Г.Лазарев

Учитесь программировать

2023 г

ОГЛАВЛЕНИЕ

п.п Название Слайд

ПРЕДИСЛОВИЕ…….. 3

1. ЭТАПЫ РАЗРАОТКИ ПРОГРАММЫ…….. 4

2. ПРИМЕР РАЗРАБОТКИ ПРОГРАММЫ…….. 5

2.1. Постановка задачи…….. 6

2.2. План решения задачи ……. 7

2.3. Разработка структур данных ……. 8

2.4. Разработка алгоритмов ……. 9

2.4.1. Элементы блок-схем …… 10

2.4.2. Алгоритм модуля “Вычисление СВВ” ……. 11

2.4.2.1. Блок-схема …… 11

2.4.2.2. Описание алгоритма ……12

2.4.3. Алгоритм модуля “Замена шифров параметров на значения” ……..13

2.4.4. Алгоритм модуля “Поиск и вычисление ПВВ” ……. 14

2.4.4.1. Блок-схема ……14

2.4.4.2. Описание алгоритма …… 15

3. НАПИСАНИЕ ПРОГРАММЫ В ОПЕРАТОРАХ ЯЗЫКА ПРОГРАММИРОВАНИЯ ……16

3.1. Операторы языка“PYTHON” (ПИТОН), которые используются в нашей программе …….. 16

3.1.1. Общие знаки …….. 16

3.1.2. Стандартные функции ……..17

3.1.3. Структуры данных ……18

3.1.4. Средства управления логикой ……21

3.1.5. Модули …….. 22

3.2. ТЕКСТЫ ПРОГРАММЫ …… 23

3.2.1. Модуль “Вычисление СВВ” ……23

3.2.2. Модуль “Замена шифров параметров на значения” …… 25

3.2.3. Модуль “Поиск и вычисление ПВВ” …… 27

3.2.4. Отладочная печать …….. 32

4. Объектно-ориентированное программирование…. 34

5. Системы Управления Базами Данных …….. 36

ПРЕДИСЛОВИЕ

Я решил поделиться своим многолетним опытом в программировании со своими внуками, чтобы отвлечь их от чрезмерного увлечения играми на компьютере и направить их энергию в конструктивное русло. Может быть сейчас им этот материал читать рано. Но надеюсь, что в ближайшем будущем они созреют для этого. Сейчас достаточно того, что старший внук начинает осваивать средства проектирования уже своих игр.

В данном материале я хочу продемонстрировать процесс программирования на универсальных языках от постановки задачи и построения алгоритмов до написания и отладки программы на конкретном примере. Когда пишут учебники или курсы по программированию, как правило, концентрируются на изучении какого-то языка программирования. Мой более, чем 40-летний опыт программирования, показывает, что языки программирования быстро устаревают и им на смену приходят новые языки.

Поэтому процесс программирования будем рассматривать как последовательность необходимых этапов, которые не зависят от языка программирования. Конечно, чем более мощный язык или технология, тем легче выполнять этап кодирования в операторах языка программирования. Но надо учитывать, что по мере совершенствования языков - усложняются и задачи, которые надо решать.

Речь идет об универсальных языках программирования, а не о проблемно-ориентированных технологических комплексах. Например, популярный инструмент бухгалтеров EXCEL – электронные таблицы, не требует знания языка и технологии программирования, а оперирует профессиональными понятиями экономистов. Но даже там, для увеличения возможностей, предусмотрена вставка макросов на универсальном языке VBA – визуальном бейсике.

II. Этапы разработки программы

1. Постановка задачи
2. План решения задачи
3. Разработка структур данных
4. Разработка алгоритмов

4.1. Элементы блок-схем

4.2.1. Блок-схема программы на уровне типовых процедур

4.2.2. Описание программы на уровне типовых процедур

4.3. Алгоритмы отдельных типовых процедур

4.3.1. Блок-схема процедуры “Выделение очередного ПВВ”

4.3.2. Описание процедуры “Выделение очередного ПВВ”

5. Написание программы в операторах языка программирования

6. Отладка программы

7. Тестирование программы

Рассмотрим все этапы разработки программы на примере разработки программы вычисления составного выражения

II. ПРИМЕР РАЗРАБОТКИ ПРОГРАММЫ

2.1. Постановка задачи

Разработать программу вычисления составного выражения:

- *Вычисляемое выражение задается в виде символьной строки*
- *В вычисляемом выражении могут использоваться только операции:*

  “+” – сложение; “-” – вычитание; “” – умножение; “/” - деление*
- *В вычисляемом выражении могут использоваться круглые скобки для задания последовательности вычисления*
- *В вычисляемом выражении заданы переменные параметры (парN), которые в процессе обработки надо заменить на указанные числовые значения. Эти параметры выделены фигурными скобками {…}*
- *В качестве элементов в вычисляемом выражении могут использоваться результаты вычислений других вычисляемых выражений и обычные константы*

