1.Что такое информация? Варианты определения данного понятия и их классификация.

Информация – это совокупность каких-либо сведений, данных, передаваемых устно (в форме речи), письменно (в виде текста, таблиц, рисунков, чертежей, схем, условных обозначений)либо другим способом (например, с помощью звуковых или световых сигналов, электрических и нервных импульсов, перепадов давления или температуры и т.д.).

Информация — это обозначение содержания, полученного из внешнего мира в процессе нашего приспособления к нему и приспособления к нему наших чувств. (Норберт Винер)

Информация-совокупность сведений, которые циркулируют в природе и об-ве ,и в технических системах)Глушков

2.Свойства информации

- Объективность(информ. обьективна, если она не зависит от методов ее фиксации и чьего либо мнения)
- Точность(определяется степенью юблизости ее близости к реальному состоянию обьекта,процесса явления)
- Достоверность(информация достоверна, если она отображает истинное положение дел.)
- Полнота информ.(если ее достаточно для понимания и принятия решений
- Актуальность (важнось для настоящего времени . Только актуальная информация полезна)
- Полезность (оценивается приминительно к нуждам конкретных потребителей)
- Синтаксическая адекватность(отображает формально-структурные характеристики информации, не затрагивая ее смыслового содержания)
- Связанность(не бывает изолирована от материальных обьектов, и всегда характеризуется связанностью со своим носителем)
- Осмысленность (смысл информации сохраняется независимо от формы ее представления)
- Неисчерпаемость(информация может иметь неограниченное число пользователей, использоваться неограниченное число раз и при этом оставаться неизменной)

3.Информатика как междисциплинарное научное направление: основные взгляды на содержание информатики.

Информатика-междисциплинарное научное направление, изучающее вопросы производства, хранения, накопления, передачи, обработки и использования информации

Информатика-методология работы с информацией, определяющая информационную культуру личности человека

Информация + Автоматика=Информатика

Информатика как прикладная дисциплина занимается:

- изучением закономерностей в информационных процессах (накопление, переработка, распространение);
- созданием информационных моделей коммуникаций в различных областях человеческой деятельности;
- разработкой информационных систем и технологий в конкретных областях и выработкой рекомендаций относительно их жизненного цикла: для этапов проектирования и разработки систем, их производства, функционирования и т.д.

Главная функция информатики заключается в разработке методов и средств преобразования информации и их использовании в организации технологического процесса переработки информации.

4.Информационное общество. Признаки перехода к информационному обществу.

Информационное общество — общество, в котором большинство работающих занято производством, хранением, переработкой и реализацией информации, особенно высшей ее формы — знаний**.**

ПРИЗНАКИ

- Осознание обществом приоритетности информации перед другим продуктом деятельности человека.
- Объектом и результатом труда большей части населения являются не материальные ценности ,а информация
- Информация и информационные технологии являются товаром, определяющим основные экономические показатели
- У граждан нет проблем с доступом к информации
- Информация-оружие
- Равные возможности в доступе к информации всех слоев населения.
- Защита интеллектуальной собственности.

5.Системы счисления: определение, классификация, позиционные системы счисления и их основные понятия, сокращенная и полиномиальная запись чисел.

Система счисления - это совокупность правил и приемов записи чисел с помощью набора цифровых знаков. Количество цифр, необходимых для записи числа в системе, называют основанием системы счисления.

Классификация:

- позиционная(Традиционные k-ичные 10-чная, 2-чная;Смешанные K-Q-ичные: 2-10; нетрадиционная :факториальная, фибоначч.)
- непозиционная(аддитивные и др.)

Позиционная система счисления — значение всех цифр зависит от позиции (разряда) данной цифры в числе.

6. Перевод целых чисел из 10ой системы счисления в систему с основанием N по машинному алгоритму и методом подбор

1) **Машинный алгоритм-деление столбико.**
2) Метод подбора:

- Подобрать число $Y = K^n$ такое, что $Y \leq X < K^{n+1}$ и выписать его.
- Найти разность $D = X - Y$.
- Проверить: если D = 0, перевод окончен. Иначе перейти к пункту а, считая X = D
- Подсчитать количество одинаковых Y и записать эту цифру в соответствующую позицию n.

Пример перевода числа 17_{10} в число с основанием 3.

K=3 (равно основанию, в которое хотим перевести число)

n-позиция числа

1. $K^n \leq 17 \leq K^{n+1}$
2. $3^2 \leq 17 \leq 3^3$
   D=17-9=8
   X=8
3. $3^1 \leq 8 \leq 3^2$
   D=8-3=5
   X=5
4. $3^1 \leq 5 \leq 3^2$
   D=5-3=2
   X=2
5. $3^0 \leq 2 \leq 3^1$
   D=2-1=1
   X=1
6. $3^0 \leq 1 \leq 3^1$
   D=1-1=0
7. Теперь смотрим на n, которые были в левом столбце: 2,1,1,0,0. Считаем кол-во одинаковых n, это число записываем в соответствующую позицию: $\frac{2\,1\,0}{1\,2\,2}$.

$17_{10}=122_3$

7.Перевод вещественных чисел из 10ой системы счисления в систему с основанием N по машинному алгоритму.

- Способ перевода дробной части числа
  Дробную часть числа умножаем на систему счисления, в которую хотим перевести. От полученного числа отделяем целую часть, и записываем в позицию, начиная с -1.

Пример перевода 0.714_{10}

0.714*2=1.428
0.428*2=0.856
0.856*2=1.712
0.712*2=1.424
0.424*2=0.848
0.848*2=1.696
0.696*2=1.392
0.392*2=0.784
0.784*2=1.568
0.568*2=1.136

-1	-2	-3	-4	-5	-6	-7	-8	-9	-10
1	0	1	1	0	1	1	0	1	1

$0.714_{10}=0.1011011011_2$

8.Перевод вещественных чисел из системы с основанием N в десятичную.

- Для перевода можно воспользоваться следующей формулой

$A=a_n*N^n+a_{n-1}*N^{n-1}+...+a_0*N^0+a_{-1}*N^{-1}+...+a_{-m}*N^{-m}$

где A –исходное число, n-кол-во разрядов целой части, m-кол-во разрядов дробной части

9. Определение количества разрядов, обеспечивающих достаточную точность, при переводе вещественного числа из десятичной системы в систему с основанием N.

Для расчета достаточного количества разрядов после запятой пользуются следующим правилом: Если единица младшего разряда числа X, заданного в p-ичной СС, есть P^{-m}, то в его K-ой записи следует сохранить L разрядов после запятой, где L удовлетворяет условию:

$$K^{-L} > \frac{P^{-m}}{2} > K^{-(L+1)}$$

Пример расчета достаточного кол-ва разрядов для числа 0.714_{10} в 2-чную систему счисления

$p=10^{-3}$
$2^{-L}>\frac{0.001}{2}>2^{-(L+1)}$
$-L=\log_2 0.0005$
L<10.96
L=10
$2^{-10}>0.0005>2^{-(10+1)}=0.00048$
Следует сохранить 10 разрядов после запятой

10. Перевод чисел из системы с основанием N в систему с основанием M, где $M=N^K$ и наоборот.

Из СС с основанием N в СС с основанием M:

В записи числа с основанием N нужно выделять группы разрядов размером K и переводить каждую группу в соответствующую цифру M-ой системы. При переводе дробной части группы выделяются, начиная со старших членов. Недостающие разряды в группах заполняются нулями.

Каждая цифра числа записывается в системе с основанием M как число из K разрядов в системе с основанием N.

Пример

$10111011_2=X_8$
$8=2^3$
Выделяем в числе группы по 3, начиная справа:
010 111 011. Каждую из групп переводим:
$010_2=2_8$
$111_2=7_8$
$011_2=3_8$
$10111011_2=273_8$

11. Арифметические действия в позиционных системах счисления: сложение, вычитание, умножение, деление (на примере двоичной системы).

Сложение/вычитание выполняются поразрядно. В случае переполнения разряда занимается следующий разряд (и наоборот).

+1100101
11
1101000

-1100101
11
1100010

Если у числа есть дробная часть, сложение/вычитание начинается с дробной части.

Если при вычитании уменьшаемое меньше вычитаемого, то результат будет отрицательным. В этом случае можно из вычитаемого вычесть уменьшаемое и добавить минус.

-1011
101
110

Умножение также происходит поразрядно, как и умножение десятичных чисел:

1100101
101
+ 1100101
1100101
111111001

Умножение числа с дробной частью происходит с дробной части по правилам умножения. Запятая ставится по правилам умножения.

101,011
10,110
101,011
101,011
101,011
1110,110010

Деление происходит как и в десятичных числах.

1100100|101
101 |10100
101
101
0

Деление меньшего числа на большее. К делимому справа приписываем 0, если получившееся число делится на делитель-делим, если нет, то продолжаем справа добавлять 0, а в ответ также ставим 0:

101|10100
|0, 01
10100
10100
0

Деление с дробными числами.

1100,1|101
101 |10,1
101
101
0

12. Двоично-десятичная система: определение, достоинства и недостатки, правила выполнения сложения и вычитания.

Форма записи рациональных чисел, когда каждый десятичный разряд числа записывается в виде его четырёхбитного двоичного кода. Используется в калькуляторах, цифровых часах.

