# РАЗДЕЛ 1. ОСНОВЫ АЛГОРИТМИЗАЦИИ

**Тема 1. ОБЩИЕ СВЕДЕНИЯ ОБ АЛГОРИТМАХ**

**Алгоритм и его свойства.** Алгоритм – сведение задачи к последовательным этапам действий используется при выполнении последующих.

Свойства алгоритма:

**Дискретность** – значение новых величин вычисляются по определенным правилам из других величин с уже известными значениями.

**Определенность** – каждое правило из набора однозначно, а сами данные однозначно связаны между собой, т.е. последовательность действий алгоритма строго и точно определена.

**Результативность** – алгоритм решаю поставленную задачу за конечное число шагов.

**Массовость** – алгоритм разрабатывается так, чтобы его можно было применить для целого класса задач, например, алгоритм вычисления определенных интегралов с заданной точностью.

**Разновидности структур алгоритмов.**

**Способы описания алгоритмов: словесное описание, графическое представление, запись на алгоритмическом языке.**

Существует несколько способов описания алгоритмов:

**Словесное** – при словесной записи алгоритма используется следующие конструкции:

* Шаг (этап) этап обработки (вычисления) значения данных – «=»;
* Проверка логического условия: если (условие) истинно, то выполнить действие 1, иначе – действие 2;
* Переход (передача управления) к определенному шагу (этапу) N.

Пример:

1) ввод исходных данных a, b, c (a,b,c ¹ 0);

2) вычислить дискриминант D = b 2 – 4×a×c ;

3) если D < 0, то перейти к п. 6, сообщив, что действительных корней нет;

4) иначе, если D ³ 0, вычислить х1= (–b+ D )/(2×a) и х2 = (–b– D )/(2×a);

5) вывести результаты х1 и х2 ;

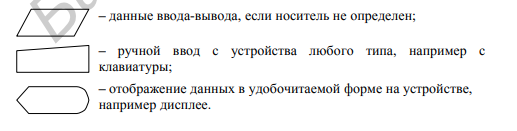
6) конец

**Графическое изображение алгоритма** – это представление его в виде схемы, состоящей из последовательности блоков, каждый из которых отображает содержание очередного шага алгоритма.

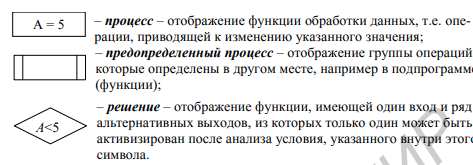
Схема данных состоит из следующих элементов

* Символ данных (символ данных могут отображать вид носителя данных)
* Символ процесса, который нужно выполнять над данным;
* Символ линии, указывающих потоки данных между процессами и носителями данных
* Специальные символы, которые используются для ввода облегчения чтения схемы алгоритма.

Символы ввода-вывода данных

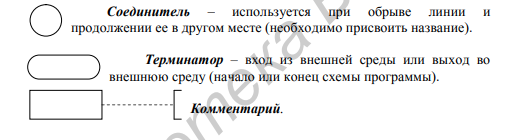


Символы процесса



Символы линий – отображают поток данных или управления. Линии горизонтальные или вертикальные, имеющие только прямой угол перегиба. Стрелки – указатели направления не ставятся, если управление идет сверху вниз или слева направо.

Специальные символы



**Запись на алгоритмическом языке**

Характеристики алгоритмического языка

* Алфавит – набор используемых символов
* Синтаксис – система правил образования конструкций языка
* Семантика – система правил, определяющих смысл и способ употребления конструкций языка

**Тема 2. РАЗНОВИДНОСТИ СТРУКТУР АЛГОРИТМОВ**

**Типы вычислительных процессов. Линейный, разветвляющийся и циклический вычислительные процессы.**

Можно выделить разделить на три виды

* Линейный (единственное направление выполнения);
* Разветвляющийся (направление выполнения определяет условие);
* Циклические (отдельные участки вычислений выполняются многократно);

**Классификация циклов.**

Циклы могут быть простыми и сложными (вложенным)

**Параметрические циклы: с предусловием, с постусловием, со счётчиком.**

**Циклы с предусловием** — это цикл, который выполняется до тех пор, пока условие остается истинным. Проверка выполняется перед выполнением условием. Используется цикл while.

**Цикл с постусловием** – это цикл, который выполняет хотя бы одну итерация, так как условие проверяется после выполнения тела цикла. Это обозначается что тела цикла будет выполнено минимум один раз, даже если условие изначально ложно. Используется цикл do while.