2.2. План решения задачи

Обозначения:

СВВ *- составное вычисляемое выражение, которое содержит несколько простых вычисляемых выражений*

ПВВ *– простое вычисляемое выражение, которое не содержит внутри круглых скобок*

ПАРn*– символьное имя переменного параметра*

ЗНЧn*– числовое значение переменного параметра*

- *Пример СВВ: (40 + ({ пар1} – (8 + ({ пар2 } * 4 + { пар3 } /2)))*
- *пар1=23 пар2=2 пар3=8*
- *Последовательность вычисления:*

1. *(40 + (23 – (8 + (2 * 4 + 8/2))) - замена шифров параметров на числовые значения*
2. *(40 + (23 – (8 + 12))) - вычисление простых выражений*
3. *(40 + (23 – 20))*
4. *(40 + 3) - вычисление конечного выражения (СВВ превратилось в ПВВ)*
5. *43*

2.3. Разработка структур данных

Для хранения исходных, промежуточных и окончательных вычислений предусмотрим соответствующие области данных

Из условия задачи исходное выражение должны быть символьной строкой. Следовательно промежуточные и окончательные результаты будем размещать тоже в символьных строках:

***ИСХСВВ** – строка для исходного выражения*

***БУФПВВ** – буферная память для хранения очередного ПВВ*

БУФСВВ – буферная строка для формирования окончательного результата

СЛОВАРЬ – словарь для хранения значений параметров

ОЧЕРЕДЬ – для временного хранения ПВВ и замены формул значениями

2.4. Разработка алгоритмов

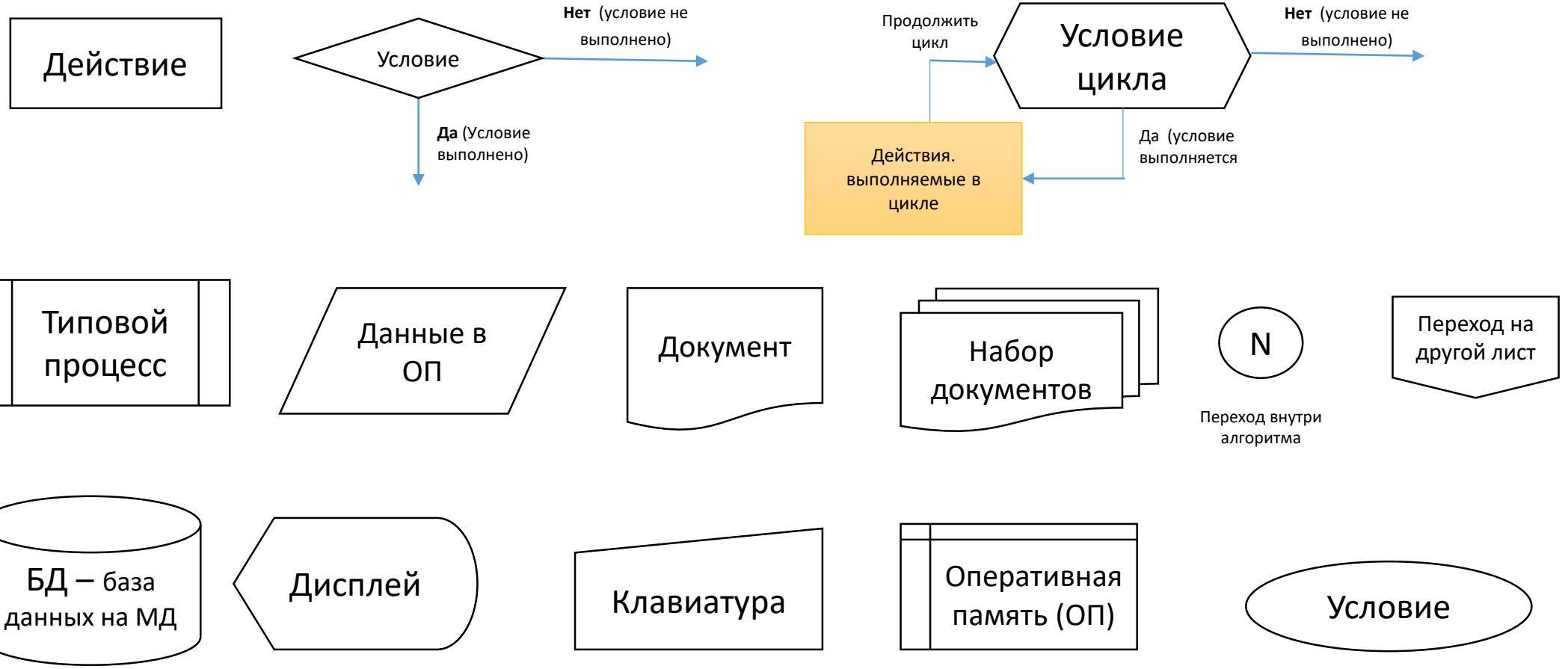
Этап разработки алгоритмов является очень важным. На нем отрабатывается методы и последовательность решение поставленной задачи.

