта, показанное на рисунке ниже. Доступ к этому окну можно получить также через главное меню Файл=>Создать=>Проектили нажав одновременно три клавиши Ctrl + Shift + N.

Как и другие языки, поддерживающие платформу .Net, Visual Basic.Netпозволяет создавать различные проекты, перечень которых приведен в области Типы проектов левой части диалогового окнаСоздать проект. Выберите папку Проекты VisualBasic, а в области Шаблонывыберите значок Приложение Windows. В текстовых полях Имяи Расположениеможно указать имя приложения и адрес, где будут сохранены его файлы.



После щелчка на кнопке ОКбудет создан новый проект, окно которого приведено ниже.



Следует иметь в виду, что не все элементы интерфейса могут быть одновременно отображены на экране. Рассмотрим более детально элементы интерфейса, отображенные на нашем рисунке.

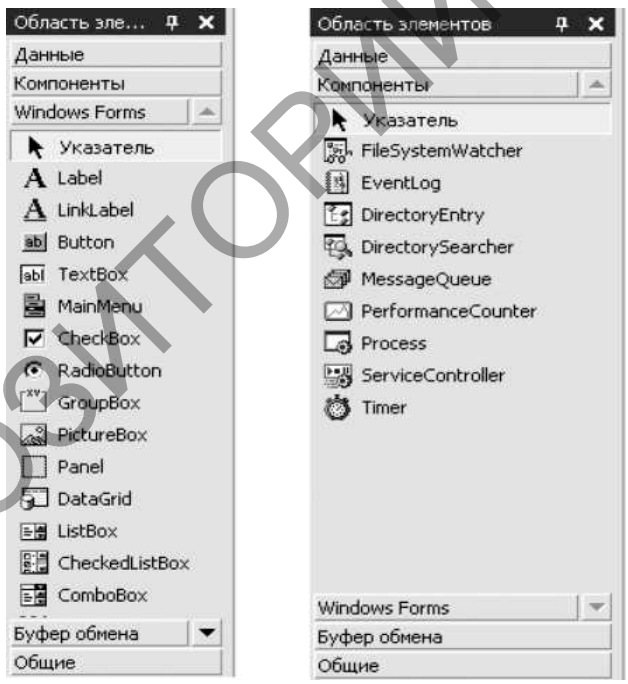
1. Главное меню.С его помощью можно получить доступ к любой команде VisualBasic .NET, однако постоянно использовать их в своей работе не всегда удобно.

2. Панели инструментов.Здесь размещены кнопки, наиболее часто используемых команд VisualBasic .NET.

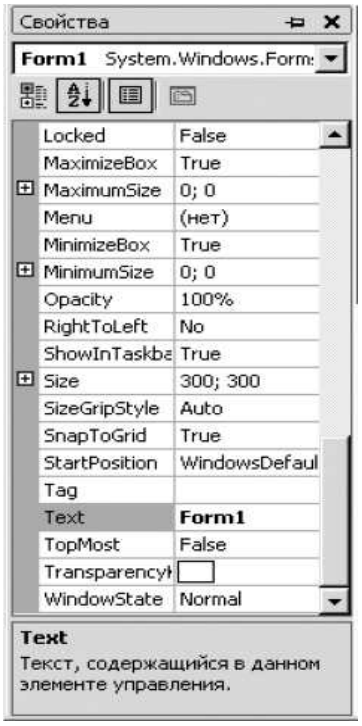
3. Форма.Форма является окном Widows, в котором размещаются различные объекты, создавая таким образом пользовательский интерфейс для программ.

4. Окно Обозреватель решений.В этом окне отображаются все файлы, относящиеся к текущему проекту VisualBasic.NET.

5. Область элементов.Область содержит закладки окон WindowsForms,Компоненты, Данныеи т. д., в которых приведены списки компонентов, размещаемых в окне формы. Это показано на рисунке ниже.

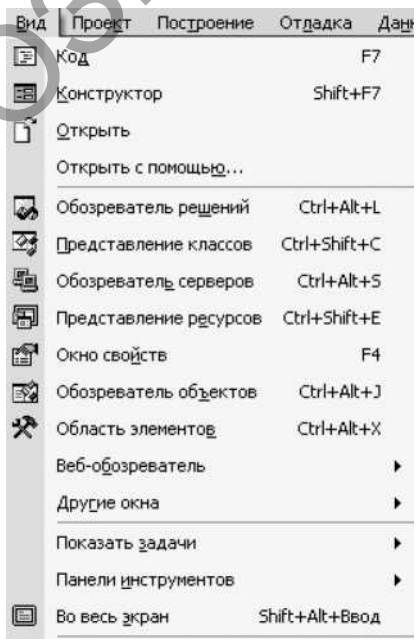


6. Окно Свойства(показано в раскрытом виде на рисунке). В окне отображаются свойства выделенных на данный момент объектов, расположенных на форме, или свойства самой формы. Для раскрытия этого окна нажмите функциональную клавишу F4.



7. Вкладки Конструктор и Код.Соответственно отображают созданные формы (например, Forml.vb [Design]) и содержат окна редакторов кодов– Вкладки Конструктор и Код.Соответственно отображают созданные формы (например, Forml.vb [Design]) и содержат окна редакторов кодов.

Настройка внешнего вида экрана, открытие и закрытие окон производится через главное меню Вид, приведенное на рисунке.



При создании пользовательского интерфейса проекта логично выделить следующие основные этапы:

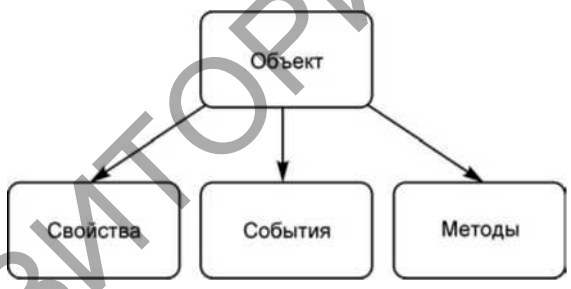
1) выбор из Области элементов(окно WindowsForms) соответствующих компонентов и размещение их на форме; после размещения компоненты становятся объектами, имеющими свойства;

2) настройка свойств объектов в окне Свойства;

3) написание кода программы в окне редактора кода.

Окно формы

Окно формы формы является главным элементом создаваемого приложения. Сама форма служит контейнером для элементов управления. У формы, как у любого объекта, есть свойства. Помимо свойств объектам в VisualBasicприписываются событияи методы. События определяются действиями пользователя (например, одинарный или двойной щелчок левой кнопкой мыши на объекте), самими объектами (например, таймер сам генерирует событие) или операционной системой. События запускают на выполнение код программы, записанной в соответствующей процедуре. Методы– это команды, которые может выполнить объект (например, метод Move, вызывающий перемещение объекта по форме).



Многие свойства оказываются общими для различных объектов. Рассмотрим некоторые из них на примере формы. Выделите форму и просмотрите перечень ее свойств на панели свойств.

Свойство Nameпозволяет изменять имя формы, под которым она фигурирует в коде программы. Это очень важное свойство. Если имя в коде программы будет указано неправильно, объект не будет найден. Поэтому при написании кода во избежание ошибок, которые могут возникнуть при наборе с клавиатуры (например, символ С выглядит одинаково в кириллице и в латинском алфавите), имена объектов лучше копировать из панели свойств и вставлять в код программы. Для формы ее имя указывается в заголовке окна проекта.

Свойство Captionпозволяет изменять заголовок формы.Для других объектов это просто надпись на их поверхности.Она нест декоративно-поясняющую функцию и не влияет на работу кода.Измените заголовок формы:

в списке панели свойств для формы выделите двойным щелчком свойство Caption и введите вместо Form1 свое имя.

Свойство BackColorпозволяет изменить цвет рабочего поля формы. Выделите свойство BackColorи выберите в палитре свой любимый цвет.

Значения свойств Leftи Topзадают координаты левого верхнего угла формы на экране при запуске программы на выполнение. Размерность координат можно выбрать с помощью свойства ScaleModeиз списка. Twip(твип) – единица измерения, введенная программистами Microsoftи равная 1/1440 логического дюйма (величина логического дюйма определяется разрешением экрана). С учетом того, что отсчет координат производится от левого верхнего угла экрана монитора. Введите (в панели свойств) для свойств Leftи Topзначения 0.

Запустите проект на выполнение, нажав клавишу F5или щелкнув на кнопке Запускна Панели инструментов ►, и убедитесь, что форма будет расположена в левом верхнем углу экрана.

Попытайтесь перетащить форму за заголовок. Если свойство формы - Movableимело значение True, то передвинуть форму удастся. Для выхода из режима выполнения нажмите одновременно две клавиши ALTи F4или щелкните по кнопке Остановкана панели инструментов ■. Установите значение свойства Movableравным False, запустите проект на выполнение и попытайтесь передвинуть форму. Это вам не удастся. Форма останется на том месте, координаты которого были заданы в свойствах Leftи Top. Положение формы можно также задать в окне макета формы, переместив мышью изображение формы на экране нарисованного монитора в требуемую позицию.