Преимущества: легкий ввод-вывод числовой информации

Недостатки: (повышенный расход памяти; осложнены арифметические операции

Сложение:

$467_{10} + 758_{10} = 1011\ 1111\ 1111$(чисел в 2-10 нет)

=> + 6 0110 0110 0110

0001 0010 0010 $0101_{2\text{-}10}$ (1225_{10})

Вычитание:

$758_{10} – 467_{10}$ =

0111 0101 1000 - 0100 0101 0111 =

0010 1111 0001 (1111 не существует)

– 0110

0010 1001 0001 (291_{10})

13. Прямой, обратный, дополнительный коды, модифицированные коды: определение, назначение, правила перевода, достоинства и недостатки.

Прямой код- п редназначен для отображения целых и дробных чисел со знаком.

Сначала ставится знаковый разряд (0 пол., 1 отр). На бумаге знаковый разряд отделяется точкой. Затем записывается само число.

Например,

$0.1011,1_{2ПР} = 11,5_{10}$;

$1.1011,1_{2ПР} = -11,5_{10}$

Преимущества

1.Получить прямой код числа достаточно просто.

2. коды положительных чисел относительно беззнакового кодирования остаются неизменными.

3.Кол-во положительных чисел равно кол-ву отр.

Недостатки:

1.Выполнение арифметических операций с отрицательными числами требует усложнения архитектуры центрального процессора (например, для вычитания невозможно использовать сумматор, необходима отдельная схема для этого).

2.Существуют два нуля: +0 (100…000) и -0 (000…000), из-за чего усложняется ариф. Сравн)

Обратный код. -положительные числа в обратном коде выглядят так же, как и в прямом, отрицательные формируются следующим образом: ставится знаковый разряд (1), а затем записывается положительное число с инвертированными цифрами (0 → 1, 1 → 0). ($1010_{2ОБР} = 10_{10}$; $1.0101_{2ОБР} = -10_{10}$.)

N-битное число в обратном коде содержит N-1 значащих разрядов и 1 знаковый!

Дополнительный код- применяется для представления целых и дробных чисел со знаком. Положительные числа выглядят так же, как и в прямом коде

В знаковый разряд отрицательных чисел ставится единица, далее берется число в обратном коде, и к младшему разряду арифметическим образом прибавляется единица.

$-14_{10} = 1.0010_{2ДОП}$ $-8_{10} = 1.1000_{2ДОП}$

Альтернативный способ перевода. Нужно переписать все биты исходного числа **справа налево** до первой единицы, **включая ее**. Остальные биты инвертировать. Поставить знаковый разряд.

Преимущества

1.Возможность заменить арифметическую операцию вычитания операцией сложения и сделать операции сложения одинаковыми для знаковых и беззнаковых типов данных,

Недостатки:

1.Ряд положительных и отрицательных чисел несимметричен.

2.В отличие от сложения, числа в дополнительном коде нельзя сравнивать как беззнаковые, или вычитать без расширения разрядности.

Модифицированный код -для придания однозначности записи числа могут использоваться ***модифицированный обратный и дополнительный код***. В модифицированных кодах используются два знаковых разряда:

00 – положительное число

11 – отрицательное число

01 – положительное число с переполнением

10 отрицательное число с переполнением

Правила вычислений в модифицированных кодах такие же.

14. Понятие разрядной сетки. Понятие переполнения. Понятие машинного нуля.

Разрядная сетка – это множество двоичных разрядов, выделяемых в памяти для изображения чисел. Величина разрядной сетки зависит от разрядности процессора. От того, как именно она используется, зависят диапазон и точность представления чисел. При выполнении арифметических операций возможны случаи, когда результат не помещается в разрядную сетку.

Если число вышло за пределы РС слева, говорят о переполнении разрядной сетки.

15. Сложение и вычитание в обратном и дополнительном коде. Переполнение и его устранение.

- **Обратный код**

	Прямой	Обратный
67	0.1000011	0.1000011
78	0.1001110	0.1001110
-67	1.1000011	1.0111100
-78	1.1001110	1.0110001

1) 67+78:

0,1000011 В данном случае переполнение является

0,1001110 признаком выхода числа из разрядной

1,0010001 сетки, так что запятую следует перенести на разряд влево. Ответ будет выглядеть

0,10010001

2) -67+78:

1.0111100 Произошло переполнение, 1 слева

0.1001110

10,0001010

прибавляем к числу.

0,0001010
1
0,0001011

3) 67-78:

0.1000011

1.0110001 Ответ можно перевести в прямой код:

1.1110100 $1,0001011 = -11_{10}$

4) -67-78:

1.0111100 Произошло переполнение. 1 прибавляем

1.0110001 к числу

10.1101101 1.1101101 Если складываются 2 отр.

1. то при переполнении в

1.1101110 знаковом остается 1

- **Дополнительный код**

	Обратный	Дополнит.
67	0.1000011	0.1000011
78	0.1001110	0.1001110
-67	1.0111100	1.0111101
-78	1.0110001	1.0110010

1) 67+78:

0.1000011
0.1001110
1.0010001

В данном случае переполнение является признаком выхода числа из разрядной сетки, так что запятую следует перенести на разряд влево. Ответ будет выглядеть

0,10010001

2) -67+78:

1.0111101
0.1001110
10.0001011

Перенос игнорируется (если разные знаки).

3) 67-78:

0.1000011
1.0110010
1.1110101

Чтобы перевести число в прямой код, нужно отнять 1, потом инвертировать.

4) -67-78

1.0111101
1.0110010
10.1101111

Переполнение является признаком выхода числа из разрядной сетки, так что запятую следует перенести на разряд влево.1.01101111

*если перед запятой получается 11

(например, -8-6)

1.1000

1.1010

11.0010, то первая 1 игнорируется, за исключение случая, когда после запятой все 0, тогда это указывается на переполнение и надо осуществить сдвиг влево (например, -60-68=-128

1.1000100

1.0111100

16. Код со смещением: определение, назначение, правила выполнения сложения и вычитания. Позволяет сдвинуть числовую шкалу, содержащую как отрицательные, так и положительные числа, полностью в область положительных чисел.

Если n – доступное количество разрядов, то

2^{n-1} – максимальное число в смещенном коде, записываемое как 1..1;

$1-2^{n-1}$ – минимальное число в смещенном коде, записываемое как 0..0;

$2^{n-1}-1$ – величина смещения;

При выполнении арифметических операций необходимо учитывать смещение. Чтобы получить в конце верный результат, смещение необходимо вычесть.

17. Представление чисел с фиксированной точкой: варианты фиксации точки для чисел со знаком и без, диапазоны представления чисел. Целочисленные типы данных.

Варианты фиксации точки:

Справа от младшего разряда

Где-то в середине сетки

Слева от старшего разряда

- **Числа с точкой справа от младшего разряда могут быть**

Без знака (только положительные)

n разрядов в сетке

Диапазон: 0..1-2^{-n}

Со знаком 1 разряд занимает знак, остается n-1

Диапазон: $0 \le |x| \le 1-2^{-(n-1)}$

- **Фиксированная в середине**

Без знака

Диапазон 0..$2^{k}-2^{-m}$

Со знаком

Диапазон $0 \le [x] \le 2^{k}-2^{-m}$

- **Фиксированная в слева**

Без знака

Диапазон 0..$1-2^{-n}$ n разрядов

Со знаком

Диапазон $0 \le [x] \le 1-2^{-(n-1)}$ n разрядов

- **Целочисленные типы:**

Беззнаковые целые Целые со знаком

18. Представление чисел с плавающей точкой. Общая идея. Диапазон представления чисел. Понятие нормализованного и ненормализованного числа.

Форма представления вещественных чисел, в которой число хранится в форме мантиссы и показателя степени.

- Представление числа

A=±M*$b^{±p}$ A-исходное число, M-мантисса, b-основание СС, p-экспонента (порядок)

- Диапазон чисел зависит от количества бит, отведённых для представления мантиссы и показателя

1) Число половинной точности:

Размер: 2 байта/16 бит

Бит в мантиссе 11

Бит в экспоненте 4

Бит на знак 1

2) Число одинарной точности:

Размер: 4 байта/32 бита

Бит в мантиссе 23

Бит в экспоненте 8

Бит на знак 1

3) Число двойной точности

Размер: 8 байт/64 бита

Бит в мантиссе 52

Бит в экспоненте 11

Бит на знак 1

- Нормализованное число-число, у которого после запятой идет значащая цифра, то есть в формуле A=±M*$b^{±p}$ M удовлетворяет условию $\frac{1}{n} \le |M| < 1$;

Пример:

$111.011*2^{10}=0.111011*2^{101}$

$0.010*2^{100}=0.10*2^{10}$

Ненормализованное число не удовлетворяет условию нормализованного.

19.Правила выполнения арифметических операций для чисел с плавающей точкой. Примеры.

- **Сложение и вычитание**: сначала производится выравнивание порядков (меньший по модулю порядок числа увеличивается до величины большего, а мантисса уменьшается в такое же количество порядков), а затем происходит сложение и вычитание мантисс.

Пример

$0.1*2^5$ и $0.1*2^3$

$+0.100*2^5$

$0.001*2^5$ –выравнивание порядка

$0.101*2^5$

$.0.100*2^5$

$0.001*2^5$

$0.011*2^5$-нормализация $0.11*2^4$

- **Умножение**: порядки складываются, мантисы перемножаются.

$0.1*2^5$

$0.1*2^3$

$0.01*2^8$-нормализация $0.1*2^7$

- **Деление**: из порядка делимого вычитается порядок делителя, а мантисса делится на мантиссу делителя.