Алгоритмы могут быть оформлены в виде:

- Блок-схем
- Ориентированных графов
- Текстовом описании

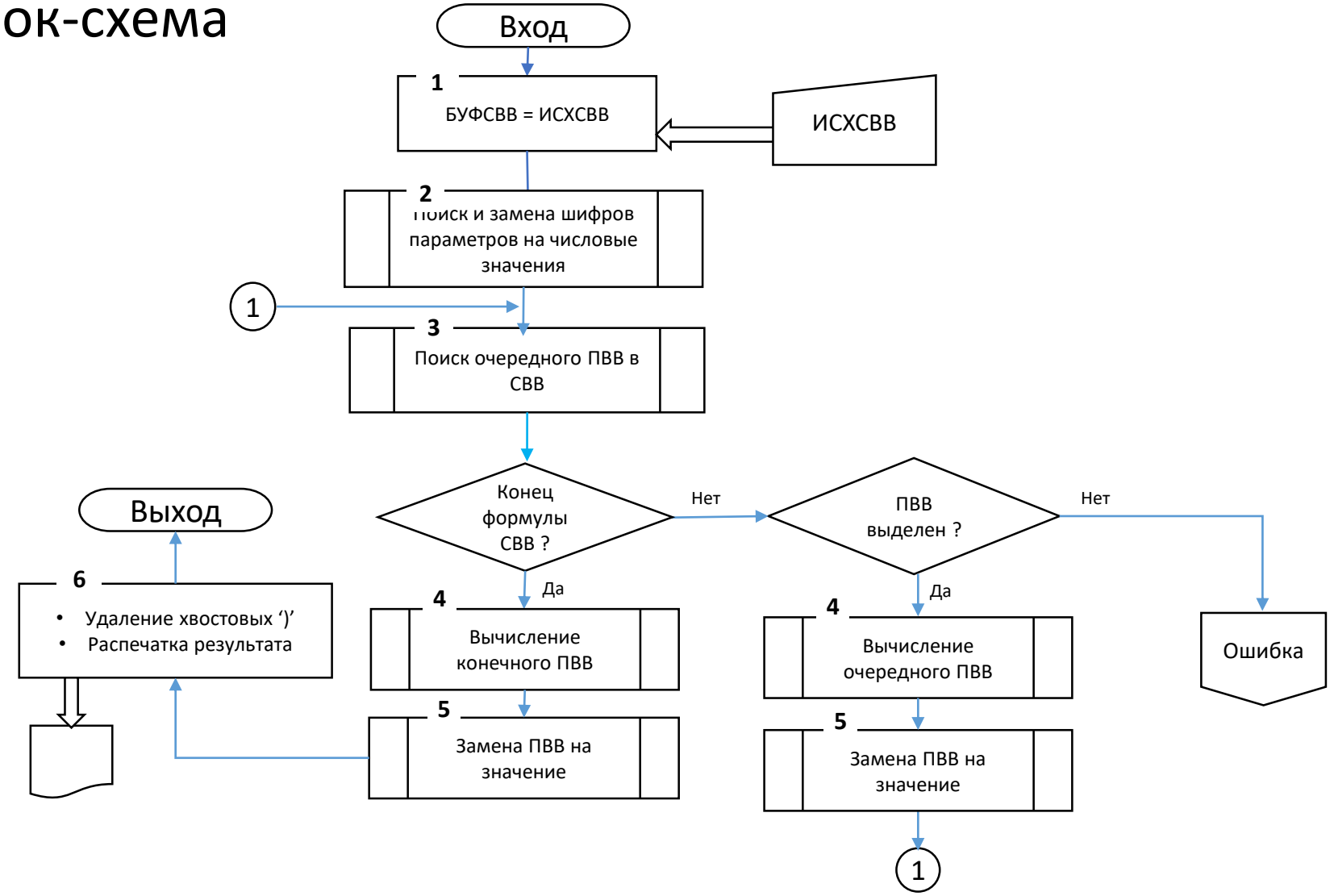
Чаще всего оформляют алгоритмы в виде блок-схем. Сложные алгоритмы обычно сопровождают текстовым описанием

2.4.1. Элементы блок-схем



2.4.2. Алгоритм модуля “Вычисление СВВ”