Свойство Borderstyleзадает различные варианты границы формы в режиме выполнения. Свойство Pictureпозволяет размещать на форме рисунок в качестве фона (подложки).

Типы данных и операции над ними

Типы данных и операции над нимиКлассификация типов данных, принятая в среде VisualBasic.Net, несколько отличается от принятой в VisualBasic 6.5. Это показано в таблице ниже.

Использование конкретного типа определяется спецификой решаемой задачи. Так, для хранения символьных данных следует использовать тип String. Если заранее неизвестно, какой тип понадобится, то можно воспользоваться типом Object, который позволяет хранить любые данные. Недостатком использования этого типа является большой объем требуемой памяти и снижение быстродействия выполнения операций.

Тип	Описание	Диапазон значений
Byte	Однобайтовое число	0 - 255
Boolean	Логическая переменная	True False
Date	Дата и время 8 байт	От 1.01.0001 до 31.12.9999
Decimal	12-байтовое десятичное число	-79 228 162 514 264 337 593 543 950 335
Double	8-байтовое вещественное число	От $4{,}94 \cdot 10^{-324}$ до $1{,}78\text{-}10^{308}$от $-1{,}79\ -10^{308}$ до $-4{,}94\ -10^{-324}$
Object	4 байта	Любое значение любого типа
Short	2-байтовое целое число со	От -32 768 до +32 768
Integer	4-х байтовое целое число	От -2 146 483 648 до +2 146
Long	8-ми байтовое целое число со знаком	От -9 223 372 036 654 775 808 до +9 223 372 036 654 775 807
Single	4-х байтовое вещественное число	От $1{,}4 \cdot 10^{-45}$до $3{,}4\text{-}10^{38}$от $-3{,}4 \cdot 10^{38}$ до $-1{,}4 \cdot 10^{-45}$

Для хранения целых или вещественных значений предусмотрено несколько типов. Выбирать из них следует, руководствуясь диапазоном значений типа и соображениями экономии памяти.

Для уточнения значений диапазонов типов данных в процессе работы можно пользоваться подсказкой, вызываемой с помощью объекта Console. Например, чтобы уточнить минимальное допустимое значение для типа Long, надо записать в окне редактора кода какой-либо процедуры строку Console.WriteLine(Long.MinValue):

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
    Console.WriteLine(Long.MinValue)
End Sub
```

и навести указатель мыши на слово MinValue. На экран будет выведена консоль с запрашиваемой информацией:

Типы данных могут быть преобразованы один в другой. Для преобразования служат специальные функции, имеющие в названии префикс C, за которым следует название типа данных, в который происходит преобразование выражения. Так, функция Cint (12.3)должна выдать в результате преобразования 12. Проверим это.

Задача 1. Преобразуйте константу действительного типа12.3в целочисленную.

Решение. Нарисуйте на форме объект Label1. В окне редактора кода откройте процедуру Label1_Click()и запишите следующую строчку:

```
Private Sub Label1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Label1.ClickLabel1.Text = Cint(12.3)
EndSub
```

Нажмите функциональную кнопку F5. В появившемся окне формы щелкните на объекте Labelли получите ответ 12.

Для преобразования символа в код ASCIIиспользуется команда:

```
Асс("преобразуемый символ")
```

Задача 2.Найдите код ASCIIдля символа @.

Решение. Разместите на форме объект Label1. В окне редактора кода введите следующий текст:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
Label1.Text = Asc("@") EndSub
```

Нажмите функциональную клавишу F5и получите число 64.

Особенностью типа данных Dateявляется то, что этот тип хранит в себе одновременно информацию о дате и о времени.

Задача 3. Напишите программу для вывода на форму значения текущей даты и времени.

Решение. Нарисуйте на форме объект Label1. Для получения текущей даты и времени используем структуру `DateTime`. Если надо получить текущие дату и время следует обратиться к свойству структуры `Now`, то есть записать DateTime.Now.

В окне редактора кода введите следующую программу:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
Label1.Text = DateTime.Now End Sub
```

Чтобы узнать только текущую дату, то надо обратиться к свойству Today.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
    Labell.Text = DateTime.Today
        End Sub
```

Чтобы ввести какую-либо дату, в начале и конце числового значения даты ставятся знаки #. Обратите внимание, что вначале указывается месяц, затем день и год.

Большое количество операций предусмотрено над типом данных String.

Задача 4. Объедините две символьных константы (две строки) в одну и вывести ее на форму.

Решение. Нарисуйте на форме объект Label1. Для объединения строк используется символ &или +. Приэтомстрокизаключаются в кавычки.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
Labell.Text = "Здравствуй, " & "Это я" EndSub
```

Чтобы слова в результирующей строке не сливались, можно после слова Здравствуй, поставить несколько пробелов. Кроме того, для создания пробелов предусмотрена специальная функция Space(), в аргументе которой записывается число требуемых пробелов. Так, чтобы в предыдущем примере между строками было выставлено три пробела, надо записать

```
Labell.Text = "Здравствуй," &Space(3) & "это я"
```

Для определения номера позиции первого вхождения одной строки (или одиночного символа) в другую существует функция InStr().

Задача 5. Напишите программу для определения номера позиции первого вхождения (появления) символа тв строке Здравствуй, это я.

Решение. Нарисуйте на форме объект Label1. В качестве первого аргумента записывается строка, в которой производится поиск, в качестве второго аргумента – строка (или символ), вхождение которого требуется определить.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
Labell.Text = InStr("Здравствуй, это я", "т")
EndSub
```

Ответ будет 7, так как счет символов строки ведется с 1.

Для определения номера позиции последнего вхождения символа в строку служит функция InStrRev().

Для подсчета количества символов в строке служит функция Len().

Задача 6. Подсчитайте количество символов в строке Здравствуй, это я.

Решение. Нарисуйте на форме объект Label1.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
Label1.Text = Len("Здравствуй, это я")
EndSub
```

Чтобы оставить от строки несколько символов, убрав остальные, используйте функцию Mid().

Задача 7. Оставьте от строки Здравствуй, это ятолько символы ра.

Решение. Нарисуйте на форме объект Label1. Функция Mid()имеет

следующие аргументы:

– первый аргумент – сама обрабатываемая строка;

– второй аргумент – номер позиции символа в строке, начиная с которого отсчитываются символы, которые следует оставить;

– третий аргумент – количество символов, которые надо оставить.

В нашем случае позиция символа рравна 3, а количество символов, которые надо оставить, равно 2.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
Labell.Text = Mid("Здравствуй, это я", 3, 2)
EndSub
```

Основные принципы объектно-ориентированного программирования

К основным принципам объектно-ориентированного программированияотносятся инкапсуляция, наследование и полиморфизм.

Инкапсуляция– это объединение данных и обрабатывающих их методов внутри одного класса. Инкапсуляция позволяет изолировать данные и методы от остальной части программы, препятствуя их случайному повреждению.

Наследование– это получение объектами (объекты-наследники) свойств, методов и переменных от других объектов (объекты-предки). Наследование позволяет получать многократно используемый код.

Полиморфизм– это возможность использовать версии одного и того же метода применительно к различным объектам.

В процедурных языках программирования последовательность выполнения строк программы всегда была строго задана. Линейность выполнения операторов нарушалась лишь операторами циклов и условной передачи управления. Такие языки не давали прямой возможности обрабатывать, например, перемещение указателя мыши. Среда Windowsпозволяет пользователю взаимодействовать с различными элементами интерфейса в любой момент времени.

Важнейшим понятием объектно-ориентированного программирования является событие. Событие– это метод объекта, который обычно генерируется самим объектом (формой, кнопкой, текстовым полем и т. д.).

Вызов событий может происходить при взаимодействии пользователя с программой (например, при щелчке по кнопке), при самостоятельном вызове объектом собственного события (например, таймер вызывает событие Tickкаждый заданный в свойстве Intervalпромежуток времени); при вызове события средой Windows(например, сообщение о необходимости перерисовать форму) и при вызове события из программы. Именно наступление определенного события запускает процедуру и, следовательно, содержащийся в

ней код на выполнение.

У событий есть свои параметры. Нарисуйте на форме объект Labelи создайте процедуру Label1_MouseMove(). Вы увидите, что событие MouseMove()имеет два параметра: senderи e. Параметры содержат данные, относящиеся к событию. Значения параметров используются для принятия решений и управления ходом выполнения программы. Параметры событий создаются и приобретают значения с помощью системы VisualBasic.Netвсякий раз, когда запускается процедура, использующая событие.

Параметр senderуказывает на объект, который вызвал событие. Параметр eопределяет, где в действительности происходит событие (в данном случае событие MouseMove()) и является объектом, имеющим свойства, относящиеся к событию. Чтобы увидеть эти свойства, откройте процедуру Label1_MouseMove(), введите имя параметра eи поставьте точку после e. Появится выпадающий список свойств.

ПРАКТИЧЕСКИЙ РАЗДЕЛ

3.1 Описание лабораторных работ

Лабораторная работа № 1

Тема: Метолы обработки информации

Цель работы: сформировать умения использовать различные методы для решения задач обработки информации.