$0.1*2^5$

$0.1*2^3$

$1*2^2$-нормализация $0.1*2^3$

В конце арифметических действий производится нормализация результата.

20. Представление чисел с плавающей точкой в соответствии со стандартом IEEE754: общие правила представления мантиссы, общие правила представления порядка.

- **Представление мантиссы**

В записи числа используется нормализованная мантисса. Стандарт определяет мантиссу следующим образом : она состоит из неяного бита, который всегда равен 1,двоичной точки и остальных разрядов..Получается, что мантисса охватывает диапазон чисел [1, 2). Мантисса представляется в прямом коде.

- **Представление порядка**

Порядок числа записывается в смещенном коде, т.е., к нему прибавляется фиксированное число, чтобы порядок был всегда неотрицательным. Это упрощает выполнения операций над порядками, избавляет от знакового разряда порядка. Истинный порядок может быть как положительным, так и отрицательным. Все доступные разряды порядка разделяются поровну между его положительными и отрицательными значениями. При выполнении арифметических операций процессор учитывает сдвиг.

Одна комбинация резервируется для специальных нужд.

Сдвиг порядка определяется по формуле $2^{n-1}-1$, где n-порядок (экспонента)

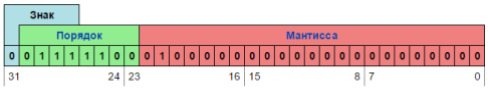
21. Представление чисел с плавающей точкой в соответствии со стандартом IEEE754: формат половинной точности.

Всего разрядов 16

Порядок 5 бит

Сдвиг порядка $2^{5-1}-1=15$

22. Представление чисел с плавающей точкой в соответствии со стандартом IEEE754: формат одинарной точности.



Всего разрядов 32

Порядок 8 бит

Сдвиг порядка $2^{8-1}-1=127$

23. Алгоритмы перевода чисел из 10ой системы в форматы стандарта IEEE754 и наоборот.

1. Перевести число из К-ичной системы счисления в двоичную (прямой код)
2. Представить двоичной число в нормализованной форме
3. Рассчитать смещенный порядок числа: $(n+m)_2$, где m – смещение, зависящее от формата хранения.
4. Разместить знак, порядок и мантиссу в соответствующие разряды сетки.
5. Разбить полученное число на тетрады и записать полученные двоичные разряды в виде числа в 16-ичной системе.

24. Базовые устройства схемотехники: понятие комбинационной схемы и цифрового автомата, классификация комбинационных схем и простых цифровых автоматов.

Комбинационные схемы — это устройства без памяти, принимающие на вход и возвращающие на выходе информационные сигналы в соответствии с их задачей и внутренним устройством. Набор выходных сигналов однозначно определяется набором входных.

Цифровой автомат – это логическое устройство, осуществляющее хранение, обработку и получение информации в соответствии с некоторым алгоритмом.

Среди комбинационных схем выделяют:

- Элементы И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ, ИСКЛ-ИЛИ
- Мультиплексоры и демультиплексоры
- Шифраторы
- и дешифраторы
- Компараторы
- Комбинационные сумматоры

Простые цифровые автоматы:

- Триггеры
- Счетчики
- Регистры

25. Основы алгебры логики: логическая переменная и логическая функция, способы задания логической функции.

Логическая переменная – это переменная, способная хранить значение из множества: ложь, истина.

Логическая функция – это функция от набора логических переменных, возвращающая значения на множестве {истина, ложь}.

Способы задания логических функций:

- Словесный – простое описание функции словами («возвращает Истину, когда две из трех истинны»)
- Табличный – для каждого набора входных переменных указывается значение функции
- Аналитический – функция задается логическим выражением
- Векторный – для каждого набора входных переменных в ряд записываются значения функции.

- Графический – изображается циклограмма (временная диаграмма). При этом предполагается, что наборы значений подаются на вход устройства в порядке, задаваемом таблицей истинности.
- Схемотехнический – задается комбинационная схема (как в Логисиме), которая реализует эту функци

26. Логические функции от двух переменных: названия, таблицы истинности, УГО.

x	x	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Заметим, что уникальных функций всего 8. Каждой соответствует инверсированная пара, иными словами:

$f_i = \overline{f_{15-i}}, i \in [0,15].$

$F_1(x_1,x_2)$ – **конъюнкция**

Элементарная логическая функция (логическое произведение, И). Конъюнкция **истинна** тогда и только тогда, когда все ее аргументы **истинны**.

$F_7(x_1,x_2)$ – **дизъюнкция**

Элементарная логическая функция (логическое сложение, ИЛИ). Дизъюнкция **истинна**, если хотя бы один ее аргумент **истинен**.

$F_6(x_1,x_2)$ – **строгая дизъюнкция**

Сложение по модулю 2, исключающее ИЛИ. Обозначение: $\oplus$.

$F_8(x_1,x_2)$ – **Элемент Вебба** (стрелка Пирса)

Реализует функцию ИЛИ-НЕ. Является базисным элементом, т.е. только через ИЛИ-НЕ можно реализовать любую логическую функцию. Возвращает **истину**, когда все аргументы **ложны**. Обозначение: ↑.

$F_{14}(x_1,x_2)$ – **Функция штрих Шеффера**

Реализует функцию И-НЕ. Является базисным элементом, т.е. только через И-НЕ можно реализовать любую функцию. Обозначение: |.

Прочие функции от двух переменных

- F_{13} – импликация ($x_1 \to x_2$)
- F_2 – отрицание импликации
- F_{11} – обратная импликация ($x_2 \to x_1$)
- F_4 – отрицание обратной импликации
- F_{12} – отрицание первого аргумента
- F_9 – отрицание М2
- F_{12} – отрицание второго аргумента

27. Основные понятия алгебры логики: конъюнкт, дизъюнкт, совершенный конъюнкт, совершенный дизъюнкт, минтерм, макстерм, дизъюнктивная форма, конъюнктивная форма.

Конъюнкт – конъюнкция некоторых переменных или их отрицаний.

Дизъюнкт – дизъюнкция некоторых переменных или их отрицаний.

Если конъюнкт (дизъюнкт) состоит из всех переменных функции или их отрицаний, где каждая переменная участвует лишь единожды, то такой конъюнкт (дизъюнкт) называется **совершенным**.

Минтерм– это логическая функция, принимающая значение истина только на одном наборе значений своих аргументов. Формальная запись минтерма – это конъюнкция всех аргументов функции, взятых с отрицанием или без него. Среди множества функций от K переменных есть 2^K минтермов. Минтерм – это совершенный конъюнкт.

Макстер – это логическая функция, принимающая значение ложь только на одном наборе значений своих аргументов. Формальная запись макстерама – это дизъюнкция всех аргументов функции, взятых с отрицанием или без него. Среди множества функций от K переменных есть 2^K макстермов. **Макстерм** – это совершенный дизъюнкт.

Дизъюнктивная нормальная форма (ДНФ) – дизъюнкция конечного числа конъюнктов.

Конъюнктивная нормальная форма (КНФ) – конъюнкция конечного числа дизъюнктов

28. Совершенная дизъюнктивная нормальная форма, совершенная конъюнктивная нормальная форма. Определение. Методы построения.

Совершенная ДНФ (СДНФ) – дизъюнкция совершенных конъюнктов (т.е. минтермов). Любая логическая функция, не являющаяся логическим нулем, имеет только одну СДНФ.

Совершенная КНФ (СКНФ) – конъюнкция совершенных дизъюнктов (т.е. макстермов). Любая логическая функция, не являющаяся логической единицей, имеет только одну СКНФ.

Построение СДНФ по таблице истинности

Выписать совершенные конъюнкции и связать их через дизъюнкцию.

Построение СКНФ по таблице истинности

Выписать совершенные дизъюнкции и связать их через конъюнкцию.

29. Основные логические законы и правила преобразования логических формул.

Коммутативный **Ассоциативный**

$XY = YX,$ $X(YZ) = (XY)Z,$

$X + Y = Y + X$ $X + (Y + Z) = (X + Y) + Z$

Дистрибутивный **Закон двойного отрицания**

$X(Y + Z) = XY + XZ$ $\bar{\bar{X}} = X,$

$X + YZ = (X + Y)(X + Z)$

Закон идемпотентности **Правило склеивания**

$XX = X,$ $(X + Y)(X + \bar{Y}) = X$

$X + X = X$ $XY + X\bar{Y} = X,$

Правило свертки **Правило поглощения**

$X(\bar{X} + Y) = XY,$ $X(X + Y) = X$

$X + \bar{X}Y = X + Y$ $X + XY = X$

Законы де Моргана

$X|Y = \overline{XY} = \bar{X} + \bar{Y}, \overline{\overline{X} + \overline{Y}} = XY,$

$X \uparrow Y = \overline{X + Y} = \bar{X}\bar{Y}\ \overline{\overline{X}\overline{Y}} = X + Y$

Правило раскрытия импликации

$X \to Y = \bar{X} + Y$

Правила раскрытия эквивалентности

$X \equiv Y = (X \to Y)(Y \to X),$

$X \equiv Y = XY + \bar{X}\bar{Y}$

Правило раскрытия строгой дизъюнкции

$XY = \bar{X}Y + X\bar{Y}$

30. Минимизация логических функций: цель минимизации, понятие МДНФ и МКНФ, минимизация методом эквивалентных логических преобразований.

