Рабочая тетрадь № 6

|  |
| --- |
| Релейно-контактная схема представляет собой схематическое изображение некоторого устройства, состоящего из переключателей, соединяющих их проводников, входов в схему и выходов из нее. В релейно-контактной схеме любой переключатель имеет только два состояния: замкнутое и разомкнутое. |

|  |
| --- |
| **1. Теоретический материал** |
| Простейшая схема состоит из одного переключателя X, одного входа и одного выхода. Переключателю X ставится в соответствие высказывание x – «Переключатель X замкнут». Если x истинно, то схема проводит электрический ток, в противном случае – не проводит. Если принимать во внимание только значение высказывания, то любому высказыванию x можно поставить в соответствие следующую двухполюсную релейно-контактную схему    Отрицание высказывания x будем изображать двухполюсной схемой    Конъюнкцию двух высказываний x и y можно представить двухполюсной схемой с последовательным соединением переключателей X и Y    а дизъюнкцию – двухполюсной схемой с их параллельным соединением    Любая формула алгебры логики может быть представлена в нормальной форме, следовательно, любой формуле алгебры логики можно поставить в соответствие некоторую релейно-контактную схему, и наоборот, каждой релейно-контактной схеме можно поставить в соответствие некоторую формулу алгебры логики. |

|  |  |
| --- | --- |
| **2. Пример** | |
| ***Задача:*** | |
|  | Дана логическая формула . Составьте для неё релейно-контактную схему. |
| ***Решение:*** | |
|  | Упросим логическую формулу:  Таким образом, релейно-контактная схема будет иметь следующий вид: |
| ***Задача:*** | |
|  | Упростить релейно-контактную схему: |
| ***Решение:*** | |
|  | Заменим релейно-контактную схему на логическую формулу:  и упростим её:  Теперь заменим упрощённую логическую формулу обратно на релейно-контактную схему: |
| ***Задача:*** | |
|  | Составить релейно-контактную схему, соответствующую импликации. |
| ***Решение:*** | |
|  | Выразим операцию импликации через основные логические операции «И», «ИЛИ» и «НЕ»:  Построим по логической формуле контактно-релейную схему: |
| ***Задача:*** | |
|  | Дана логическая формула:  Составьте для неё релейно-контактную схему. |
| ***Решение:*** | |
|  | Упростим логическую формулу с помощью равносильных преобразований:  Составим по упрощённой логической формуле контактно-релейную схему: |
| ***Задача:*** | |
|  | Построить релейно-контактную схему, соответствующую *f(x, y, z)*, если известно, что: *f(1, 0, 0) = f(0, 1, 1) = f(0, 1, 0) = f(0, 0, 0) = 1.*  Для остальных комбинаций аргументов значения функции равны нулю. |
| ***Решение:*** | |
|  | Составим таблицу истинности для и запишем по ней СДНФ.   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |  | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | |  | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |  | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |   Упростим получившуюся СДНФ:  Теперь построим контактно-релейную схему. |

|  |  |
| --- | --- |
| ***Задача:*** | |
|  | Упростить релейно-контактную схему: |
| ***Решение:*** | |
|  | По контактно-релейной схеме запишем логическую формулу и далее упростим её:  Составим по упрощённой логической формуле контактно-релейную схему. |
| ***Задача:*** | |
|  | Доказать равносильность следующих релейно-контактных схем:    и |
| ***Решение:*** | |
|  | Запишем для каждой контактно-релейной схему логическую формулу и упростим их для дальнейшего сравнения.  Видно, что , а следовательно они эквиваленты. Отсюда контактно-релейные схемы также эквивалентны. |

|  |  |  |
| --- | --- | --- |
| **3. Задания** | | |
| 1. | ***Задача:*** | |
|  | Составить релейно-контактную схему, соответствующую эквивалентности. |
| ***Решение:*** | |
|  |  |
| 2. | ***Задача:*** | |
|  | Составить релейно-контактные схемы, соответствующие следующим формулам:  1) ;  2) . |
| ***Решение:*** | |
|  | 1)  2) |

|  |  |  |
| --- | --- | --- |
| 3. | ***Задача:*** | |
|  | Построить релейно-контактные схемы для логических функций если известно, что:  1) ;  2) ;  3) остальные значения функций равны нулю. |
| ***Решение:*** | |
|  | 1)  2) |
| 4. | ***Задача:*** | |
|  | Упростите релейно-контактную схему: |
| ***Решение:*** | |
|  |  |
| 5. | ***Задача:*** | |
|  | Докажите эквивалентность следующих релейно-контактных схем: |
| ***Решение:*** | |
|  | f(x, y, z) = (x and y) or (not x and y and not z) or (y and z)  при y = 0 => 0 or 0 or 0  при y = 1 => 1 т. к. не зависит от x и z |