Содержание работы

Задание 1

Изучите приведенный ниже метод двоичного поиска элемента a[l], равного целому числу b, в упорядоченном по неубыванию (невозрастанию) массиве a, содержащем целые числа.

```
алгДВ_поиск (целтаб a[1:n], натn, целb, k)
    арг a, n, b
    резk
начцелl, m
    l:=1, m:=n, k:=0
    нцпокаl≤m
    l:=int((l+m)/2)
    выбор
    приa[l]<b: l:=l+1
    приa[l]>b: m:=l-1
    иначеk:=l,m:=0
    все
    кц
кон
```

В этом алгоритме запись int((l+m)/2) означает целую часть от величины (l+m)/2. Если элемент a[l] массива a, равный величине b будет найден, то переменной k будет присвоен его номер, в противном случае переменной k будет присвоено значение 0.

Изучите приведенный алгоритм и исполните его для следующих наборов исходных данных:

1) b=3, a=[1; 3; 5; 8; 9];

2) b=4, a=[-2; 3; 5; 9; 10].

При исполнении алгоритма продолжите заполнение следующей таблицы:

l	m	k	l≤m	a[l]<b	a[l]>b	a[l]=b
1	5	0	да	нет	да	
…	…	…	…	…	…	…

Задание 2

Под сортировкой понимают процесс переупорядочивания некоторого множества объектов с целью их размещения в заданном порядке. Это универсальный вид деятельности с точки зрения обработки данных. При упорядочивании по неубыванию после сортировки будет выполнено условие: a[1] ≤ a[2] ≤ a[3] ≤ . . . ≤ a[n]. В ходе сортировки элементы последовательности меняются местами.

Изучите приведенный ниже алгоритм, реализующий метод обменной сортировки.

```
алгОбмен (целтаб a[1:n], натn)
    арг a, n
    резa, n
начцелi, j, b
    дляi=1 доn-1
    нц
    дляj=1 доn-i
    нц
    еслиa[j]>a[j+1]
    тоb:=a[j], a[j]:=a[j+1], a[j+1] :=b
    все
    кц
    кц
кон
```

Изучите приведенный алгоритм и исполните его для следующих наборов исходных данных:n=5, a=[5; 3; 1; 8; 7].

При исполнении алгоритма продолжите заполнение следующей таблицы:

i	j	a[j]>a[j+1]	b	a[j]	a[j+1]
1	5	да	0	нет	да
…	…	…	…	…	…

Объясните, почему иногда метод обменной сортировки называют методом пузырька.

Лабораторная работа № 2

Тема: Алгоритмы в криптологии

Цель работы: сформировать умения шифровать информацию с помощью простых алгоритмов шифрования.

Содержание работы

Задание 1

Зашифруйте текст MICROSOFTOFFICE, используя шифр простой подстановки при k=5.

Задание 2

Отождествите каждую букву текст БЕЗОПАСНОСТЬ ИНФОРМАЦИИ с ее порядковым номером в русском алфавите, начиная с нуля. Будем предполагать, что буква «ё» отсутствует, а пробел имеет номер 32. Для шифрования приведенного текста используйте модулярный шифр $E_a(p)=(ap+k)\text{mod}33$. В качестве ключа возьмите $a=7$, $k=11$.

Задание 3

Выполните шифрование текста EASYREPLACEMENT с помощью модулярного шифра $E_7(p)=(7\cdot p+k)\text{mod}26$.

Задание 4

Объясните идею гомофонического шифрования. Составьте таблицу с гомофонией для текста FREQUENCYPROTECTION, используя таблицу частот английских букв в процентах, и зашифруйте этот текст.

Задание 5

Используя шифр Вернама, зашифруйте текст UNIVERSITY. В качестве ключа возьмите текст CRYPTOLOGY. Преобразование исходного текста и ключа в двоичную строку выполните следующим образом. Отождествите каждую букву текста и ключа с ее порядковым номером в английском алфавите. Нумерацию букв начните с цифры 0. Каждую букву представьте пятью двоичными разрядами, соответствующими номеру буквы в алфавите (A→ 00000, B→ 00001, …, Z → 11000).

Задание 6

Используя приведенный ниже квадрат, выполните биграммное шифрование текста INFORMATIONSECURITY.

Z	Y	M	X	W
U	L	D	V	N
F	T	P	E	O
G	S	B	I	H
C	R	Q	K	A

Задание 7

Зашифруйте текст БЕЛАРУСЬ методом Вернама. В качестве ключа используйте текст СТОЛИЦА. Для превращения текста в двоичную строку используйте подход, описанный в предыдущей лабораторной работе. При представлении букв Ь и Ъ используйте один числовой код. Буквы русского алфавита при формировании двоичной строки представляйте пятью двоичными разрядами следующим образом:(А→ 00000, Б → 00001, …, Я → 11111).

Лабораторная работа № 3

Тема: Алгоритмы в базах данных

*Цель работы***:** сформировать умения выполнять простые алгоритмы для преобразования ключа записи в адрес памяти базы данных.

Содержание работы

Задание 1

Преобразуйте следующие ключи записей в четырехзначные относительные адреса пакета методом средних квадратов:

а) 145610, б) 743118, в) 910261.

Для вычислений используйте электронную таблицу MicrosoftExcel.

Задание 2

В электронной таблице MicrosoftExcelнайдите все возможные делители для хеширования методом деления в диапазоне 9971÷9999. Делителем в указанном методе может быть либо простое число, либо число, не имеющее небольших сомножителей. Преобразуйте методом деления восьмизначные ключи записей в четырехзначные относительные адреса:

а) 31465105, б) 24371180, в) 56910217.

В качестве делителя возьмите наибольшее простое число из приведенного диапазона.

Задание 3

Методом сдвига разрядов преобразуйте в четырехзначные относительные адреса следующие восьмизначные ключи записей:

а) 53146510, б) 80243711, в) 91021756.

Задание 4

Используя метод складывания, преобразуйте в трехзначные относительные адреса следующие семизначные ключи записей:

а) 4316510, б) 7243118, в) 1569021.

Задание 5

Для получения четырехзначных относительных адресов из шестизначных ключей записей

а) 145610, б) 743118, в) 910261

воспользуйтесь методом преобразования системы счисления. В качестве основания системы счисления возьмите число 13.

Задание 6

Даны семизначные ключи записей:

а) 4316510, б) 7243118, в) 1569021.

Преобразуйте их в трехзначные относительные адреса методом Лина. При преобразовании возьмите следующие параметры: q=313, m=2.

Лабораторная работа № 4

Тема: Основы языка JavaScript

*Цель работы***:** сформировать умения выполнять простые алгоритмы на языке JavaScript.

Содержание работы

Задание 1

Для каждой инструкции, приведенного ниже кода JavaScript, с правой стороны напишите комментарий после двойного слеша //, описывающий результат выполнения инструкции. В этом коде метод alert() выводит на экран модальное окно с данными, указанными в скобках. Модальное окно означает, что выполнение сценария и дальнейшее взаимодействие со страницей приостанавливается до тех пор, пока не закроется данное окно, например, нажатием кнопки OK для продолжения работы.

```
letarr = [1, 2, 3, 4, 5];
arr.length = 2;
alert( arr );
```

```
arr.length = 5
alert( arr[3] );
```

Задание 2

Напишите, что будет выведено на экран после выполнения следующего кода JavaScript. Оператор + соединяет текстовые данные, записанные в скобках метода alert().

```
varnum = 2;
if (num == 1) {
alert("значениеnum: " + num);
}
else {
if (num == 2) {
alert("значениеnum: " + num);
}
else {
alert("Не знаю такого числа!");
}
}
```

Задание 3

Напишите, что будет выведено на экран после выполнения следующего кода JavaScript.

```
vari = 0;
while (i< 3) {
alert("i: " + i);
i++;
}
```

Задание 4

Напишите, что будет выведено на web-страницу после выполнения следующего кода JavaScript. Метод document.write(arg1,arg2,arg3,...) выводит на страницу переданные ему аргументы (аргументы записаны в скобках).

```
<html>
<body>
<script>
  var count = 0;
  do {
document.write(count + " ");
count++;
} while(count < 5);
</script>
</body>
</html>
```

Задание 5

Напишите, что будет выведено на web-страницу после выполнения ниже приведенного кода JavaScript. В нем тег *
* устанавливает перевод строки на web-странице в том месте, где этот тег находится.

```
<html>
<body>
<script>
for (var i = 1, j = 5; i<= 5; i++, j--)
document.write(i + " " + j +"<br>");
</script>
</body>
</html>
```

Лабораторная работа № 5

Тема: Управляющие конструкции языка JavaScript

*Цель работы***:** сформировать умения разрабатывать программы на языке JavaScript суправляющими конструкциями.