Задача **минимизации** логической функции заключается в том, чтобы найти наиболее компактное ее представление в виде нормальной формы минимальной сложности – **минимальной дизъюнктивной нормальной формы (МДНФ)** или **минимальной конъюнктивной нормальной формы (МКНФ)**.

Минимальная дизъюнктивная нормальная форма – это дизъюнкция минимального числа конъюнкций переменных, взятых с отрицанием или без.

Минимальная конъюнктивная нормальная форма – это конъюнкция минимального числа дизъюнкций переменных, взятых с отрицанием или без.

Метод эквивалентных логических преобразований заключается в упрощении аналитической формы записи логической функции с помощью законов логики. Например, найдем МДНФ, имея СДНФ некоторой функции:

$$\begin{aligned} F_{A,B,C_{\text{СДНФ}}} &= \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC \\ &= \bar{A}BC + A\bar{B}C + AB \\ &= B(\bar{A}C + A) + A\bar{B}C \\ &= B(A + C) + A\bar{B}C \\ &= AB + BC + A\bar{B}C \\ &= AB + C(B + A\bar{B}) \\ &= AB + C(A + B) \\ &= AB + AC + BC \end{aligned}$$

31. Минимизация логических функций методом диаграмм Вейча: идея метода, понятие интервала логической функции, формы интервалов, правила выделения интервалов, правила построения диаграммы с целью получения МДНФ функции от 3-х переменных, алгоритм минимизации.

Графический способ минимизации логических функций. Работает на основе операций склеивания и поглощения. Представляет собой особым образом переупорядоченную таблицу истинности. В диаграмме Вейча ячейки таблицы истинности сгруппированы таким образом, что переход из одной ячейки в другую по вертикали или горизонтали связан с изменением значения только одной переменной. В результате этого наборы, между которыми возможно склеивание, получаются сгруппированными вместе и их легко заметить. Метод Вейча подходит для минимизации функций до 7 переменных. При большем количестве теряются достоинства метода.

Интервал логической функции от K переменных – это такое множество наборов значений переменных, что:

- Значение функции на этом множестве постоянно; **Мощность (величина, размер интервала)** этого множества равна 2^N, $N \le K$;
- N является количеством переменных, которые упрощаются на этом множестве, а оставшихся $(K - N)$ переменных достаточно для описания логической функции на данном множестве;
- Если $N > 0$, то каждый следующий набор отличается от предыдущего значением только одной переменной.

Правила выделения интервалов:

- Необходимо стараться выделить максимально большие интервалы;
- Каждый новый интервал должен содержать хотя бы одно значение, принадлежащее только ему;
- Необходимо выделить минимально возможное количество интервалов.

Диаграмма МДНФ для 3 переменных:

	y		$\bar{y}$	
x	$f(110)$	$f(111)$	$f(101)$	$f(100)$
$\bar{x}$	$f(010)$	$f(011)$	$f(001)$	$f(000)$
	$\bar{z}$	z		$\bar{z}$

Алгоритм минимизации:

1. Нарисовать исходную таблицу диаграммы и сделать ее разметку в зависимости от количества переменных функции.
2. Заполнить таблицу значениями функции с учетом цели минимизации (удобно выписывать только 1 для МДНФ и только 0 для МКНФ).
3. Выделить контурами интервалы из единиц (МДНФ) или нулей (МКНФ), соблюдая описанные выше правила.
4. Выписать формулу МДНФ (МКНФ), для чего:

a. Для каждого интервала выписать конъюнкт (дизъюнкт), в который будут входить только те переменные или их отрицания, которые сохраняют свое значение на интервале. Остальные переменные упростятся.
b. Соединить выписанные конъюнкты (дизъюнкты) через дизъюнкцию (конъюнкцию).

-

32. с целью получения МДНФ функции от 4-х переменных, алгоритм минимизации.

То же, что вопрос 31, кроме таблицы:

		b		$\bar{b}$	
a		$f(1100)$	$f(1101)$	$f(1001)$	$f(1000)$	$\bar{c}$
		$f(1110)$	$f(1111)$	$f(1011)$	$f(1010)$	c
$\bar{a}$		$f(0110)$	$f(0111)$	$f(0011)$	$f(0010)$	
		$f(0100)$	$f(0101)$	$f(0001)$	$f(0000)$	$\bar{c}$
		$\bar{d}$	d		$\bar{d}$	

33. МКНФ функции от 3-х переменных, алгоритм минимизации.

То же, что вопрос 31, кроме таблицы:

		y		$\bar{y}$	
x		$f(001)$	$f(000)$	$f(010)$	$f(001)$
$\bar{x}$		$f(101)$	$f(100)$	$f(110)$	$f(111)$
		$\bar{z}$	z	$\bar{z}$	

34. МКНФ функции от 4-х переменных, алгоритм минимизации.

То же, что вопрос 31, кроме таблицы:

		b		$\bar{b}$	
a		$f(0011)$	$f(0010)$	$f(0110)$	$f(0111)$	$\bar{c}$
		$f(0001)$	$f(0000)$	$f(0100)$	$f(0101)$	c
$\bar{a}$		$f(1001)$	$f(1000)$	$f(1100)$	$f(1101)$	
		$f(1011)$	$f(1010)$	$f(1110)$	$f(1111)$	$\bar{c}$
		$\bar{d}$	d		$\bar{d}$	

35. Минимизация частично определенных функций при помощи диаграмм Вейча.

В некоторых задачах нам известно, что определенные входные комбинации никогда не возникнут. В таком случае неопределенные значения интерпретируются так, как удобно. Для МДНФ все неопределенности считаем равными 1, для МКНФ – 0. При этом не ставим себе целью покрыть интервалами неопределенные значения, используем их только для поиска больших интервалов единиц (нулей).

36. Приведение минимизированной логической функции к базису «ИЛИ-НЕ».

(МДНФ к базису ИЛИ-НЕ)

$$F_{МДНФ} = a\bar{c} + b\bar{c}\bar{d} + bcd + \bar{a}\bar{b}c + \bar{a}\bar{b}d =$$

$$= \overline{\overline{a\bar{c}} + \overline{b\bar{c}\bar{d}} + \overline{bcd} + \overline{\bar{a}\bar{b}c} + \overline{\bar{a}\bar{b}d}}$$

(МКНФ к базису ИЛИ-НЕ)

$$F_{МКНФ} = (\bar{a} + b + \bar{c})(\bar{b} + \bar{c} + d)(a + b + c + d)$$

$$(a + \bar{b} + c + \bar{d}) = \overline{(\bar{a} + b + \bar{c})} + \overline{(\bar{b} + \bar{c} + d)} +$$

$$\overline{(a + b + c + \bar{d})} + \overline{(a + \bar{b} + c + \bar{d})}$$

37. Приведение минимизированной логической функции к базису «И-НЕ».

(МДНФ к базису И-НЕ)

$$F_{МДНФ} = a\bar{c} + b\bar{c}\bar{d} + bcd + \bar{a}\bar{b}c + \bar{a}\bar{b}d =$$

$$= \overline{a\bar{c}} * \overline{b\bar{c}\bar{d}} * \overline{bcd} * \overline{\bar{a}\bar{b}c} * \overline{\bar{a}\bar{b}d}$$

(МКНФ к базису И-НЕ)

$$F_{МКНФ} = (\bar{a} + b + \bar{c})(\bar{b} + \bar{c} + d)(a + b + c + d)$$

$$(a + \bar{a} + c + \bar{d}) = \overline{\overline{a\bar{b}c} * \overline{bc\bar{d}} * \overline{\bar{a}\bar{b}\bar{c}\bar{d}} * \overline{\bar{a}\bar{b}c\bar{d}}}$$

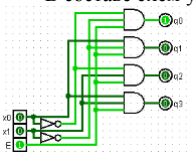
38. **Дешифраторы: определение, УГО, области применения, функциональная схема на примере дешифратора 2-4.**

Дешифратор – комбинационная схема, обладающая -адресными входами, одним разрешающим входом и 2^n выходами. На адресные входы подается двоичное число, которое в своем десятичном представлении задает номер выхода, на котором формируется значащий сигнал. Предназначена для преобразования n-разрядного двоичного кода в унитарный двоичный код разрядности 2^n.
В унитарном коде только один разряд из множества может принимать значение 1 (или 0). Это означает, что двоичное число (в своем десятичном представлении) задает номер того выхода, на котором появится 1 (или 0, если выходы инверсные).

УГО-Условное графическое обозначение

Области применения:

- В составе схем управления другими



устройствами для последовательной подачи разрешающих сигналов;
- В составе схем преобразователей кодов;
- Для реализации логических функций.

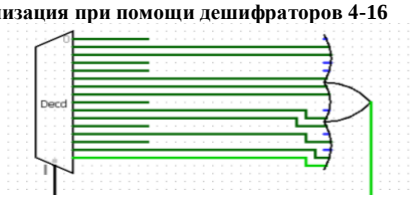
Функциональная схема на примере дешифратора 2-4

Пусть дешифратор имеет n входов. На выходах формируется код , разрядность которого меньше или равна 2^n

39. **Реализация логических функций на дешифраторах достаточной разрядности.**

Дешифратор называется **полным**, если число выходов равно максимально возможной разрядности выходного слова (2^n).

Реализация при помощи дешифраторов 4-16



40. **Реализация логических функций на дешифраторах меньшей разрядности, чем количество переменных.**

Дешифратор называется **неполным**, если часть входных разрядов не используется (то есть число выходов меньше 2^n).