|  |
| --- |
| **1. Теоретический материал** |
| Любое вычислительное устройство состоит из набора так называемых «вентилей» – минимальных логических устройств, способных преобразовывать входные сигналы в выходные. Ниже в таблице 1 представлены основные обозначения логических вентилей в различных стандартах.  Таблица 1 – Логические вентили и их условные графические обозначение в стандартах «ГОСТ 2.743-91», «IEC 60617-12: 1997» и «US ANSI 91-1984»   |  |  |  |  | | --- | --- | --- | --- | | **Название вентиля** | **Условные графические обозначения (УГО)** | | | | **ГОСТ 2.743-91** | **IEC 60617-12: 1997** | **US ANSI 91-1984** | | **НЕ** | NOT gate RU.svg | IEC NOT label.svg | Not-gate-en.svg | | **И** | AND gate RU.svg | IEC AND label.svg | Logic-gate-and-us.svg | | **ИЛИ** | OR gate RU.svg | IEC OR label.svg | Or-gate-en.svg | | **НЕ И** | NAND gate RU.svg | IEC NAND label.svg | Nand-gate-en.svg | | **НЕ ИЛИ** | NOR gate RU.svg | IEC NOR label.svg | Nor-gate-en.svg | | **исключающее ИЛИ** | XOR gate RU.svg | IEC XOR label.svg | Xor-gate-en.svg |   Маленькие кружочки в УГО обозначают инверсию сигнала (т.е. отрицание НЕ). Они могут стоять и на входе вентиля, обозначая инверсию входного сигнала.  Рассмотрим первое простое устройство, которое можно собрать на вентилях: **полусумматор**. Его таблица истинности и схема представлена на рисунке 1.    Рисунок 1 – Таблица истинности для сложения одноразрядных чисел  и схема полусумматора  Полусумматор предназначен для нахождения суммы двух бит. При необходимости сложения чисел, состоящих из двух и более разрядов, эта схема уже не работает (нельзя просто взять необходимое число полусумматоров, т.к. в этой схеме не предусмотрен учёт переноса из младшего разряда в старший). Для этого нужно уже соединять между собой так называемые **полные сумматоры**. Таблица истинности и схема полного сумматора представлена на рисунке 2.    Рисунок 2 – Таблица истинности для полного сумматора (а);  схема для полного сумматора (б)  В вычислительной технике также можно встретить применение **мультиплексоров**. Мультиплексор имеет один выход и две группы входных линий: информационные и адресные. Код, подаваемый на адресные линии, определяет, какой из информационных входов в данный момент подключен к выходному выводу. Поскольку n-разрядный двоичный код может принимать 2n значений, то если число адресных входов мультиплексора равно n, число его информационных входов должно равняться 2n. УГО мультиплексора с 8 информационными входными линиями , 3 адресными входными линиями и 1 выходной линией приведено на рисунке 3.    Рисунок 3 – Мультиплексор 8 в 1  Используя мультиплексор можно реализовывать логические функции (пример ниже). |

|  |  |  |
| --- | --- | --- |
| **2. Пример** | | |
| 1. | ***Задача:*** | |
|  | Нарисовать схему (от руки или в любом графическом редакторе), реализующую следующую логическую функцию: |
| ***Решение:*** | |
|  | C:\Users\Qverty\g3488.png |
| 2. | ***Задача:*** | |
|  | Записать логическую формулу (формулы) и таблицу истинности для следующей схемы:  C:\Users\Qverty\g3488-6.png |
| ***Решение:*** | |
|  | ,  .   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | **A** | **B** |  |  |  |  |  |  | | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | |

|  |  |  |
| --- | --- | --- |
| 3. | ***Задача:*** | |
|  | Нарисовать схему (от руки или в любом графическом редакторе), позволяющую суммировать трёхбитовые двоичные числа (, , и C). |
| ***Решение:*** | |
|  | C:\Users\Qverty\rect2970-2.png |
| 4. | ***Задача:*** | |
|  | Используя мультиплексор, нарисовать схему (от руки или в любом графическом редакторе), реализующую функцию большинства от трёх логических переменных. |
| ***Решение:*** | |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Таблица истинности для функции большинства** | | | | **Схема, реализующая необходимую логическую функцию** | | **A** | **B** | **C** | **F** |  | | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 0 | | 0 | 1 | 0 | 0 | | 0 | 1 | 1 | 1 | | 1 | 0 | 0 | 0 | | 1 | 0 | 1 | 1 | | 1 | 1 | 0 | 1 | | 1 | 1 | 1 | 1 | |