Содержание работы

Задание 1

Сформулируйте условие задачи, которую решает следующий JavaScript код:

```
<html>
<body>
<script>
var lang = 'ru';
if (lang == 'ru') {
    var arr = ['пн', 'вт', 'ср', 'чт', 'пт', 'сб', 'вс'];
}
if(lang =='en'){
    arr=['mn','ts','wd','th','fr','st','sn'];
}
alert(arr);
</script>
</body>
</html>
```

Напишите в комментариях к коду, что будет выведено на страницу.

Задание 2

Убедитесь в том, что приведенный ниже JavaScript код даст такой же результат, что и JavaScript код в задании 1.

```
<html>
<body>
<script>
var lang ='ru';
vararr={
        'ru':['пн','вт','ср','чт','пт','сб','вс'],
        'en':['mn','ts','wd','th','fr','st','sn'],
};
alert(arr[lang]);
</script>
</body>
</html>
```

Напишите комментарии к данному JavaScript коду.

Задание 3

Выполните следующий JavaScript код и поясните в нем назначение тега br. Как выглядел бы результат, если бы в методе document.write(i +'
')отсутствовал тегbr?

```
<html>
<body>
<script>
vari=1;
while(i<=50){
        document.write(i+'<br>');
        i++;
}
</script>
</body>
</html>
```

Задание4

Дан массивarrс элементами[1, 2, 3, 4, 5]. С помощью циклаforвыведите все эти элементы на экран. Для определения количества элементов в массиве используйтесвойствоlengthв следующем виде:arr.length.К элементам массива обратитесь так:arr[i].

Задание 5

Дан массивarrс элементами[2, 3, 4, 5].С помощью циклаforнайдите произведение элементов этого массива. Результат запишите в переменнуюresult.Значение этой переменно выведите на экран.

Задание 6

Дан массив arr с элементами [2, 5, 9, 15, 0, 4]. С помощью цикла for и оператора if выведите на экран столбец тех элементов массива, которые больше 3-х, но меньше 10.

Задание 7

Дано число n=1000. Делите его на 2 столько раз, пока результат деления не станет меньше 50. Какое число получится? Посчитайте количество итераций, необходимых для этого (итерация – это проход цикла), и запишите его в переменную num.

Задание 8

Даны переменные a и b. Проверьте, что a делится без остатка на b. Если это так – выведите 'Делится' и результат деления, иначе выведите 'Делится с остатком' и остаток от деления.

Лабораторная работа № 6

Тема: Работа с объектами

*Цель работы***:** сформировать умения разрабатывать программы с объектами на языке JavaScript.

Содержание работы

Задание 1

Напишите JavaScript код, выполнив задание из каждого пункта отдельной строкой:

1. Создайте пустой объект user.
2. Добавьте свойство name со значением John.
3. Добавьте свойство surname со значением Smith.
4. Измените значение свойства name на Pete.
5. Удалите свойство name из объекта.

С помощью цикла for-in выведите на экран ключи и элементы этого объекта.

Задание 2

Выполните приведенный ниже JavaScript код и сформулируйте задачу, которую он решает. Напишите комментарии к коду.

```
<html>
<body>
<script>
varobj = {
        'Минск': 'Беларусь',
        'Москва': 'Россия',
        'Киев':'Украина'
};

for(var key in obj){
        alert(key +' - это '+ obj[key]+'.');
}
</script>
</body>
</html>
```

Задание 3

Дан объект obj:

```
varobj={green:'зеленый',red:'красный',blue:'голубой'}
```

С помощью цикла for-in выведите на экран ключи и элементы этого объекта.

Задание 4

Выполните вначале нижеприведенныйJavaScript код:

```
letcodes = {
  "49": "Германия",
  "41": "Швейцария",
  "44": "Великобритания",
"1": "США"
};
for (let code in codes) {
alert(code);
},
```

а затем следующий JavaScript код:

```
letcodes = {
  "+49": "Германия",
```

```
  "+41": "Швейцария",
  "+44": "Великобритания",
"+1": "США"
};
for (let code in codes) {
alert(+code);
}
```

Выясните, чем отличаются выведенные на экран результаты приведенных кодов, и объсните почему.

Задание 5

Выполните приведенный ниже JavaScript код и сформулируйте задачу, которую он решает. Напишите комментарии к коду.

```
letuser = {
name: "John",
surname: "Smith"
};
user.age = 25;
for (let prop in user) {
alert( prop );
}
```

Лабораторная работа № 7

Тема: Работа в среде VisualBasic

*Цель работы***:** сформировать умениеработать в среде VisualBasic.

Содержание работы

Задание 1

Напишите программу перевода градусов по шкале Цельсия в градусы по шкале Фаренгейта. Разместите на форме два объекта TextBox1и TextBox2. Поле TextBox1 используйте для ввода значения температуры, выраженной в градусах по шкале Цельсия, а по щелчку в поле TextBox2 надо получить результат пересчета в градусы по шкале Фаренгейта. Расчетная формула перевода из градусов Цельсия в градусы Фаренгейта: f=c*9/5+32, где c – переменная для градусов по шкале Цельсия, а f – переменная для градусов по шкале Фаренгейта.

Задание 2

Наберите приведенную ниже программу, несколько раз ее выполнитеи сформулируйте задачу, которую она решает.

```
Private Sub TextBox3_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles TextBox3.Click
TextBox3.Text = CStr(Convert.ToInt16(TextBox1.Text) *
Convert.ToInt16(TextBox2.Text))
End Sub
```

Укажите недостаток, которым она обладает.Поясните, как его можно устранить.

Задание 3

Объясните, какой недостаток программы предыдущего задания устранен следующей программой, и каким образом.

```
Private Sub TextBox3_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles TextBox3.Click
    Try
        Convert.ToInt16(TextBox1.Text)
    Catch ex As Exception
        MessageBox.Show("Вполевведенонечисло")
        TextBox1.SelectAll()
        TextBox1.Focus() Exit Sub
    End Try
```

```
        Try
            Convert.ToInt16(TextBox2.Text)
        Catch ex As Exception
            MessageBox.Show("Вполевведенонечисло")
            TextBox2.SelectAll()
            TextBox2.Focus() Exit Sub
        End Try
        TextBox3.Text = CStr(Convert.ToInt16(TextBox1.Text) *
        Convert.ToInt16(TextBox2.Text))
    End Sub
```

Задание 4

Наберите приведенную ниже программу, выполните ее и сформулируйте задачу, которую она решает.

```
    Private Sub Form1_Load(ByVal sender As Object, ByVal e As
    System.EventArgs) Handles MyBase.Load
        Timer1.Enabled = True
            Timer1.Interval = 600
        End Sub
    Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e
    As System.EventArgs) Handles
    Timer1.Tick
            Dim t As New DateTime
            t = Now
            TextBox1.Text = t.ToLongTimeString
        End Sub
```

Лабораторная работа № 8

Тема: Программный код (4 часа)

Цель работы: сформировать умения разрабатывать программы на языке VisualBasic.

Содержание работы

Задание1

Класс PictureBox позволяет получить элемент управления PictureBox (графическое поле), которое используется для вывода растровых изображений форматов GIF, JPEG, PNG и BMP. Свойство Image позволяет загрузить изображение в графическое поле. Свойство SizeMode позволяет подогнать

изображение под размер графического поля, как в режиме конструирования формы, так и в режиме выполнения программы.

Загрузите изображение в графическое поле в режиме конструирования. Для этого выполните следующее.Расположите на форме объект PictureBoxl. Щелкнув в окне свойств на кнопке конструктора (на кнопке нарисованы три точки) в строке свойства Image, откройте диалоговое окноОткрыть, с помощью которого выберите нужный графический файл, например, Мой_компьютер/Общие_документы/Рисунки (общие)/Образцы_рисунков-/Закат. Чтобы подогнать размер изображения под размер PictureBoxl, установите значение свойства SizeModeравным Stretchimage.

Задание2

Найдите код ASCIIдля символа #. Для этого выполните следующие действия:

Разместите на форме объект Label1. В окне редактора кода введите следующий текст:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
    Label1.Text = Asc("#")
EndSub
```

Нажмите функциональную клавишу F5 и вы получите искомый результат.

Задание3

Выясните, что будет выведено в окне формы после выполнения следующих действий:

– Нарисуйте на форме объект Label1.

– В окне редактора кода откройте процедуру Label1_Click()и запишите следующую строчку:

```
Private Sub Label1_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Label1.Click Label1.Text = Cint(9.83)
EndSub
```

– Нажмите функциональную кнопку F5.

– В появившемся окне формы щелкните на объекте Label1.

Задание4

Создайте на форме объект Label1.

В окне редактора кода наберите программу:

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
```

```
System.EventArgs) Handles MyBase.Load
        Label1.Text = Len("Алгоритмы обработки данных")
    EndSub
```

Нажмите функциональную кнопку F5. В появившемся окне формы щелкните на объекте Label1.

Что будет выведено в окне формы?