Реализация при помощи дешифраторов 2-4

41. **Мультиплексоры: определение, УГО, области применения, функциональная схема мультиплексора на примере мультиплексора 4-1.**

Определение

Мультиплексор – комбинационная схема, у которой имеется 2^n информационных входов, n адресных входов, может присутствовать разрешающий вход и имеется 1 выход. Представляет собой управляемый переключатель, т.е. осуществляет подключение одного из информационных входов к единственному выходу под управлением адресных и разрешающего кодов. Сигнал на разрешающем коде управляет мультиплексором в целом: либо разрешает прохождение сигналов на выход, либо нет (в этом случае на выходе обычно 0).

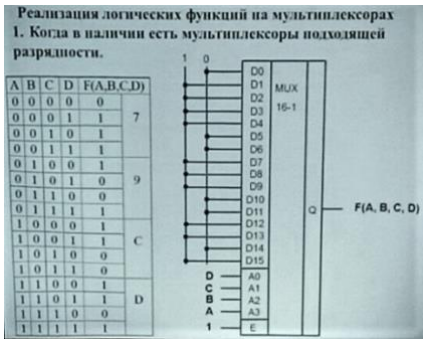
Области применения

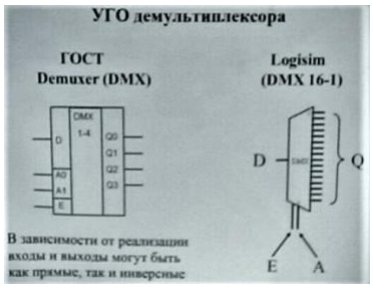
- «Ленивая» реализация логических функций, когда минимизацией можно пренебречь. Удобно для разработчика, но приводит к дополнительным затратам.
- В качестве коммутатора n к 1:
  - Для преобразования параллельного входа в последовательный;
  - Для поочередного подключения многих источников информации к одному потребителю.

Функциональная схема мультиплексора на примере мультиплексора 4-1. В основе мультиплексора лежит дешифратор.

42. **Реализация логических функций на мультиплексорах достаточной разрядности.**

Когда в наличии есть мультиплексоры подходящей разрядности



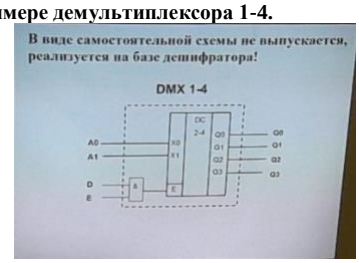


Области применения

В качестве коммутатора 1 к n (для поочередного подключения одного источника информации ко многим потребителям);

Для реализации логических функций (получается реализация на дешифраторах)

Функциональная схема демультиплексора на примере демультиплексора 1-4.



43. Реализация логических функций на мультиплексорах меньшей разрядности, чем количество переменных.

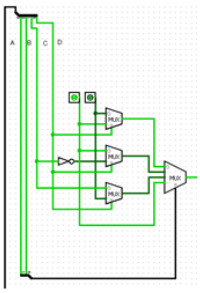
Когда в наличии есть мультиплексоры меньшей разрядности

4-1

потребуется дополнительный мультиплексор для управления первыми 4 переменными. Всего потребуется 5 мультиплексоров. подключение многих источников к одному получателю

Предположим, что в наличии есть только мультиплексоры 2-1.

A	B	C	D	F(A,B,C,D)
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1



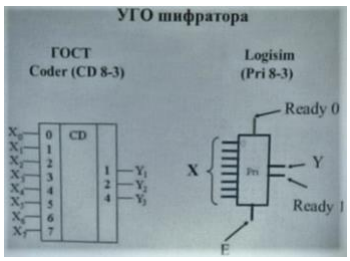
44. Демультиплексоры: определение, УГО, области применения, функциональная схема демультиплексора на примере демультиплексора 1-4.

Определение

Демультиплексор – комбинационная схема, у которой имеются: один информационный вход, n адресных входов, 2^n выходов, может присутствовать разрешающий вход. Осуществляет коммутацию единственного информационного входа к одному из выходов под управлением адресных и разрешающего входов. Сигнал на разрешающем входе управляет демультиплексором в целом: либо разрешает прохождение сигналов на выход, либо нет. В виде самостоятельной схемы не выпускаются. Реализуются на базе дешифратора.

45. Шифраторы: определение, УГО, области применения, таблица истинности и функциональная схема простого полного шифратора 4-2.

Шифратор – комбинационная схема, выполняющая обратную дешифратору функцию, имеющая: 2^n информационных входов, n выходов. Преобразует номер информационного входа, на котором сформирован значащий уровень сигнала, из десятичной системы счисления в двоичную.

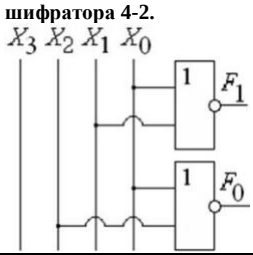


Области применения

Получение кодов нажатых клавиш;

В составе преобразователей кодов.

Комбинационная схема простого полного шифратора 4-2.



X0	X1	X2	X3	Y0	Y1
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

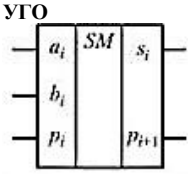
46. Таблица истинности и функциональная схема приоритетного шифратора 8-3.

47. Сумматоры: определение, УГО, классификация, четвертьсумматор, полусумматор.

Определение

Сумматор- цифровое устройство, выполняющее арифметическое сложение двух чисел в том или ином коде. Если используется специальные коды, то на сумматоре можно выполнять вычитание.

УГО

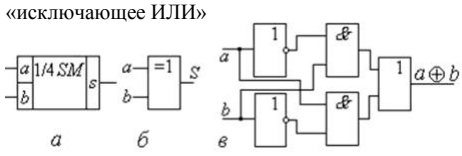


Классификация

1. В зависимости от с.с. (двоичные, 2-10, десятичные, другие)
2. По количеству одновременно обрабатываемых разрядов складываемых чисел сумматоры (одноразрядные, многоразрядные)
3. По числу входов и выходов одноразрядных двоичных сумматоров (четвертьсумматоры, полусумматоры, полные одноразрядные двоичные сумматоры)
4. По способу представления и обработки складываемых чисел многоразрядные сумматоры делятся на (последоват. , парал.)
5. По возможности сохранения результата(комбинационный, накапливающий)

Четвертьсумматор

Простейший двоичный сумматор. Имеет два входа для двух одноразрядных чисел и один выход, на который формируется их сумма. Реализуется логическими элементами «исключающее ИЛИ»



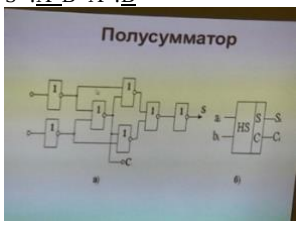
Полусумматор

Это комбинационная схема, которая вырабатывает сигналы суммы и переноса при сложении двух двоичных одноразрядных чисел A и B

A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

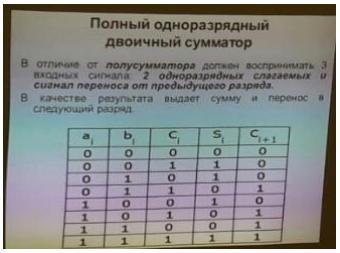
C=A*B

S=!A*B+A*!B



48. Полный одноразрядный сумматор, многоразрядный сумматор параллельного действия с последовательным переносом.

Полный одноразрядный сумматор должен воспринимать 3 входных сигнала: 2 одноразрядных слагаемых и сигнал переноса от предыдущего разряда. В качестве результата выдает сумму и перенос в следующий разряд.



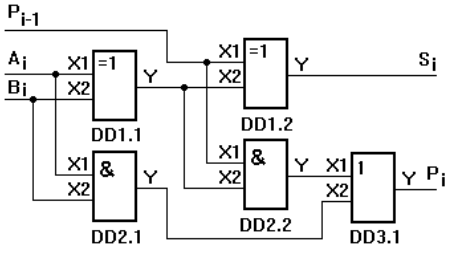
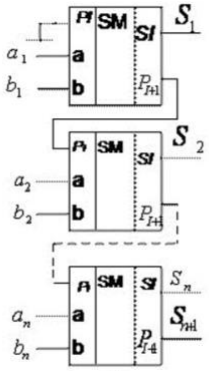


Рис. 1.32

Многоразрядный сумматор параллельного действия(аргументы подаются одновременно по всем разрядам) с последовательным переносом. Для сложения двух многоразрядных двоичных чисел на каждый разряд необходим один полный сумматор.

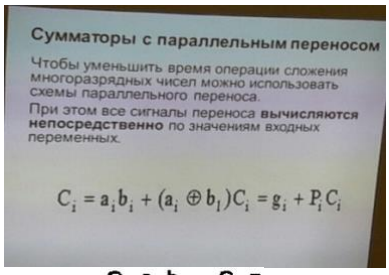


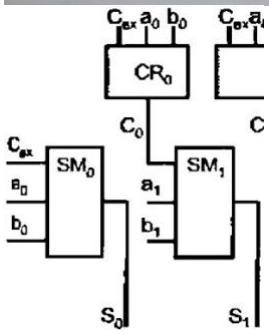
б)

сумматор паралл д с послед пер

49. Сумматор последовательного действия – принцип работы, обобщенная структурная схема, достоинства и недостатки.

50. Сумматоры с параллельным переносом – рекуррентная формула для вычисления переносов, пример схемы для 2х разрядного сумматора, достоинства и недостатки.