|  |  |  |
| --- | --- | --- |
| **3. Задания** | | |
| 1. | ***Задача:*** | |
|  | Нарисовать схему (от руки или в любом графическом редакторе), реализующую следующую логическую функцию: |
| ***Решение:*** | |
|  |  |
| 2. | ***Задача:*** | |
|  | Записать логическую формулу (формулы) и таблицу истинности для следующей схемы:  C:\Users\Qverty\g3907.png |
| ***Решение:*** | |
|  | F0 = ((not A & B) and (A or B)) or A  F1 = not (A or B)   |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | A | B | Not A | Not A & B | A or B | (Not A & B) and (A or B) | not((Not A & B) and (A or B)) | F0 | F1 | | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | |
| 3. | ***Задача:*** | |
|  | Используя мультиплексор, нарисовать схему (от руки или в любом графическом редакторе), реализующую функцию импликации . |
| ***Решение:*** | |
|  |  |
| 4. | ***Задача:*** | |
|  | Нарисовать схему (от руки или в любом графическом редакторе), позволяющую умножать двухбитовые двоичные числа. |
| ***Решение:*** | |
|  |  |

|  |  |  |
| --- | --- | --- |
| **Тест 6** | | |
| **1.** | Для сложения многоразрядных двоичных чисел используется последовательность соединённых между собой: + | |
|  | а) мультиплексоров; б) триггеров;  в) полусумматоров; **г) полных сумматоров.** |
| ***Ответ:*** | |
|  |  |
| **2.** | Устройство, выполняющее базовые логические операции, называется: + | |
|  | а) реестр; б) ячейка; **в) вентиль;**  г) триггер |
| ***Ответ:*** | |
|  |  |
| **3.** | Графическое изображение логического выражения называется: + | |
|  | **а) схема** б) рисунок в) чертеж г) график |
| ***Ответ:*** | |
|  |  |
| **4.** | Электронная схема, применяемая в регистрах компьютера для запоминания одного разряда двоичного кода это … + | |
|  | а) вентиль б) логическая схема  **в) триггер** г) электронная схема |
| ***Ответ:*** | |
|  |  |
| **5.** | Мультиплексор – это устройство для … + | |
|  | а) умножения чисел  **б) коммутации одного из информационных входов с единственным выходом на основе кода, подаваемого на адресные входы**  в) произведения операции сдвига на основе данных на информационных входах и кода на адресных входах  г) хранения одного бита информации |
| ***Ответ:*** | |
|  |  |
| **6.** | Какому логическому выражению соответствует схема на рисунке? | |
|  | а) б)  в) **г)** |
| ***Ответ:*** | |
|  | ГГГ |
| **7.** | Какой минимально необходимо взять мультиплексор, чтобы реализовать заданную таблицей истинности логическую функцию от 4 логических переменных? | |
|  | а) с 4 информационными входными линиями, 2 адресными входными линиями и 1 выходной линией  **б)** с 16 информационными входными линиями, 4 адресными входными линиями и 1 выходной линией  в) с 4 информационными входными линиями, 4 адресными входными линиями и 2 выходными линиями  г) с 4 информационными входными линиями, 16 адресными входными линиями и 2 выходными линиями |
| ***Ответ:*** | |
|  |  |
| **8.** | Такой структурной схемой обозначается  XOR gate RU.svg | |
|  | **а) исключающее ИЛИ** + б) дизъюнктор  в) бинарное НЕ г) инвертор |
| ***Ответ:*** | |
|  |  |
| **9.** | Равносильны или нет следующие релейно-контактные схемы? | |
|  | https://fsd.videouroki.net/html/2011/03/31/98657307/98657307_14.png |
| ***Ответ:*** | |
|  | нет |

|  |  |  |
| --- | --- | --- |
| **10.** | Какому логическому выражению соответствует схема на рисунке? | |
|  | а) б)  в) **г)** |
| ***Ответ:*** | |
|  | Г |