Задание5

Наберите приведенную ниже программу и выполните ее.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load
Dim m As DialogResult m = MessageBox.Show("Вырешилипрерватьпроцесс?", "Вашедействие", MessageYesNo, MessageBoxicon.Question)
        EndSub
```

Объясните результат работы программы.

Задание6

На форме расположены два объекта Label1и Label2. Написать программу, реализующую следующий алгоритм. По щелчку на Labellпроисходит изменения свойств для Label1и Label2. У объекта Label1изменяется цвет фона на красный, появляется надпись "Желтый", написанная желтым цветом шрифтом TimesNewRoman, 14 пунктов курсивом. У объекта Label2изменяется цвет фона на желтый, появляется надпись "Красный", написанная красным цветом шрифтом TimesNewRoman, 14 пунктов с подчеркиванием. По щелчку на Label2объекты меняются значениями свойств.

3.2 Практические работы

Практическая работа 1

Алгоритмический язык

Цель работы: сформировать умения разрабатывать и записывать алгоритмы на алгоритмическояэыке.

Содержание работы

Задача 1

1. Исполните приведенный ниже алгоритм, записанный на алгоритмическом языке, для следующих исходных данных: n=6, a = (5,-1, 2, 0, 4, -3).

```
алг Пр2 (целтаба[1:n], целn)
арга,n
резс
начцелi,m,p
i:=1, m:=0, p:=0
пока i<=n
нц
если a[i]<0
тоm:=m+1
все
если a[i]>0
тор:=p+1
все
i:=i+1
кц
еслир>m
тос:= 'Положительных элементов больше'
иначееслир<m
тос:= 'Отрицательных элементов больше'
иначес:= 'Одинаковое количество'
все
все
кон
```

Для этого заполните следующую таблицу:

i	m	p	i<=n	c
…	…	…	…	…

2. Сформулируйте условие задачи, которую решает данный алгоритм.

3. Представьте этот алгоритм в виде блок-схемы.

Задача 2

Дана квадратная матрица a размерности n×n, элементы которой целые числа. Разработайте алгоритм, который будет вычислять в ней количество нулевых элементов z.

1. Запишите алгоритм на алгоритмическом языке.

2. Исполните алгоритм для следующей матрицы:

$$\begin{vmatrix} 1 & -2 & 4 \\ 3 & 0 & 2 \\ 0 & 4 & 1 \end{vmatrix}$$

Вычислите характеристики времени и памяти для разработанного вами алгоритма.

Практическая работа 2

Типы алгоритмов

Цель работы: сформировать умения разрабатывать и записывать алгоритмы разных типов на алгоритмическо языке.

Содержание работы

Задача 1

Изучите приведенный ниже алгоритм.

```
алг Число (целn,F)
аргn
резF
начцелF1,F2
еслиn=1
тоF:=1
иначеF1:=0; F2:=1; F:=1
      покаF<=n
      нц
      F:=F1+F2; F1:=F2; F2:=F
      кц
      F:=F1
   все
кон
```

Исполните этот алгоритм для всех значений n от 1 до 10. Для этого заполните следующую таблицу.

n=1	F	F1	F2	F<=n
…	…	…	…	…

Сформулируйте условие задачи, которую решает этот алгоритм.

Задача 2

Изучите приведенный ниже алгоритм.

```
алгКоличество (целтаба[1:n], b[1:m], k[1:m], целn,m)
   арга, n
   резb, k, m
```