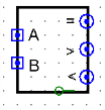


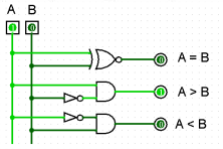
CR – специальная логическая схема, вычисляющая величину переноса.

51. Компараторы: определение, таблица истинности, выражения для вычисления отношений «равно» и «больше» на примере компаратора двухразрядных чисел.

Компаратор— электронная схема, принимающая на свои входы два аналоговых сигнала и выдающая логическую «1», если сигнал на прямом входе («+») больше, чем на инверсном входе («−»), и логический «0», если сигнал на прямом входе м
еньше, чем на инверсном входе.

Входы		Выходы		
x	y	$F_{A=B}$	$F_{A>B}$	$F_{A<B}$
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0





A(i+1)= B(i+1)	A(i+1)> B(i+1)	A(i+1)< B(i+1)	A(i)	B(i)	A=B	A>B	A<B
1	0	0	0	0	1	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	1	0
1	0	0	1	1	1	0	0
0	1	0	x	x	0	1	0
0	0	1	x	x	0	0	1

Признак равенства двух разрядов

$$r_i = a_i b_i + \bar{a}_i \bar{b}_i = \overline{a_i \bar{b}_i + \bar{a}_i b_i} = \overline{a_i \oplus b_i}$$

Признак равенства многих разрядов

$$R = r_{n-1} r_{n-2} \ldots r_0$$

Признак «больше» для двухразрядных чисел

$$F_{A>B} = a_1 \overline{b_1} + a_0 \overline{b_0} r_1,$$

Признак «больше» для многоразрядных чисел

$$F_{A>B} = a_{n-1}\overline{b_{n-1}} + a_{n-2}\overline{b_{n-2}}\, r_{n-1} + a_{n-3}\overline{b_{n-3}}\, r_{n-2} r_{n-1} + \ldots a_0 \overline{b_0} z_1 \ldots z_{n-1}$$

52. Триггеры: определение, классификация, способы синхронизации, области применения. Сигнал синхронизации: назначение, структура.

Триггер – простейший цифровой автомат, который представляет собой элементарную ячейку памяти и может хранить 1 бит информации. Находится только в одном из двух устойчивых состояний: 0 или 1.

Реализуется на базисах логических элементов: ИЛИ-НЕ, И-НЕ.

Эффект запоминания возникает благодаря наличию обратных связей м/у элементами.

Классификация:

1. По логическому функционированию:
   - RS, D, T, JK
2. По способу записи информации:
- Асинхронные
  Запись информации (переключение состояния) осуществляется в момент подачи сигнала на информационные входы.
- Синхронные
  Запись возможна только при наличии разрешающего сигнала C (clock), т.е. сигнала синхронизации.
   i. Со статическим управлением (стробируемые)
   ii. С динамическим управлением (тактируемые)
3. По количеству ступеней:
- Одноступенчатые

Для записи информации используется только одна ступень. Возникают проблемы при записи и считывании информации в пределах одного такта. (Что считано? Старая или новая информация?)

- Двухступенчатые

Состоят из двух одноступенчатых, работают в противофазе, в 2 раза медленнее, но решают

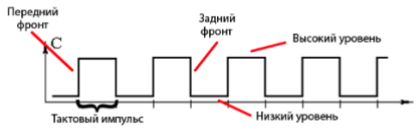
вышеописанную проблему: когда первая ступень хранит старую информация, вторая может принимать новую.

Способы синхронизации:

1. По уровню синхронизирующего сигнала (статические триггеры):
   - По уровню «1»
   - По уровню «0»
2. По фронту (динамические триггеры):
   - По переднему фронту
   - По заднему фронту

Сигнал синхронизации – последовательность дискретных импульсов стабильной частоты.

Структура:



Вход С называется прямым динамическим, если переключение триггера происходит в момент прихода переднего фронта, инверсным динамическим, если в момент прихода заднего фронта.

Назначение сигнала синхронизации: разрешать или запрещать запись информации в синхронных триггерах.

Область применения триггеров: используются в создании более сложных устройств: регистров и счётчиков.

Инфа по этому поводу из инета: использование триггеров позволяет реализовывать устройства оперативной памяти (то есть памяти, информация в которой хранится только на время вычислений). Однако это не единственная их область применения. Триггеры широко используются для построения цифровых устройств с памятью, таких как счётчики, преобразователи последовательного кода в параллельный, последовательные порты или цифровые линии задержки, применяемые в составе цифровых фильтров.

53. Триггеры: асинхронный RS-триггер на элементах «И-НЕ» и на элементах «ИЛИ-НЕ», его таблица истинности, функциональная схема, объяснение принципа работы, УГО.

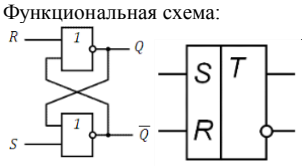
Асинхронный RS-триггер на элементах ИЛИ-НЕ

Таблица истинности:

S	R	Q_i	$\overline{Q_i}$	Пояснения
0	0	Q_{i-1}	$\overline{Q_{i-1}}$	Режим хранения информации
0	1	0	1	Сброс в 0
1	0	1	0	Установка 1
1	1	*	*	Неопределённое состояние

* - неопределённое состояние

Функциональная схема:

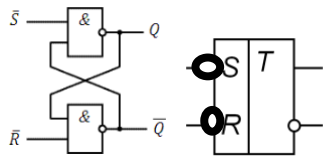


Асинхронный RS-триггер на элементах И-НЕ

Таблица истинности

S	R	Q_i	$\overline{Q_i}$	Пояснения
1	1	Q_{i-1}	$\overline{Q_{i-1}}$	Режим хранения информации
1	0	0	1	Сброс в 0
0	1	1	0	Установка 1
0	0	*	*	Неопределённое состояние

Функциональная схема: УГо



Принцип работы: RS-триггер получил название по названию своих входов. Вход S (Set — установить англ.) позволяет устанавливать выход Q в единичное состояние. Вход R (Reset — сбросить англ.) позволяет сбрасывать выход Q в нулевое состояние.

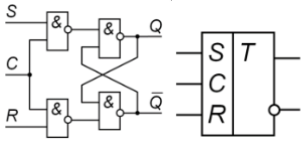
Отличие RS-триггеров, построенных на базисах ИЛИ-НЕ и И-НЕ, следует из их таблиц истинности. Они зеркальны по отношению друг к другу.

54. Триггеры: синхронный RS-триггер на элементах «И-НЕ» со статическим управлением, его таблица истинности, функциональная схема, объяснение принципа работы, УГО.

Таблица истинности:

C	S	R	Q_i	$\overline{Q_i}$	Пояснения
0	X	X	Q_{i-1}	$\overline{Q_{i-1}}$	Режим хранения информации
1	0	0	Q_{i-1}	$\overline{Q_{i-1}}$	Режим хранения информации
1	0	1	0	1	Сброс в 0
1	1	0	1	0	Установка 1
1	1	1	*	*	Неопределённое состояние

Функциональная схема: УГО



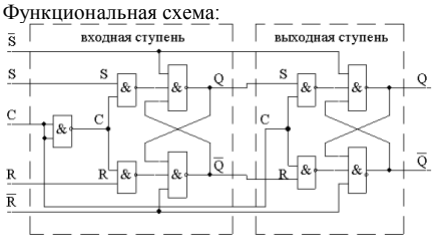
Принцип работы: пока разрешающий сигнал C = 0, триггер работает в режиме хранения информации. Какие бы сигналы не приходили на информационные входы, значение триггера не меняется. При разрешающем сигнале C = 1, триггер работает аналогично асинхронному RS-триггеру на элементах ИЛИ-НЕ (таблицы истинности

55. Триггеры: синхронный двухступенчатый RS-триггер с асинхронными входами и статическим управлением, его таблица истинности, функциональная схема, объяснение принципа работы, УГО.

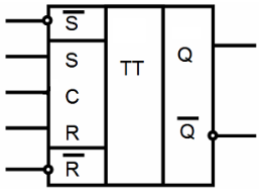
Таблица истинности:

C	S	nS	R	nR	Q_i	$\overline{Q_i}$	Пояснения
0	X	1	X	1	Q_{i-1}	$\overline{Q_{i-1}}$	Режим хранения информации
0	X	0	X	1	1	0	Установка в 1 асинхронно
0	X	1	X	0	0	1	Сброс в 0 асинхронно
0	X	0	X	0	*	*	Неопределённое состояние
X	0	1	0	1	Q_{i-1}	$\overline{Q_{i-1}}$	Режим хранения информации
0->	1	1	0	1	1	0	Установка в 1 синхронно
1							
0-> 1	0	1	1	1	0	1	Сброс в 0 синхронно
0-> 1	1	1	1	1	*	*	Неопределённое состояние

Функциональная схема:



Условное графическое обозначение (УГО):



*символ ТТ означает 2 ступени

Принцип работы:

Данный триггер имеет 2 режима работы: синхронный и асинхронный.

Асинхронные входы изменяют значение триггера независимо от сигнала синхронизации, аналогично асинхронному RS-триггеру на элементах И-НЕ (входы nS = R, nR = S)

Синхронный режим работы аналогичен синхронному статическому RS-триггеру, за исключением того, что смена значения происходит во время прихода переднего фронта на сигнал синхронизации.