**Цикл со счетчиком** – это цикл в котором известно количество итераций заранее известно и управляется счетчиком. Такой цикл обычно используется для волнения блока кода определенное количество раз. Используется конструкция for.

Итерационные циклы.

Итерационные циклы – это циклы, в которых количество повторений заранее неизвестно и зависит от получаемого в результате.

Примеры вычислительных алгоритмов.

**Тема 3. СТРУКТУРНОЕ ПРОГРАММИРОВАНИЕ**

Общие сведения о структурном программировании.

Представление структурированных схем.

Восходящее и нисходящее проектирование.

Представляют собой 2 подхода к проектированию систем и программного обеспечения.

Нисходящее проектирование начинается с общего представления системы и последовательно разбирает его на более мелкие компоненты.

Преобразование неструктурированных программ в структурированные.

Тема 4. ГРАФИЧЕСКОЕ ПРЕДСТАВЛЕНИЕ АЛГОРИТМОВ

Способы графического представления алгоритмов.

Стандартизация графического представления алгоритмов.

Символы схем.

Единая система программной документации ГОСТ 19.701-90.

# РАЗДЕЛ 2. ОСНОВЫ КОМПЬЮТЕРНОЙ ТЕХНИКИ И ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ

Тема 5. АРИФМЕТИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРА

Системы счисления. Выделяются несколько основных систем:

* Десятичная;
* Двоичная
* Восьмеричная
* Шестнадцатеричная
* Двоично-десятичная

Классификация систем счисления.

Перевод чисел из одной системы счисления в другую.

Для перевода из 10 в двоичную надо / 10 а зачем читать с конца

Арифметические операции над положительными числами.

Сложение, вычитание, деление, умножение,

Арифметика с алгебраическими числами.

Сложение и вычитание, умножение, деление, степень

Двоичные коды и операции с двоичными кодами.

Двоичные коды – это представление информации с использованием двух символов, обычно 0 и 1

* Побитовое И (AND)
  + Операция, где результатом является 1 только тогда, когда оба соответствующих бита равны 1;
  + 1010 AND 1100 = 1000;
* Побитовое ИЛИ (OR)
  + Операция, где результатом является 1, если хотя бы один из соответствующих битов равен 1;
  + Пример: 1010 OR 1100 = 1110;
* Побитовое исключающее ИЛИ (XOR)
  + Операция, где результатом является 1, если биты различны, и 0, если биты одинаковы;
  + Пример: 1010 XOR 1100=0110;
* Побитовое НЕ (NOT)
  + Операция, которая инвертирует каждый бит (0 становится 1, а 1 становится 0);
  + Пример: NOT 1010=0101;
* Сдвиг влево
  + Операция, которая сдвигает все биты влево на заданное число позиций, добавляя нули справа.
  + Пример: 1010≪1=10100
* Сдвиг вправо
  + Операция, которая сдвигает все биты вправо на заданное число позиций. При арифметическом сдвиге вправо добавляются нули слева, при логическом – копируется старший бит.
  + Пример: 1010≫1=0101

Представление данных в компьютерной технике.

Данные в компьютерной технике хранятся двоичной системе счисления, которая использует только два символа 0 и 1.

Представление чисел с фиксированной и с плавающей точкой.

Хранение чисел с плавающей точной используется для хранения целых и дробных чисел в системе. Для хранения в системе обычно используют различные типы данных такие как double. Float, decimal. Также у чисел есть точность (кол-во знаков после запятой)

Тема 6. ПРИНЦИП ПРОГРАММНОГО УПРАВЛЕНИЯ

Принцип программного управления.

Автоматическое выполнение команд программы.

Процессы сборки, компиляции и исполнения программ.

**Процесс сборки** включает в себя все шаги, необходимые для создания исполняемого или интерпретируемого кода из исходных файлов. Это может включать компиляцию, связывания и другие шаги.

Основные этапы сборки:

* Компиляция: Преобразование исходного кода в объектный код

# РАЗДЕЛ 3. ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММНОМ ОБЕСПЕЧЕНИИ

Тема 7. СТАДИИ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Разработка программного обеспечения как многошаговый процесс.

Разработка включает в себя:

* Сбор и анализ требований
* Проектирование
* Разработка
* Тестирование
* Внедрение и сопровождение

Этапы постановки и решения задачи на компьютере. Цели и задачи каждой стадии разработки программного обеспечения.