```
нач цел i, l
    m:=0
    для l от 1 до n
    нц
    b[m+1]:=a[i], l=1
    пока b[l]≠a[i]
    нц
        l:=l+1
    кц
если l=m+1
то m:=m+1, k[m]:=1
иначе k[l]:=k[l]+1
    все
кон
```

Исполните этот алгоритм для следующих исходных данных: n=6, a=[3; 2; 3; 5; 5; 3]. Для этого заполните следующую таблицу.

m	i	b[m+1]	l	b[l]≠a[i]	l=m+1	k[m]	k[l]
…	…	…	…	…	…	…	…

Сформулируйте условие задачи, которую решает этот алгоритм.

Практическая работа 3

Введение в терию сложности алгоритмов

Цель работы: сформировать умения вычислять характеристики времени и памяти алгоритмов.

Содержание работы

Задача 1

Разработайте алгоритм, который будет формировать массив a размерности

n, содержащий нечетные числа 1,3, …, 2n-1.Алгоритм запишите на алгоритмическом языке и представьте его в виде блок-схемы. Вычислите характеристики времени и памяти для разработанного вами алгоритма. Исполните алгоритм для n, равного 4.

Задача 2

Дан массив a, содержащий n целых чисел. Найдите в нем первый по порядку отрицательный элемент и укажите его номер. Алгоритм запишите на алгоритмическом языке и представьте его в виде блок-схемы. Вычислите характеристики времени и памяти для разработанного вами алгоритма. Исполните алгоритм для n, равного 5.

Задача 3

Дана матрица a размерности (n×m), элементы которой целые числа. Разработайте алгоритм, который будет заменять в ней нули на число 1. Алгоритм запишите на алгоритмическом языке и представьте его в виде блок-схемы.

Вычислите характеристики времени и памяти для разработанного вами алгоритма. Исполните алгоритм для следующей матрицы:

$$\begin{vmatrix} 1 & -2 & 4 \\ 3 & 0 & 2 \\ 0 & 4 & 1 \end{vmatrix}$$

Задача 4

Даны две матрицы a и b размерности (n×m). Требуется найти матрицу c, которая является суммой матриц a и b. Алгоритм запишите на алгоритмическом языке и представьте его в виде блок-схемы. Вычислите характеристики времени и памяти для разработанного вами алгоритма. Подберите матрицы a и b размерности (2×3) и исполните алгоритм.

Практическая работа 4

Алгоритмы в криптологии

Цель работы: сформировать умения шифровать информацию с помощью биграммнойкриптосхемы и транспозиционных подходов.

Содержание работы

Задача 1

Используйте биграммнуюкриптосхему Хилла для шифрования текста DEFINITION. В качестве ключа возьмите квадратную матрицу

$$M = \begin{bmatrix} 2 & 5 \\ 3 & 3 \end{bmatrix}.$$

Покажите, что данная матрица *M*является обратимой по модулю 26.

Задача 2

Найдите еще одну квадратную матрицу размерности 2×2, которая будет обратимой по модулю 26.

Задача 3.

Покажите, что квадратные матрицы *A*и *B*размерности 2×2 являются обратимыми по модулю 26:

$$A = \begin{bmatrix} 1 & 24 \\ 24 & 5 \end{bmatrix}, B = \begin{bmatrix} 5 & 2 \\ 2 & 1 \end{bmatrix}.$$

Задача 4

Выясните, является ли квадратная матрица размерности3×3

$$M = \begin{bmatrix} 14 & 8 & 3 \\ 8 & 5 & 2 \\ 3 & 2 & 1 \end{bmatrix}$$

обратимой по модулю 26.

Задача 5

Распространите метод шифрования биграмм Хилла на шифрование триад символов и зашифруйте текст PASCAL. Для шифрования возьмите квадратную матрицу размерности 3×3

$$M = \begin{bmatrix} 1 & 24 & 1 \\ 24 & 5 & 22 \\ 1 & 22 & 6 \end{bmatrix},$$

которая является обратимой по модулю 26.

Задача 6

Дан текст ПОДСТАНОВОЧНОЕ ШИФРОВАНИЕ. Для его шифрования используйте шифр Вижинера. С этой целью текст разбейте на блоки по пять символов в каждом. Пусть последовательность букв k1, k2, …, k5 слова ШРИФТ является ключом. Отождествите каждую букву исходного текста и ключа с номером буквы в алфавите русского языка так, как это сказано в задании 2 лабораторной работы 1. При шифровании используйте следующиефункции fi(a) = (a+ki)mod33, гдеa есть буква i-го блока.

Задача 7

Зашифруйте текст PERMUTATIONENCRYPTION, используя два подхода транспозиционного шифрования.

3.3 Семинарские занятия

Раздел 1.Основы алгоритмизации

Тема 1.Технологии проектирования алгоритмов (4 часа)

*Цель:*сформировать представление о различных технологиях проектирования и разработки алгоритмов.

Вопросы:

1. Структурное программирование.
2. Нисходящее и восходящее проектирование алгоритмов.
3. Метод пошаговой детализации алгоритма.
4. Алгоритмическое программирование
5. Процедурное программирование.
6. Объектно-ориентированное программирование.
7. Визуальное программирование
8. Логическое программирование.
9. Функциональное программирование.
10. Прототипное программирование.
11. Аспектно-ориентированное программирование.
12. Компонентно-ориентированное программирование.

Краткая теория

Структурное программирование

Значительное увеличение сложности задач, решаемых с помощью ЭВМ, приводит к увеличению размеров и сложности программ, что порождает дополнительные трудности при их разработке и отладке. Увеличение продолжительности жизненного цикла программ приводит к тому, что с течением времени из-за изменения условий использования программ возникает необходимость их модификации, повышения их эффективности, удобства пользования ими.

Для разрешения возникших при этом проблем в практике программирования выработан ряд приемов и методов, которые принято называть методами структурного программирования.

Под структурным программированием понимают такие методы разработки и записи программы, которые ориентированы на максимальные удобства для восприятия и понимания ее человеком. При прочтении программы в ее следующих друг за другом фрагментах должна четко прослеживаться

логика ее работы, т. е. не должно быть "скачков" на фрагменты программы, расположенные где-то в другом месте программы.

Структурное программирование – "программирование без goto", т. е. не используются операторы перехода без особой необходимости. В связи с этим отдельные фрагменты программы представляют собой некоторые логические (управляющие) структуры, которые определяют порядок выполнения содержащихся в них правил обработки данных. Любая программа получается построенной из стандартных логических структур, число типов которых невелико.

При использовании технологии структурного программирования сначала пишется текст основной программы, в котором, вместо каждого связного логического фрагмента текста, вставляется вызов подпрограммы, которая будет выполнять этот фрагмент. Вместо настоящих, работающих подпрограмм, в программу вставляются фиктивные части – заглушки, которые, говоря упрощенно, ничего не делают.

Если говорить точнее, заглушка удовлетворяет требованиям интерфейса заменяемого фрагмента (модуля), но не выполняет его функций или выполняет их частично. Затем заглушки заменяются или дорабатываются до настоящих полнофункциональных фрагментов (модулей) в соответствии с планом программирования. На каждой стадии процесса реализации уже созданная программа должна правильно работать по отношению к более низкому уровню. Полученная программа проверяется и отлаживается.

После того, какпрограммистубедится, что подпрограммы вызываются в правильной последовательности (то есть общая структура программы верна), подпрограммы-заглушки последовательно заменяются на реально работающие, причём разработка каждой подпрограммы ведётся тем же методом, что и основной программы. Разработка заканчивается тогда, когда не останется ни одной заглушки.

Такая последовательность гарантирует, что на каждом этапе разработки программист одновременно имеет дело с обозримым и понятным ему множеством фрагментов, и может быть уверен, что общая структура всех более высоких уровней программы верна.

При сопровождении и внесении изменений в программу выясняется, в какие именно процедуры нужно внести изменения. Они вносятся, не затрагивая части программы, непосредственно не связанные с ними. Это позволяет гарантировать, что при внесении изменений и исправлении ошибок не выйдет из строя какая-то часть программы, находящаяся в данный момент вне зоны внимания программиста.

Следует также учесть, что в «Предисловии» к книге «Структурное программирование»Тони Хоаротмечает, что принципы структурного программирования в равной степени могут применяться при разработке программ как «сверху вниз», так и «снизувверх».

Алгоритмическое программирование

В 1950-х годах появились языки, для написания программ в которых использовались общеупотребительные слова и простые правила синтаксиса. Перевод таких программ на машинные коды производился специальной программой-компилятором, поэтому программиста уже не волновало, в каких ячейках хранятся результаты расчета и какие регистры процессора используются для операций. Программист был сосредоточен на записи разработанного им алгоритма на алгоритмическом языке программирования. Таков, например, язык ФОРТРАН (Fortran – переводчик формул), разработанный для инженерных расчетов, или КОБОЛ для экономических расчетов. Во главу угла было поставлено понятие алгоритма, который, как известно, является последовательностью однозначно определенных действий, приводящих за конечное число шагов к результату.

Процедурное программирование

Было замечено, что в программах встречаются одинаковые группы операторов, отличающиеся только значениями входящих в ни параметров. Повторяющиеся блоки операторов стали стандартизировать, то есть выделять из общей программы в отдельные подпрограммы, процедуры и функции. В результате возникли процедурные языки программирования, как Паскаль, С. Это – универсальные языки. Они не ориентированы на решение конкретного типа задач, т. е. не были проблемно-ориентированными, как, например, Фортран.

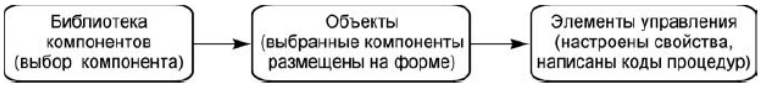
Объектно-ориентированное программирование

В середине 1980-х годов был введен принцип повторного использования кода ранее написанных программ. Такие готовые программные блоки назвали объектами. При создании новой программы объекты просто перенастраиваются в соответствии с требованиями решаемой программистом задачи. Таковы встречающиеся в разных программах одинаковые по форме окна, командные кнопки, списки, похожие меню, одинаковые шрифты и т. Д. Все они являются настраиваемыми объектами. Примерами объектно-ориентированных языков являются, например, языки C++ и ObjectPascal.

Визуальное программирование

С разработкой в середине 1990-х годов операционной системы Windows компьютеры приобрели возможность графического управления. На экране

монитора размещаются уже не просто картинки, а графические элементы управления, реагирующие на определенные действия, в частности, на действия со стороны пользователя (на события). Щелчком мыши по графическому элементу можно запустить на выполнение целую программу, не пользуясь при этом клавиатурой. Появление графических (визуальных) систем управления сделало программирование также визуальным. Теперь можно с помощью мыши



выбирать из библиотек компоненты, размещать их в рабочем окне будущей программы (делая таким образом из компонентов объекты), и настраивать объекты с помощью свойств, добавляя процедуры для обработки событий (создавая таким образом из объектов графические элементы управления). На рисунке приведена схема «превращения» компонента через объект в элемент управления.

Можно сказать, что компонент сродни чертежу, по которому изготавливают множество совершенно одинаковых объектов, например, одинаковых квартир. Люди, поселившись в квартире, обустраивают ее в соответствии со своими потребностями и вкусом, делают квартиру уникальным элементом своего образ жизни.

При подготовке к семинарским занятиям рекомендуется:

1. Подготовить краткий конспект по каждому вопросу семинарского занятия.

2. Подготовить интерактивную презентацию по любому из вопросов.

4 РАЗДЕЛ КОНТРОЛЯ ЗНАНИЙ

4.1 Задания для контролируемой самостоятельной работы студентов

Самостоятельная работа студентов направлена на совершенствование их умений и навыков по дисциплине «Алгоритмизация и программирование». Цель самостоятельной работы студентов – способствование усвоению в полном объеме учебного материала дисциплины через систематизацию, планирование и контроль собственной деятельности. Преподаватель дает задания по самостоятельной работе и регулярно проверяет их исполнение.

Вариант 1

1. Разработайте алгоритм для вычисления следующей функции:

$$y = 2 \mid 3x+1 \mid - \mid 5 - 2x \mid + 3.$$

Алгоритм должен быть записан без использования функции модуль.

а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для трех точек, принадлежащих различным интервалам числовой прямой.

2. Разработайте алгоритм для вычисления первых *n*слагаемых следующей суммы ряда:

$$S = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots .$$

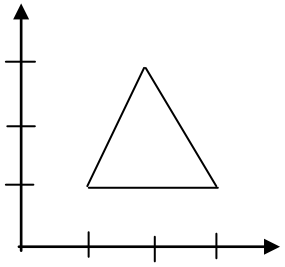
Переменная *x*– вещественная.

а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для $n = 4$.

3. Даны n точек с координатами (x_1; y_1), (x_2; y_2), …, (x_n; y_n). Координаты точек – вещественные числа. Разработайте алгоритм для подсчета количества точек, попавших в фигуру, изображенную на рисунке, включая ее границу.



а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для четырех точек с координатами (1; 1), (2; 2), (3;1), (1;3).

4. Дана матрица a размерности *n*x*m*, элементы которой целые числа. Разработайте алгоритм, который будет заменять в ней положительные элементы на нули.

а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для следующей матрицы:

$$\begin{vmatrix} -2 & 0 & 3 & -4 \\ 1 & -4 & 2 & 0 \\ 3 & 1 & 0 & -6 \end{vmatrix}.$$

Вариант 2

1. Разработайте алгоритм для вычисления следующей функции:

$$y = |4x+2| - 2\,|3 - x| + 5.$$

Алгоритм должен быть записан без использования функции модуль.

а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для трех точек, принадлежащих различным интервалам числовой прямой.

2. Разработайте алгоритм для вычисления первых *n*слагаемых следующей суммы ряда:

$$S = x + \frac{x^3}{3 \cdot 4} + \frac{x^5}{5 \cdot 6} + \frac{x^7}{7 \cdot 8} + \ldots .$$

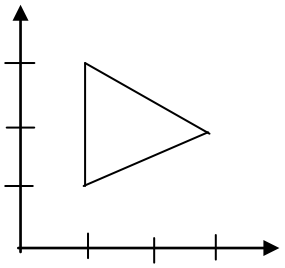
Переменная *x*– вещественная.

а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для $n = 4$.

3. Даны *n* точек с координатами (x_1; y_1), (x_2; y_2), …, (x_n; y_n). Координаты точек – вещественные числа. Разработайте алгоритм для подсчета количества точек, попавших в фигуру, изображенную на рисунке, включая ее границу.



а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для четырех точек с координатами (1; 1), (2; 2), (3;1), (1;3).

4. Дана матрица a размерности *n*x*m*, элементы которой целые числа. Разработайте алгоритм, который будет заменять в ней отрицательные элементы на нули.

а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для следующей матрицы:

$$\begin{vmatrix} -2 & 0 & 3 & -4 \\ 1 & -4 & 2 & 0 \\ 3 & 1 & 0 & -6 \end{vmatrix}.$$

Вариант 3

1. Разработайте алгоритм для вычисления следующей функции:

$$y = |\, 4x+3| - 3 \,|\, 2x - 1 \,| - 3.$$

Алгоритм должен быть записан без использования функции модуль.

а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для трех точек, принадлежащих различным интервалам числовой прямой.

2. Разработайте алгоритм для вычисления первых *n*слагаемых следующей суммы ряда:

$$S = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \ldots \ .$$

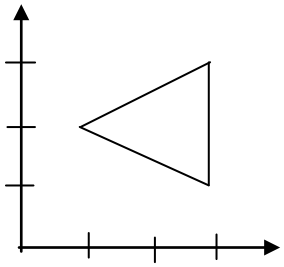
Переменная *x*– вещественная.

а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для $n = 4$.

3. Даны n точек с координатами (x_1; y_1), (x_2; y_2), …, (x_n; y_n). Координаты точек – вещественные числа. Разработайте алгоритм для подсчета количества точек, попавших в фигуру, изображенную на рисунке, включая ее границу.



а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для четырех точек с координатами (1; 1), (2; 2), (3;1), (1;3).

4. Дана матрица a размерности nxm, элементы которой целые числа. Разработайте алгоритм, который будет заменять в ней нули на число 1.

а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для следующей матрицы:

$$\begin{vmatrix} -2 & 0 & 3 & -4 \\ 1 & -4 & 2 & 0 \\ 3 & 1 & 0 & -6 \end{vmatrix}.$$

Вариант 4

1. Разработайте алгоритм для вычисления следующей функции:

$$y = | 5 - 2x | - 2 | x - 3 | + 2.$$

Алгоритм должен быть записан без использования функции модуль.

а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для трех точек, принадлежащих различным интервалам числовой прямой.

2. Разработайте алгоритм для вычисления первых nслагаемых следующей суммы ряда:

$$S = \frac{x^2}{2!} - \frac{x^4}{4!} + \frac{x^6}{6!} - \frac{x^8}{8!} + \ldots .$$

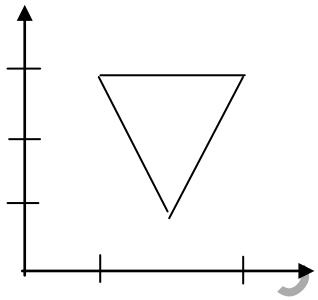
Переменная *x*– вещественная.

а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для $n = 4$.

3. Даны n точек с координатами (x1; y1), (x2; y2), …, (xn; yn). Координаты точек – вещественные числа. Разработайте алгоритм для подсчета количества точек, попавших в фигуру, изображенную на рисунке, включая ее границу.



а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для четырех точек с координатами (1; 1), (2; 2), (3;1), (1;3).

4. Дана матрица a размерности *n*x*m*, элементы которой целые числа. Разработайте алгоритм, который будет заменять в ней элементы, равные единице, на нули.

а) Запишите алгоритм на алгоритмическом языке.

б) Представьте алгоритм в виде блок-схемы.

в) Исполните алгоритм для следующей матрицы:

$$\begin{vmatrix} -2 & 0 & 3 & -4 \\ 1 & -4 & 2 & 0 \\ 3 & 1 & 0 & -6 \end{vmatrix}.$$

4.2 Перечень вопросов к экзамену

1. Понятие алгоритма. Свойства алгоритма.
2. Типы алгоритмических структур: следование, разветвление и цикл.
3. Основная теорема структурного программирования.
4. Формы записи алгоритмов.
5. Общий вид алгоритма на алгоритмическом языке.
6. Обработка строковых данных на алгоритмическом языке.
7. Оператор цикла для в алгоритмическом языке.
8. Оператор цикла пока в алгоритмическом языке.
9. Оператор цикла если в алгоритмическом языке.
10. Исполнение алгоритма. Пример.
11. Задача о коне на шахматной доске.
12. Быстрый последовательный поиск.
13. Двоичный поиск.
14. Алгоритм нахождения из трех чисел наименьшего.
15. Последовательный поиск.
16. Алгоритм нахождения n-го числа Фибоначчи.
17. Алгоритм нахождения n!.
18. Алгоритмический язык как вычислительная машина.
19. Характеристики времени и памяти алгоритмов.
20. Единицы измерения времени и памяти алгоритма.
21. Оценка алгоритма. Размерность задачи. Пример.
22. JavaScript. Модальные окна.
23. Методы в JavaScript. Метод document.write().
24. Методы в JavaScript. Метод prompt().
25. Методы в JavaScript. Метод alert().
26. JavaScript. Способы записи чисел.
27. JavaScript. Массивы.
28. JavaScript. Инструкция цикла while.
29. JavaScript. Инструкция разветвления if.
30. JavaScript. Инструкцияцикла do-while.
31. JavaScript. Инструкция цикла for.
32. JavaScript.Работа с объектами.
33. JavaScript. Операторы присваивания.
34. JavaScript. Операторы сравнения.
35. JavaScript. Арифметические операторы.

36. JavaScript. Бинарные операторы.
37. JavaScript. Логические операторы.
38. JavaScript. Строковые операторы.
39. Шифр простой подстановки. Модулярный шифр.
40. Гомофоническое шифрование. Пример.
41. Полиграммное и биграммное шифрование.
42. Криптосхема Хилла. Шифр Плейфера.
43. Шифр Виженера. Шифр Вернама.
44. Метод средних квадратов. Деление с остатком.
45. Сдвиг разрядов. Метод складывания.
46. Преобразование системы счисления.
47. Анализ отдельных разрядов.
48. Метод Лина.

4.3 Задачи к экзамену

Задача 1

Зашифруйте текст PROTECTION, используя шифр простой подстановки при k=5. Буквы английского алфавита занумеруйте от 0 до 25.

Задача 2

В массиве целых чисел A размерности n переставьте элементы так, чтобы вначале разместились отрицательные элементы, а затем – неотрицательные. Алгоритм представьте в виде блок-схемы.

Задача 3

Для получения четырехзначного относительного адреса из ключа записи 743118 воспользуйтесь методом преобразования системы счисления. В качестве основания системы счисления возьмите 13.

Задача 4

Что выведет на страницу следующий JavaScript код?