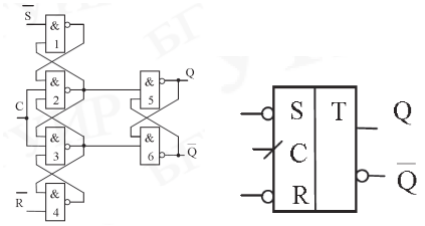
Способ синхронизации: по переднему фронту

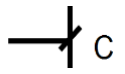
56.Триггеры: синхронный RS-триггер с динамическим управлением, его таблица истинности, функциональная схема, объяснение принципа работы, УГО.

Таблица истинности:

C	nS	nR	Q_i	$\overline{Q_i}$	Пояснения
0	X	X	Q_{i-1}	$\overline{Q_{i-1}}$	Режим хранения информации
1	X	X	Q_{i-1}	$\overline{Q_{i-1}}$	Режим хранения информации
1->0	X	X	Q_{i-1}	$\overline{Q_{i-1}}$	Режим хранения информации
0->1	0	0	*	*	Неопределённое состояние
0->1	1	0	0	1	Сброс в 0
0->1	0	1	1	0	Установка в 1
0->1	1	1	Q_{i-1}	$\overline{Q_{i-1}}$	Режим хранения информации

Функциональная схема:





*работает по переднему фронту

*работает по заднему фронту

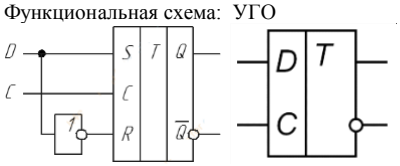
Принцип работы: смена значений происходит только при поступлении переднего фронта на синхросигнал. Работает аналогично асинхронному RS-триггеру на элементах И-НЕ (идентичные таблицы истинности).
Чтобы триггер работал по заднему фронту, необходимо подключить синхровход С через инверсию, т.е. через элемент И-НЕ.
Способ синхронизации: по фронту.

57. Триггеры: синхронный D-триггер, его таблица истинности, функциональная схема, объяснение принципа работы, УГО.

Принцип работы: триггер-задержка - хранит значение, пока С = 0. Когда С = 1, значение триггера равно значению информационного входа D

C	D	Q_i	$\overline{Q_i}$	Пояснения
0	0	Q_{i-1}	$\overline{Q_{i-1}}$	Режим хранения информации
0	1	Q_{i-1}	$\overline{Q_{i-1}}$	Режим хранения информации
1	0	0	1	Сброс в 0
1	1	1	0	Установка в 1

Функциональная схема: УГО



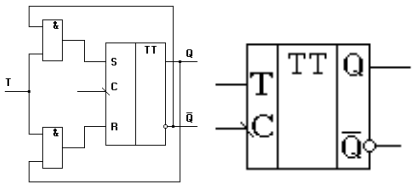
Способ синхронизации: по уровню синхронизирующего сигнала 1.

58. Триггеры: синхронный Т-триггер, его таблица истинности, функциональная схема, объяснение принципа работы, УГО.

Принцип работы: триггер-счетчик – с приходом очередного счетного импульса меняет свое состояние на противоположенное. В нашем случае синхровход С совпадает со счётным входом Т.
Способ синхронизации: по фронту.

C	T	Q_i	$\overline{Q_i}$
0	0	Q_{i-1}	$\overline{Q_{i-1}}$
0	1	Q_{i-1}	$\overline{Q_{i-1}}$
1	0	Q_{i-1}	$\overline{Q_{i-1}}$
1	1	$\overline{Q_{i-1}}$	Q_{i-1}

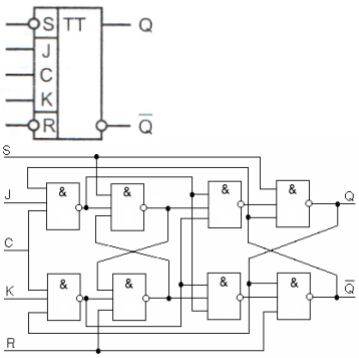
Функциональная схема: УГО



59. Триггеры: синхронный двухступенчатый JK-триггер с асинхронными входами предустановки, его таблица истинности, функциональная схема, объяснение принципа работы, УГО.

Таблица истинности:

C	J	n S	K	n R	Q_i	$\overline{Q_i}$	Пояснения
0	X	1	X	1	Q_{i-1}	$\overline{Q_{i-1}}$	Режим хранения информации
0	X	0	X	1	1	0	Установка в 1 асинхронно
0	X	1	X	0	0	1	Сброс в 0 асинхронно
0	X	0	X	0	*	*	Неопределённое состояние
X	0	1	0	1	Q_{i-1}	$\overline{Q_{i-1}}$	Режим хранения информации
0->1	1	1	0	1	1	0	Установка в 1 синхронно
0->1	0	1	1	1	0	1	Сброс в 0 синхронно
1->0	1	1	1	1			Режим Т-триггера



Универсальный триггер – отличается от RS-триггера тем, что не имеет запрещенных состояний. Имеет 3 режима работы: асинхронный, синхронный и Т-триггер.
Первые два аналогичны двухступенчатому синхронному RS-триггеру (J = S, K = R).
При J и K одновременно равных единице JK-триггер работает как Т-триггер, меняя свое состояние на противоположенное с приходом обратного фронта сигнала синхронизации.

60. Регистры: определение, выполняемые функции, классификация, виды сдвига.

Регистр- цифровой автомат, реализованный на триггерах, основным назначением которого является прием двоичной информации, временное хранение двоичной информации и выдача информации потребителю.
выполняемые функции:

- Параллельный прием
- Параллельная выдача
- Последовательный прием
- Последовательная выдача
- Преобразовывать последовательный код в параллельный и наоборот
- Быстро выполнять операции умножения и деления на 2

Классификация

- Количеству разрядов
- Триггерам, на которых они реализованы
- Способу приема и выдачи данных
  - Параллельные (простейшие регистры хранения)
  - Последовательные (последовательный прием, последовательная выдача)
  - Параллельно-последовательные (сдвиговый регистр)

виды сдвига:

- Логический (в освободившийся крайний триггер заносится 0)
- Арифметический (содержмое регистра понимается как число в дополнительном коде. При сдвиге влево справа появляется 0, при сдвиге вправо слева дублируется предыдущее значение)
- Циклический (вытесняемое из регистра значение заносится в освободившийся триггер на другом конце)

61.Четырехразрядный параллельный регистр на D-триггерах: УГО, внутреннее устройство, выполняемые функции.

- условное гос обозначение:

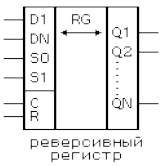
- внутреннее устройство:

- выполняемые функции:

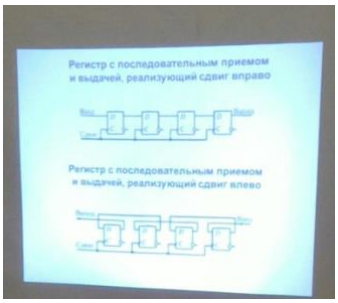
Регистры сдвига могут выполнять функции хранения и преобразования информации. Они могут быть использованы для построения умножителей и делителей чисел двоичной системы счисления, т. к. сдвиг двоичного числа влево на один разряд соответствует умножению его на два, а сдвиг вправо - делению на два.

62. Четырехразрядный регистр с последовательными приемом и выдачей на D-триггерах, с выбором направления сдвига: УГО, внутреннее устройство, варианты использования

- условное гос обозначение:?



- варианты использования: ?



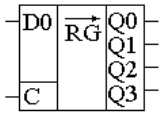
63.Четырехразрядный регистр с параллельно-последовательным приемом и выдачей, реализующий сдвиг вправо: УГО, внутреннее устройство, варианты использования.

- условное гос обозначение:?

- внутреннее устройство:

64 .Универсальный сдвиговый регистр: УГО, внутреннее устройство (на примере одного разряда), варианты использования.

- условное гос обозначение:



- внутреннее устройство:

- варианты использования:

65.Счетчики: определение, основные параметры, классификация.

Счетчик - это цифровой автомат, реализованный на триггерах. Предназначен для счета входных импульсов и хранения их количества в виде двоичного числа

Основные параметры:

Модуль счета M – максимальное количество единичных импульсов, которое может быть сосчитано счетчиком. Счетчик обнуляется, когда приходит -ый импульс.

Шаг счета – приращение значения счетчика при приходе очередного импульса.

Направление счета – в сторону увеличения или уменьшения значений.

Классификация:

По модулю счета:

- Двоичные
- Двоично-десятичные
- С постоянным модулем счета
- С переменным модулем счета

По шагу счета:

1,2... $K < M$

По направлению счета:

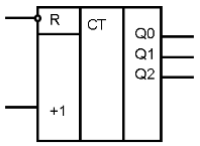
- Суммирующие
- Вычитающие
- Реверсивные

По способу организации межразрядных связей:

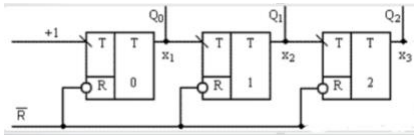
Счетчики с последовательным переносом (асинхронные счетчики), в которых переключение триггеров разрядных схем осуществляется последовательно один за другим;
Счетчики с параллельным переносом (синхронные счетчики), в которых переключение всех триггеров разрядных схем осуществляется одновременно по сигналу синхронизации;
Счетчики с комбинированным последовательно-параллельным переносом, при котором используются различные комбинации способов переноса.