Тема 8. СИСТЕМЫ ПРОГРАММИРОВАНИЯ

История развития систем программирования.

Классификация систем программирования.

Классифицируются по нескольким видам:

* По уровню реализации
  + Низкоуровневые (C, Assembler)
  + Высокого (C#, Java и тд)
* По способу реализации
  + Компилируемые (С++, Delphi, Go)
  + Интерпретируемые (Python, Ruby);
* По парадигмам программирования
  + Процедурное программирование (C, Pascal)
  + Объектно-ориентированное программирование (C++, Python, Ryby)
  + Декларативное программирование (Haskell, Lisp, SQL);

Назначение и состав системы программирования.

Основные понятия языка.

Классификация языков программирования.

# РАЗДЕЛ 4. ОСНОВЫ ПРОГРАММИРОВАНИЯ И СРЕДА РАЗРАБОТКИ

Тема 9. ОСНОВНЫЕ ЭЛЕМЕНТЫ ЯЗЫКА ПРОГРАММИРОВАНИЯ

Основные понятия языка.

Лексемы языка: алфавит, константы, зарезервированные слова.

Алфавит языка программирования включает: латинские буквы, цифры, специальные символы, пробелы и управляющие символы.

Константы — это неизменяемые значения. Также выделяются несколько типов такие как числовые константы, символьные константы, числовые константы, булевые константы.

К зарезервированным словам можно отнести такие слова как float, int, while, try, for, class и тд. Те слова, которые используются для написания кода.

Типы лексем.

К лексемам можно отнести несколько типов лексем:

Идентификаторы, комментарии и операции.

Идентификаторы используются для именования различных программных элементов, таких как переменная, класс и тд.

Комментарии — это текст, который игнорируется программой и используется для написания комментариев к программе. Выделяются несколько типов комментариев: однострочные и многострочные.

К операция относится несколько типов операций:

* Математические: сложение, вычитание, умножение, деление
* Операторы сравнения: ==, !=, < и тд
* Логические: и, или, исключающее или, не;
* Операторы присваивания: =, +=, -+ и тд

Выражения.

Приоритет выполнения операций.

Основные операторы языка.

Присваивание.

Операция присваивания представляет собой присваивания правого значения в левое. Также есть сокращенные формы присваивания, которые сокращают запись такие как +=, -= и тд

Пустой оператор.

Пустой оператор представляет собой строку которая ничего не делает и пишется с помощью “;”

Условный оператор, оператор выбора.

Условный оператор представляет собой блок кода, который выполняет в зависимости от условия. Выделяются такие конструкции как if, if else. Также есть сокращенная форма написания цикла, она представляет собой тернарный оператор.

Оператор представляет собой выбор одного из нескольких возможных значений вариантов выполнения кода на основе значений выражения. Используется switch case

Организация циклов.

Цикл **for** используется количество итераций известно заранее. Он состоит из инициализации, условия и итерационного выражения.

Синтаксис (for( инициализация;условие;итерация){})

Пример (for(int i = 0; i < 10; i++){})

Цикл **while** используется, когда количество итераций заранее неизвестно и цикл должен выполняться, пока выполняется определенное условие.

Синтаксис (while(условие))

Пример (while(i < 10){})

Пример **do-while** похож на цикл while, но условие проверяется после выполнения блока кода. Это гарантирует, что блок кода выполнится хотя бы один раз.

Синтаксис (do{}while(условие);)

Пример (do{}while (i < 10);))

Цикл **foreach** используется для перебора элементов в коллекции (например, массиве или списке). Это цикл удобен, когда нужно пройтись по всем элементам коллекции.

Синтаксис (foreach(тип переменная in коллекция){}){})

Пример (foreach (int number in numbers){})

Простой и составной оператор.

Основные возможности организации ввода/вывода.

Структура программного модуля.

Обязательные и дополнительные компоненты программного модуля.

Стандартные функции и структура простейшей программы.

Тема 10. ТИПЫ ДАННЫХ

Понятие типизации.

Каждая переменная или константа имеет тип, как и каждое выражение, результатом которого является значение;

Понятие типа данных.

Тип данных – классификация возможных значений, которые могут быть расположены в данной переменной.

Классификация типов данных.

К классификации типов данных относятся такие типы

* Простые типы;
* Ссылочные типы;
* Типы значений;
* Обобщенные типы;
* Пользовательские типы;

Базовые типы данных языка.

К базовым типам языка программирования относятся:

* Целочисленные значения
* Числа с плавающей запятой
* Логический тип
* Символьный тип
* Строки
* Объектный тип данных

Типизированные объекты (константы, переменные).