```
var arr = [1, 2, 3, 4, 5];
for (var i = 1; i<= arr.length-1; i++) {
  alert(arr[i]); }
```

Задача 5

Выполните шифрование текста EASYREPLACEMENT с помощью модулярного шифра $E3(p)=(3\cdot p+2) \bmod 26$.

Задача 6

Дан семизначный ключ записи 7243118. Преобразуйте его в трехзначный относительный адрес методом Лина. При преобразовании возьмите следующие параметры: q=313, m=2.

Задача 7

Объясните идею гомофонического шифрования. Составьте таблицу с гомофонией для текста MICROSOFTOFFICE, используя таблицу частот английских букв в процентах, и зашифруйте этот текст.

Задача 8

Дана строка a. Определите, сколько раз в ней встречается пробел. Алгоритм представьте в виде блок-схемы.

Задача 9

Преобразуйте ключи записи 24371 в четырехзначный относительный адрес пакета методом средних квадратов.

Задача 10

Представьте в виде блок-схемы алгоритм нахождения номера наибольшего элемента в целочисленном массиве A размерности n.

Задача 11

Дан массив целых чисел A размерности n. Запишите на алгоритмическом языке алгоритм, который будет подсчитывать в нем количество положительных элементов.

Задача 12

Что выведет на страницу следующий JavaScript код?

```
var min = 10;
if (min >= 0 && min <= 14) {
alert('В первую четверть.'); }
if (min>= 15 &&min<= 30) {
alert('Во вторую четверть.');}.
```

Задача 13

Что выведет на страницу следующий JavaScript код?

```
varobj = {Коля: 200, Вася: 300, Петя: 400};
for (key in obj) {
  alert(key); }
```

Задача 14

Используя собственный квадрат с английскими буквами размерности 5x5, выполните биграммное шифрование текста INFORMATION.

Задача 15

Используйте биграммнуюкриптосхему Хилла для шифрования слова TEXT. В качестве ключа возьмите квадратную матрицу

$$M=\begin{bmatrix} 2 & 5 \\ 3 & 3 \end{bmatrix}$$

Задача 16

Задана матрица целых чисел A размерности (n×n). Запишите на алгоритмическом языке алгоритм, который будет находить сумму её элементов на вспомогательной диагонали.

Задача 17

Что выведет на страницу следующий JavaScript код?

```
var a = ['a', 'b', 'c'];
var b = [1, 2, 3];
var c = a.concat(b);
alert(c);
```

Задача 18

Дан массив целых чисел A размерности n. Запишите на алгоритмическом языке алгоритм, который будет находить среднее значение элементов массива A.

Задача 19

Преобразуйте методом деления восьмизначный ключ записи 24371180 в четырехзначный относительный адрес. В качестве делителя возьмите число 9991.

Задача 20

Что выведет на страницу следующий JavaScript код?