66.Счетчики: трехразрядный суммирующий двоичный счетчик на Т-триггерах с последовательным переносом, его таблица истинности, УГО, функциональная схема достоинства и недостатки.

Q0(t)	Q1(t)	Q0(t)			
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0



:



+ счетчиков с последовательным переносом:
простота схемы;
наращивание разрядности осуществляется подключением нужного количества триггеров к выходу последнего триггера.

-:Сравнительно низкое быстродействие, т. к. триггеры последовательно срабатывают один за другим;из – за накопления временных сдвигов на выходах таких счетчиков могут появляться кратковременные ложные импульсы.

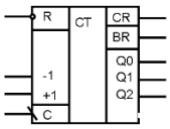
67.Счетчики: трехразрядный суммирующий двоичный счетчик на Т-триггерах с ускоренным переносом, его таблица истинности, УГО, функциональная схема достоинства и недостатки.

Таблица истинности:

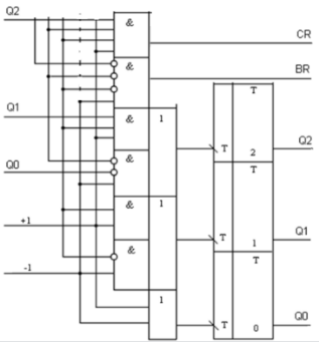
Q0(t)	Q1(t)	Q0(t)			
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1

1	1	1	0	0	0

- УГО



- функциональная схема:

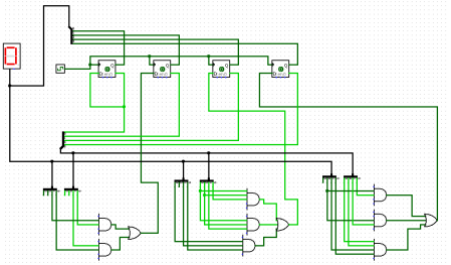


- достоинства и недостатки:
- Достоинства:
  - Легок в построении
- Недостатки
  - Требуется шифратор и дешифратор

68.Синтез оптимальных счетчиков с требуемым модулем, шагом и направлением на D-триггерах.

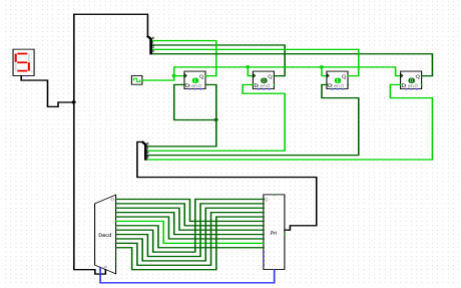
МДНФ

$\overline{abd} + \overline{abc}$ + acd $b\bar{d} + b\bar{c} + \overline{ab}cd$ $c\bar{d} + \bar{c}d$



69.Быстрый синтез

С помощью преобразователя кодов



70. Основы алгоритмизации. Понятие алгоритма, свойства алгоритмов.

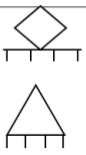
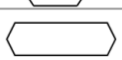
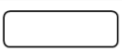
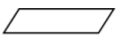
Алгоритм - это конечная последовательность правил по преобразованию исходных данных задачи в конечный результат

Свойства алгоритма

- Дискретность. Алгоритм-последовательность, состоящая из конечного числа команд. Вычислительный процесс должен быть разбит на последовательность раздельных шагов.
- Понятность. Алгоритм должен быть записан на языке исполнителя.

- Детерминированность. Алгоритм должен четко и понятно описывать вычислительный процесс.
- Результативность. Алгоритм применим к допустимым исходным данным, если с его помощью можно получить результат за конечное число шагов.
- Массовость. Применимость алгоритма к огромному множеству исходных данных.
- Корректность. Для всех допустимых исходных данных алгоритм позволяет получить правильный результат и ни для каких исходных данных не дает неправильный результат.
- Эффективность, надежность

71. Основы алгоритмизации. Понятие алгоритма, правила построения блок-схем.

Условие		**Проверка значение выражение и разветвление алгоритма**
Множественные условия	 Или	**Проверка значение выражение и разветвление алгоритма на более, чем три ветви.**
Границы цикла с предусловием или постусловием		Обозначает начало и конец цикла, где указываются условия цикла.
Цикл со счетчиком		Описывает начальное значение переменной, ее приращение и конечное значение
Соединитель		Нужен для соединения линий алгоритма при их разрыве или переходе на другую страницу
Комментарий		Применятся для описания набора блоков
Начало-конец (пуск-остановка)		**Обозначает начало и конец программы или подпрограммы**
Действие		**Выполнение одной или нескольких операций**
Ввод/вывод		**Операции ввода или вывода даных**
Предопределенный процесс		**Вызов подпрограммы**

72. **Основы алгоритмизации. Алгоритм поиска максимума и минимума**.

```
Минимум:
min=A[0]
i=1
ПОКА (i<n)
   ЕСЛИ (min<A[i]) ТО
      min=A[i]
   i=i+1
```

73. Основы алгоритмизации. Принцип структурного программирования Дейкстры.

- Следует отказаться от использования оператора безусловного перехода.
- Любая программа строится на основе трех базовых конструкций: действие, условие, цикл.
- Базовые конструкции могут быть вложены друг друга в произвольной форме и неограниченном количестве.
- Повторяющиеся фрагменты желательно оформлять в виде подпрограмм.
- Логически законченные группы инструкций желательно объединять в блоки.
- Все перечисленные конструкции должны иметь один вход и один выход.
- Разработка программы ведется пошагово, используя метод «сверху-вниз».

74. Основы алгоритмизации. Алгоритм сортировки «Пузырек».

```
i=0
ПОКА (i<n)
   j=i+1
   ПОКА (j<n)
      ЕСЛИ (A[i]>A[j]) ТО //сортируем по возрастанию
            врем=A[i]
            A[i]=A[j]
            A[j]=врем
      j=j+1
   i=i+1
```

75. Основы алгоритмизации. Алгоритм быстрой сортировки Хоара.

Обобщенный алгоритм быстрой сортировки:

- Выбирается опорный элемент.
- Производится разделение массива:
  - Выбираются два индекса l и r в которые заносятся значения левой и правой границ массива.
  - Индекс l увеличивается пока l-ый элемент не будет больше или равен опорному.
  - Индекс r уменьшается пока r-ый элемент не будет меньше или равен опорному.
  - Если l равен r, то операция разделения закончена. Иначе возвращаемся к шагу 2.2.
- Рекурсивно упорядочиваются полученные в результате разделения подмассивы. База рекурсии – пустой массив или массив из одного элемента.

```
procedure  Qsort(var x: array of integer;l,r:Integer) ;
 var
 p,i,j,temp:Integer;
  begin
    p:=x[(r - ((r - l) div 2))];
    i:=l;j:=r;
       while (i <= j) do
       begin
             while (x[i] < p) do inc(i);
              while (x[i] < p) do Dec(j);
              if (i <= j) then
              begin
                 temp:= x[i];
                 x[i] := x[j];
                 x[j] := temp;
                 inc(i);
                 Dec(j);
              end;
          if (i < r) then qSort(x,i, r);
          if (l < j) then qSort(x,l, j);
       end;
  end;
```

76. Основы алгоритмизации. Рекурсия – назначение, виды, примеры организации.

Рекурсией называется вызов функцией самой себя с некоторым изменением входных параметров.

Главным условием для использования рекурсии в функциях является наличие базы рекурсии – то есть значения, которое зависит только от входного параметра и позволяет выйти из рекурсии.

Еще одним ограничивающим фактором применения простой рекурсии является объем специальной области памяти, где хранится контекст и адреса возврата при вызове функций. При большой глубине рекурсии эта память может быть переполнена, что приведет к ошибке выполнения программы.

Вид. **хвостовая**(бесконечная)-для ее работы нужно, чтобы рекурсивный вызов был последним в теле рекурсивной ф-ции, и чтобы компилятор умели обнаруживать подобный вызов

77. Основы алгоритмизации. Проверка вводимых данных – типичные ошибки и методы борьбы с ними.

- Все операнды функций, имеющих ограничения на область допустимых значений (ОДЗ), должны в обязательном порядке проверяться на соответствие ОДЗ.
- Память данных и команд в современных ЭВМ в большинстве случаем не разделена и слабо защищена от попадания в чужую область памяти.
- Так, объявив массив на 5 элементов можно попытаться обратиться к 6, 10, -5 элементам попав в чужую область памяти и исказив как данные, так и программные коды.
- Аналогичные ошибки бывают при копировании данных:
- Запрос пользователю: Введите имя
- Скопировать в строку введенное имя
- При этом размер копируемой области памяти в большинстве случаев определяется длинной введенной строки, без учета размера строки получателя, что приведет к переполнению, если пользователь ввел очень большое имя.
- При любых операциях с массивами и памятью необходимо контролировать выход индексных переменных за границы.
- При копировании блоков памяти необходимо проверять как размер копируемого блока, так и размер области получателя выбирая минимальное из них или отказываясь от копирования при превышении допустимого объема.
- Как уже упоминалось знаковые и беззнаковые переменные это разная интерпретация одних и тех же значений разрядной сетки числа.
- Нельзя использовать в одном выражении знаковые и беззнаковые переменные, так как это может привести к непредсказуемой их интерпретации и ошибкам, как следствие того, что знаковое число воспринимается как большое беззнаковое и наоборот.
- Перед вычислениями знаковые и беззнаковые числа должны быть принудительно приведены к одной форме.