**Переменная** именованная область кода, которая хранит в себе некие значения и может изменяться во время выполнения программы.

**Константы** представляют собой именованную область кода, которая не может быть изменена во время выполнения программы.

Совместимость типов.

Совместимость типов представляет собой возможность разных типов быть преобразованным в другой тип данных. Выделяются 2 типа.

Неявное преобразование – это преобразование, которое происходит автоматически, когда нет риска потери данных или переполнения. Обычно это происходит при преобразовании меньшего типа в больший.

Int intValue = 100;

Long longValue = intValue;

Явное преобразование – преобразование при котором требуется использование оператора приведения (type) и применяется, когда есть риск потери данных или переполнения.

Double doubleValue = 123.45;

Int intValue = (int)doubleValue;

Тема 11. СЛОЖНЫЕ ТИПЫ ДАННЫХ

Массивы: определение, описание, размещение в памяти, использование. Особенности работы с массивами.

Массив – это структура данных, которая содержит несколько элементов одного типа, расположенных в непрерывной области памяти;

Массивы могут быть одномерными и многомерными.

Элементы в памяти располагаются последовательно в блоках памяти.

Для многомерных массивов строки располагаются построчно.

Для работы с массивом используются индексы для обращения к полям. Массивы могут быть статическими и динамическими.

Также у массивом есть методы для работы с ним

Символьные переменные и строки.

Символьная переменная представляет собой один символ и хранится в типе данных **char**.

Строка представляет собой последовательность символов и хранится в типе данных **string**. Строки являются неизменяемыми объектами, что обозначает что после создания строки ее значения не может быть изменено. Строки заключены в двойные строки.

Особенности представления строк.

Особенностью в представлении строк является их **неизменяемость**. То, что строка после создания неизменяемая, т.е. любая операция со строкой, это ее пересоздание и выполнение той операции далее.

Специальные символы, которые используются для создания перехода на новую строку, кавычек и тд.

Организация, размещение в памяти, процедуры и функции обработки строк и символов.

Строки размещаются в памяти последовательно и заканчиваются нулевым символом.

Для строк характерны такие функции как strlen для возращения длины строки, для копирования строк и тд.

Множества: определение, размещение в памяти, операции, процедуры и функции над множествами.

Записи. Организация, размещение. Записи с вариантами.

Тема 12. ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ

Указатели.

Указатели – это переменные, которые хранят адреса других переменных в памяти

Динамические структуры данных: назначение, виды, организация. Стандартные процедуры резервирования и освобождения памяти. Динамические массивы.

Списковые структуры.

**Односвязный список** – которые состоит из узлов, каждый из которых содержит данные и указатель на следующий узел. Последний узел указывает на nullptr, чтобы обозначить конец списка.

**Двухсвязный список** – это структура, где каждый узел содержит указатель на предыдущий или следующий узел. Это позволяет легко перемещаться в обе стороны по списку.

**Циклический список** – это структура, где последующий узел указывает на первый узел, создавая замкнутый круг. Могут быть как одно так и двусвязные циклические списки.

Классификация списковых структур.

Стеки, очереди, односвязные и двухсвязные линейные списки, кольца. Основные операции со стеком и с очередью.

Линейный упорядоченный список (словарь).

Рекурсивная и итеративная (нерекурсивная) реализации основных операций со списком.

Тема 13. ПОДПРОГРАММЫ

Модульность в программировании.

Модульность — это принцип проектирования программного обеспечения, который предполагает разделение программы на отдельные модули или компоненты, каждый из которых отвечает за выполнение определенной задачи или функциональности.

Понятие и структура подпрограммы.

Описание процедур и функций.

Виды параметров подпрограмм, локальные и глобальные переменные. Организация вызова подпрограмм, особенности их использования. Понятие рекурсии. Рекуррентные выражения.

Тема 14. БИБЛИОТЕКИ

Организация библиотек.

Использование библиотек в программе.

Для использования библиотеки используется команда using

Стандартные библиотечные модули и внешние модули пользователя.

Тема 15. РАБОТА С ФАЙЛАМИ

Виды файлов.

Выделяются такие типы файлов как:

* Текстовые: .txt, .md;
* Двоичные файлы: .dat, .bin;
* Файлы изображений: .jpg, .png, .gif;
* Аудио файлы: .mp3, .aac, .wav;
* Видео файлы: .mp4, .avi, .mkv, .mov;
* Архивные файлы: .zip, .rar, .tar;
* Документные файлы: .docx, .xlsx;
* Файлы базы данных: .sql, .sqlite, .mdb.