Реализация задач на языке программирования Python

|  |
| --- |
| В языке Python имеются множество удобных контейнеров для хранения различной информации: строки, списки, кортежи, множества и словари. Их использование обеспечивает эффективность решения широкого спектра задач. |

|  |
| --- |
| **1. Теоретический материал** |
| Строки - это последовательности символов. Создать строки можно следующим образом:  s = "Hello, World!"  p = 'Миру - мир!'  Доступ к символу с индексом n (начиная с нуля) осуществляется с помощью выражения s[n]. Подстроку можно выделить, указав интервал s[n:m]. Такая возможность в языке называется срезом. Пример:  s = "Мамамылараму"  print(s[6])  p = s[5:9]  print(p)  Если индекс отрицательный, то отсчет идет от конца строки. При этом индекс -1 соответствует последнему символу. Строки в Python не изменяемы, т.е. по индексу можно только получить символ, но не поменять его!  Поскольку строка задается с помощью кавычек (двойных или одинарных), то для того, чтобы в самой строке использовать символы кавычек, их необходимо экранировать следующим образом:  s = "А Балда приговаривал с укоризной: \"Не гонялся бы ты, поп, за дешевизной\""  print(s)  Также можно воспользоваться одинарными кавычками, чтобы можно было использовать двойные (и наоборот):  s = 'А Балда приговаривал с укоризной: "Не гонялся бы ты, поп, за дешевизной"'  print(s)  В строках можно использовать специальные символы (управляющие последовательности), которые задается следующим образом:  \n – новая строка;  \t – табуляция.  Пример красивого вывода:  s = "\n\n\t\tА Балда приговаривал с укоризной:\n \  \t\t\"Не гонялся бы ты, поп, за дешевизной\"\n\n"  print(s)  Со строками можно выполнять операции конкатенации с помощью оператора "+", который мы уже проходили. Строку можно умножить на целое число **n**, в результате чего строка будет повторена **n** раз:  s = "Мама"  s = s \* 3  print(s)  Узнать длину строки можно с помощью функции **len(s)**.  Часто возникает задача поиска подстроки в строке. Для этих целей используется функция **s.find(str)**:  s = "Жили у Бабуси два веселых гуся"  n = s.find("два")  print(n)  С помощью функции **s.split(str)** строки можно разделять по заданному символу:  s = "Мама|Папа|Я"  S = s.split('|')  print(S[1])  Полный список функций для строк:   * [https://pythonworld](https://pythonworld.ru/tipy-dannyx-v-python/stroki-funkcii-i-metody-strok.html).ru/tipy-dannyx-v-python/stroki-funkcii-i-metody-strok.html |

|  |  |
| --- | --- |
| **2. Пример** | |
| ***Задача:*** | |
|  | Удалить все пробелы из данной строки. |
| ***Решение (код программы):*** | |
|  | txt = "I like bananas"  x = txt.replace(" ", "")  print(x) |

|  |  |  |
| --- | --- | --- |
| **3. Задания** | | |
| 1. | ***Задача:*** | |
|  | Пользователь вводит произвольное слово. Необходимо проверить, что оно является палиндромом. **Палиндромом** называется слово, которое одинаково читается как слева направо, так и справа налево, например, в английском языке такими словами являются "radar" и "racecar". |
| ***Решение (код программы):*** | |
|  | s = input()  print(s[-1::-1])  if s == s[-1::-1]:  print("Палиндром")  else:  print("Не палиндром") |
| 2. | ***Задача:*** | |
|  | Пользователь вводит строку. Необходимо вывести её на экран в нижнем регистре. |
| ***Решение (код программы):*** | |
|  | S = input()  print(s.lower()) |

|  |  |  |
| --- | --- | --- |
| 3. | ***Задача:*** | |
|  | Пользователь вводит строку. Необходимо вывести её на экран в верхнем регистре. |
| ***Решение (код программы):*** | |
|  | S = input()  print(s.upper()) |
| 4. | ***Задача:*** | |
|  | Пользователь вводит строку. Необходимо проверить есть ли в ней числа (цифры). |
| ***Решение (код программы):*** | |
|  | s = input()  flag = 1  for i in s:  if i not in "0123456789":  print("Have number(s)")  flag = 0  break  if flag == 1:  print("Have not number") |
| 5. | ***Задача:*** | |
|  | Задана строка. Необходимо между её символами вставить по одному пробелу. |
| ***Решение (код программы):*** | |
|  | s = input()  h = ""  for i in range(len(s) - 1):  h = h + s[i] + " "  h = h + s[-1]  print(h) |
| 6. | ***Задача:*** | |
|  | Дан символ C. Вывести его код (то есть номер в кодовой таблице). |
| ***Решение (код программы):*** | |
|  | C = input()  print(ord(c)) |