```
var name = 'Страна';
alert('Привет, +name+'!');
```

Задача 21

Дана строка a. Определите, сколько она содержит слов. Предполагается, что слова в строке разделяются одним пробелом. Алгоритм представьте в виде блок-схемы.

Задача 22

Методом сдвига разрядов преобразуйте в четырехзначный относительный адрес восьмизначный ключ записи 80243711.

Задача 23

Используя метод складывания, преобразуйте в трехзначный относительный адрес семизначный ключ записи 7243118.

Задача 24

Что выведет на страницу следующий JavaScript код?

```
var arr = [1, 2, 3, 4, 5];
for (var i = 1; i<= arr.length-1; i++) {
  alert(arr[i]); }
```

4.4 Критерии оценки результатов учебной деятельности студентов

Для выявления и исключения пробелов в знаниях студентов рекомендуется использовать следующие средства:

1) фронтальный опрос на лекциях, лабораторных и семинарских занятиях;
2) критериально-ориентированные тесты для контроля теоретических знаний современных информационных настольных издательских систем, основных определений, терминологии и правил набора, редактирования, форматирования и верстки текстовой и графической информации;
3) выполнение тестовых заданий с произвольной формой ответа для контроля умения анализировать, грамотно излагать и формулировать свои соображения и выводы в данной предметной области;
4) выполнение творческих заданий, которые предполагают эвристическую деятельность и поиск неформальных решений.

5 ВСПОМОГАТЕЛЬНЫЙ РАЗДЕЛ

5.1 Учебнаяпрограмма

Гляков, П. В. Программно-технические средства.Часть2.Алгоритмы обработки ланных: учебнаяпрограмма учреждения высшего образования для специальности1-21 04 01 Культурология (по напрамлениям), направления специальности 1-21 04 01-02 Культурология (прикладная, специализации 1-21 04 01-02 04 Информационные системы в культуре / П. В. Гляков. – Минск : БГУКИ, 2021. – 16 с.

5.2 Учебно-методическая карта учебной дисциплины для дневной формы получения высшего образования

Номер раздела, темы	Название раздела, темы	Количество аудиторных часов				Количество часов УСР	Форма контроля Знаний
		Лекции	Практические занятия	Семинарские занятия	Лабораторные занятия		
Раздел 1	**Основы алгоритмизации**						
Тема 1	Понятие алгоритма и способы представления	2					
Тема 2	Алгоритмический язык		2				Отчет
Тема 3	Типы алгоритмов		2				Отчет
Тема 4	Методы обработки информации	2			2		Алгорит-мы
Тема5	Введение в теорию сложности алгоритмов	2	2				Отчет
Тема 6	Технологии проектирования алгоритмов			4		4	Реферат
Тема 7	Алгоритмы в криптологии	2	2		2		Отчет

Тема 8	Алгоритмы в базах данных	2			2		Отчет
Раздел2	**Язык программирования JavaScript**						
Тема 9	Основы языка JavaScript	2			2		Програм-мы
Тема 10	Управляющие инструкции языка JavaScript				2		Програм-мы
Тема 11	Работа с объектами				2		Програм-мы
Раздел 3	**Язык программирования VisualBasic**						
Тема 12	Работа в средеVisualBasic	2			2		Програм-мы
Тема 13	Программный код				4		Програм-мы
	Всего…	**14**	**8**	**4**	**18**	**4**	

5.3 Учебно-методическая карта учебной дисциплины для заочной формы получения высшего образования

Номер раздела, темы	Название раздела, темы	Количество аудиторных часов				Количество часов УСР	Форма контроля Знаний
		Лекции	Практические занятия	Семинарские занятия	Лабораторные занятия		
Раздел 1	**Основы алгоритмизации**						
Тема 1	Понятие алгоритма и способы представления	2					
Тема 2	Алгоритмический язык		2				Отчет
Тема 3	Типы алгоритмов		2				Отчет
Тема 4*	Методы обработки информации	2			2		Алгорит-мы

Тема5	Введение в теорию сложности алгоритмов	2	2				Отчет
Тема 6*	Технологии проектирования алгоритмов			4		4	Реферат
Тема 7	Алгоритмы в криптологии	2	2		2		Отчет
Тема 8*	Алгоритмы в базах данных	2			2		Отчет
Раздел2	**Язык программирования JavaScript**						
Тема 9	Основы языка JavaScript	2			2		Програм-мы
Тема 10	Управляющие инструкции языка JavaScript				2		Програм-мы
Тема 11*	Работа с объектами				2		Програм-мы
Раздел 3	**Язык программирования VisualBasic**						
Тема 12	Работа в средеVisualBasic	2			2		Програм-мы
Тема 13*	Программный код				4		Програм-мы
	Всего…	**14**	**8**	**4**	**18**	**4**	

Темы, помеченные звёздочкой (*), выполняются студентами в межсессионный период самостоятельно.

5.4 Список основной литературы

1. Теоретические основы информационных технологий : учеб.-метод. комплекс / Сост. : П. В. Гляков, Т. С. Жилинская, Т. И. Песецкая. – Минск : БГУКИ, 2017. – 319 с.

2. Руководство. Начало работы с VisualBasic в VisualStudio [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/visualstudio/get-started/visual-basic/tutorial-console?view=vs. – Дата доступа: 10.09.2019.

3. Гляков, П. В.Игровая форма обучения алгоритмизации культурологов-менеджеров/П. В. Гляков // Информатизация образования. – 2010. – № 3. – С. 21–27.

4. Батищев, П. С. Основы программирования на VisualBasic 6.0 : электронный учебник [Электронный ресурс]. – Режим доступа: psbatishev.narod.ru › vb. – Дата доступа: 10.09.2019.

5. МакГрат, М. Программирование на VisualBasic для начинающих / М. МакГрат. – М. : Эксмо, 2017. – 193 с.

6. Современный учебник JavaScript[Электронный ресурс]. – Режим доступа: learn.javascript.ru. – Дата доступа: 16.09.2020.

7. JavaScript : онлайн учебник [Электронный ресурс]. – Режим доступа: www.wisdomweb.ru › JS › javascript-first, свободный. – Дата доступа: 28.06.2020.

8. Зеньковский, В. А. Программирование на VisualBasic6.5 и VisualBasic.net/ В. А. Зеньковский. – М. : СОЛОН-ПРЕСС, 2006. – 248 с.

9. Браун, Э. Изучаем JavaScript = Learning JavaSript : руководство по созданию современных веб-сайтов / Э. Браун. – 3-е изд. – М. ; СПб. : Диалектика, 2020. – 367 с.

10. Трофимов, В. В. Алгоритмизация и программирование : учебник для студентов высших учебных заведений, обучающихся по экономическим направлениям / В. В. Трофимов, Т. А. Павловская ; под ред. В. В. Трофимова. – М. : Юрайт, 2020. – 136 с.

11. Колокольникова, А. И. Практикум по информатике: основы алгоритмизации и программирования / А. И. Колокольникова. – М. ; Берлин : Директ-Медиа, 2019. – 424 с. – Режим доступа: https://biblioclub.ru/index.-php?page=book&id=560695. – Дата доступа: 16.04.2021.

12. Нагаева, И. А. Основы алгоритмизации и программирования: практикум / И. А. Нагаева, И. А. Кузнецов. – М. ; Берлин : Директ-Медиа, 2021. – 169 с. – Режим доступа: https://biblioclub.ru/index.php?page=book&-id=598404. – Дата доступа: 16.04.2021.

13. Грошев, А. С. Информационные технологии: лабораторный практикум / А. С. Грошев. – 2-е изд. – М. ; Берлин : Директ-Медиа, 2015. – 285 с. – Режим доступа: https://biblioclub.ru/index.php?page=book&id=434666. – Дата доступа: 16.04.2021.

14. Лукин, С. Н. VisualBasic: самоучитель для начинающих / С. Н. Лукин. – М. : Диалог-МИФИ, 2012. – 448 с. – Режим доступа:https://-biblioclub.ru/index.php?page=book&id=136080. – Дата доступа: 16.04.2021.

5.5 Список дополнительной литературы

1. Ахо, А. Построение и анализ вычислительных алгоритмов / А.Ахо, Дж. Хопкрофт, Дж.Ульман.– М.: Мир, 1979. – 536 с.

2. Вирт, Н. Алгоритмы и структуры данных /Н. Вирт. – М.: Мир, 1989. – 374 с.

3. Язык программирования JavaScript [Электронный ресурс]. – Режим доступа: learn.javascript.ru › js. – Дата доступа: 22.11.2019.

4. Берков, Н. А. Программирование на VisualBasic : учеб. пособие / Н. А. Берков. – М. : МГИУ, 2001. – 152 с.

5. Программирование на VisualBasic : учебник самоучитель [Электронный ресурс]. – Режим доступа: samouchitelbox.ru › pro-basic-main. – Дата доступа: 12.09.2020.

6. Кнут, Д. Искусство программирования для ЭВМ: В 3 т. Т.1: Основные алгоритмы / Д.Кнут. – М. : Мир, 2000. – 884 с.