Процедуры и функции, определенные над файлами различного вида.

Нетипизированные файлы, их назначение и особенности использования.

Процедуры и функции обработки типизированных файлов.

# РАЗДЕЛ 5. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Тема 16. БАЗОВЫЕ ПОНЯТИЯ ОБЪЕКТНО-ОРИЕНТИРОВАННОЙ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Класс, объект, свойства.

**Класс** — это шаблон, по которому определяется форма объекта. Класс определяет состояние и поведение объекта посредством полей и методов.

**Объект** — это экземпляр класса. При создании объекта выделяется память для хранения данных, определенных в классе и объект может использовать методы и свойства класса.

**Свойства** – это механизм доступа к полям класса. Они позволяют безопасно читать и изменять значения полей. Свойства могут содержать логику для управления доступом к данным.

Синтаксис и особенности описания.

**Синтаксис** представляет собой идентификатор доступа, название класса, поля, свойства, конструктор, методы.

Создание класса представляет собой

Основные принципы объектно-ориентированной парадигмы: инкапсуляция, наследование, полиморфизм.

**Инкапсуляция** заключается в сокрытии деталей реализации и предоставлении доступа к данным и методам через определённый интерфейс. Это позволяет защитить данные от некорректного использования и упрощает управление кодом.

**Наследование** позволяет создавать новые классы на основе уже существующих, что способствует повторному использованию кода и упрощению его модификации. Производный класс наследует поля и методы базового класса.

**Полиморфизм** позволяет использовать объекты разных классов через общий интерфейс или базовый класс. Это достигается с помощью переопределения методов и реализации интерфейсов.

Тема 17. ОПИСАНИЕ КЛАССА

Описание класса.

Присваивание объектов.

Уровни инкапсуляции.

Выделяются несколько основных видов инкапсуляции. Такие как:

* public. Доступен всем классам и методам.
* private. Доступен внутри класса.
* Protected. Доступен внутри класса и в производных классах.
* Internal. Доступен внутри текущей сборки, но не за ее пределами.
* Protected internal. Доступен внутри текущей сборке и производных классах.
* Private protected. Доступен внутри текущего класса и в производных классах, которые находятся в той же сборке.

Реализация методов.

Методы объявляются внутри класса и структур. Синтаксис выглядит примерно вот так.

[модификатор доступа] [возвращаемый тип] [название метода] [(параметры)]

Создание и удаление объекта.

Создание объектов происходит при помощи конструктора с использованием оператора new.

В c# используется GarbageCollector для удаления объектов. Не требуется явного удаления.

В некоторых языках используется десткруктор для удаления созданного объекта.

Вызов родительских методов.

Для вызовов родительских методов, конструкторов и свойств. Используется ключевое слово base.

Обращение к полям и методам внутри объекта.

Для обращения к полям и методом класса используется ключевое слово this. Оно используется для доступа к членам класса или методам.

Дружественность.

Дружественные функции - это функции, которые не являются членами класса, однако имеют доступ к его закрытым членам - переменным и функциям, которые имеют спецификатор private.

а

Тема 18. МЕТОДЫ КЛАССА. ИНИЦИАЛИЗАЦИЯ И РАЗРУШЕНИЕ

Типы методов класса: статические, виртуальные, динамические, абстрактные, перегружаемые.

**Статические методы класса –** это методы, которые принадлежат классу, а не объекту класса, т.е. может использоваться без создания экземпляра класса.

**Виртуальные методы** – это методы, которые могут быть переопределены в производственных классах. Они определяются с помощью слова virtual в базовом классе и override для переопределения метода.

**Динамические методы** – это методы, которые выполняются во время выполнения, а не компиляции.

**Абстрактные методы** – это методы, которые определяются в абстрактный классах и не имеют реализацию. Данные методы необходимо реализовать в абстрактных классах.

**Перегружаемые методы** – это методы, которые могут иметь несколько реализаций с одинаковым именем, но разными параметрами.

Инициализация и разрушение объекта.

Инициализация объекта происходит при помощи оператора new и конструктора класса.

Разрушение объекта происходит при помощи сборщика мусора.

Конструкторы и деструкторы.

Конструктор — это метод, который создает экземпляр класса и возвращает его.

Деструктор — это метод, который уничтожает экземпляр класса. Используется для экономии памяти устройства.