|  |
| --- |
| **1. Теоретический материал** |
| В Python списки используются для хранения данных и позволяют получать доступ к ним по индексу (их порядковому номеру). Другими словами списки в Python реализуют такую структуру данных как массивы, но добавляют и некоторые специфические для себя операции.  A = [] # пустой список  B = [1, 1, 2, 3, 5, 8, 13, 21] # числа Фибоначчи  C = ["мама", 120, 'z', "список"] # смешанный список  D = [[1, 2], ['a', 'b', 'c'], [[3.14, "abc"], [120, 140]]] # список списков  К элементам списка можно обращаться через индексы, которые начинаются с нуля. В отличии от строк, элементы списков можно заменять:  A = [1, 2, 3]  A[1] = "новое значение"  print(A)  Для работы со списками существует большое количество методов. Вот основные:   |  |  | | --- | --- | | **Метод** | **Описание** | | list.append(x) | добавляет элемент в конец списка | | list.insert(i, x) | вставляет на i-ый элемент значение x | | list.count(x) | возвращает количество элементов со значением x | | list.reverse() | разворачивает список | | list.clear() | очищает список |   В Python есть неизменяемый аналог рассмотренного выше списка: кортежи. За счёт неизменяемости интерпретатор Python работает с кортежами эффективнее. Создать кортеж можно следующим образом:  A = (“Ivanov", “Ivan", 1988)  Surname, Name, Date = A  print(Surname + " " + Name + " " + Date.\_\_str\_\_()) |

|  |  |
| --- | --- |
| **2. Пример** | |
| ***Задача:*** | |
|  | Заполните список 20 случайными числами. |
| ***Решение (код программы):*** | |
|  | import random  a = []  for i in range(1,21):  a.append(random.randint(5,10))  print(a) |

|  |  |
| --- | --- |
| ***Задача:*** | |
|  | Напишите программу, которая запрашивает с ввода восемь чисел, добавляет их в список. На экран выводит их сумму, максимальное и минимальное из них. Для нахождения суммы, максимума и минимума воспользуйтесь встроенными в Python функциями sum(), max() и min(). |
| ***Решение (код программы):*** | |
|  | l = []  for i in range(0,8):  a = int(input())  l.append(a)  print(l)  print(sum(l))  print(min(l))  print(max(l)) |

|  |  |  |
| --- | --- | --- |
| **3. Задания** | | |
| 1. | ***Задача:*** | |
|  | Если объект range (диапазон) передать встроенной в Python функции list(), то она преобразует его к списку. Создайте, таким образом, список с элементами от 0 до 100 и шагом 17. |
| ***Решение (код программы):*** | |
|  | L = list(range(0, 101, 17)) |
| 2. | ***Задача:*** | |
|  | Напишите программу, которая заполняет список пятью словами, введенными с клавиатуры, измеряет длину каждого слова и добавляет полученное значение в другой список. Например, список слов – ['yes', 'no', 'maybe', 'ok', 'what'], список длин – [3, 2, 5, 2, 4]. Оба списка должны выводиться на экран. |
| ***Решение (код программы):*** | |
|  |  |

|  |
| --- |
| **1. Теоретический материал** |
| Теперь рассмотрим множества в Python. Множества отличаются от списков и кортежей тем, что хранящиеся в них объекты являются неупорядоченными. Кроме того, множества содержат только уникальные элементы. Зато с множествами можно производить стандартные операции объединения, пресечения и разности  Кроме того, множества содержат только уникальные элементы. Зато с множествами можно производить стандартные операции объединения, пресечения и разности:   |  |  | | --- | --- | | **Операция** | **Описание операции** | | | | объединение | | & | пересечение | | - | разность | | ^ | симметричная разность |   Множество можно создать с помощью оператора **set()**:  A = set(['a', "apple", 120])  Добавить один элемент в множество можно с помощью метода **A.add(x). Ч**тобы добавить другое множество используется операция **A.update(conteiner).** Для удаления элементов используется метод **A.remove(x)**. Примеры работы с множествами:  A = set(["apple", "orange"])  A.add("pear")  B = set(["apple", "pear", "banana"])  C = A | B  D = A&B  print(C)  print(D)  В Python есть контейнер, который называется словарем. Словари позволяют хранить связки «ключ–значение» и осуществлять доступ к элементам по ключу. Пример:  people = {  "Name": “Ivan”,  "Surname": “Ivanov",  "Date": 1988  }  print(people)  people["Name"] = “Alex"  people["Date"] = 2008  print(people) |

|  |  |  |
| --- | --- | --- |
| **2. Задания** | | |
| 1. | ***Задача:*** | |
|  | Даны два множества X и Y. Найти:  1) их объединение  2) их пересечение  3) элементы множества A, которых нет во множестве B |
| ***Решение (код программы):*** | |
|  | X = set(["a", "b", "c", "d", "e", "f"])  Y = set(["e", "a", "g", "b", "c", "d"])  Z = X.union(Y)  F = X.difference(Y)  print(Z)  print(X and Y)  print(F) |