2.4.2.1. Блок-схема



2.4.2.2. Описание алгоритма

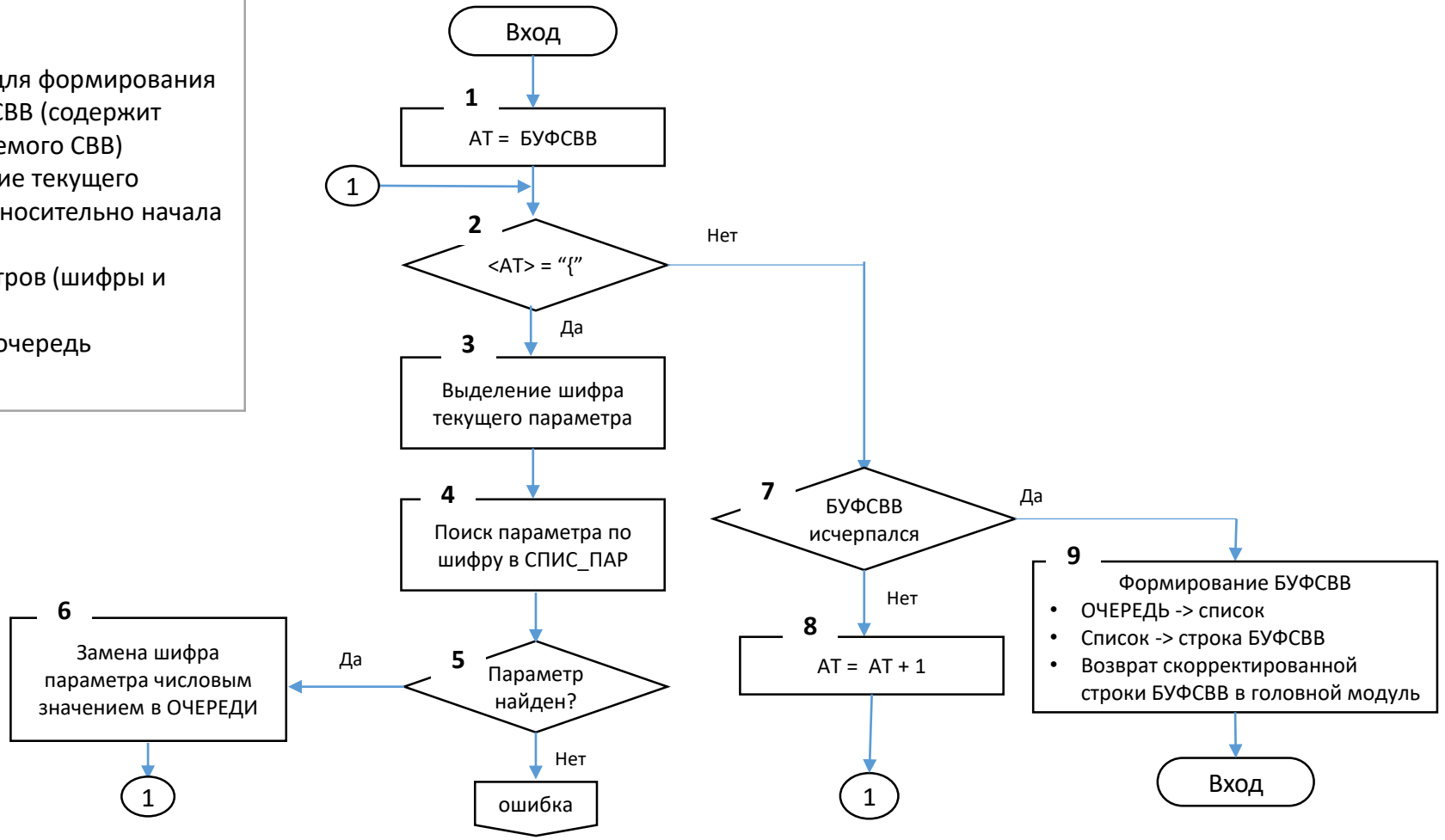
1. *Ввод с клавиатуры заданную СВВ и размещение его в строке ИСХСВВ. Копирование ИСХСВВ в буферную строку БУФСВВ для последующего формирования конечной ПВВ и вычисления конечного результата*
2. *Поиск и замена шифров параметров на числовые значения*
3. *Поиск и выделение очередного ПВВ из БУФСВВ*

   Проверка условия “ПВВ найден?”:
   - *Если ПВВ найден, то размещаем его в строке БУФПВВ для вычисления в процессе **4***
   - *Если очередной ПВВ не найден, то цикл завершается и переходим в процесс 6 для вычисления конечного результата*
4. *Вычисляем значение очередного выделенного ПВВ.*
5. *Замена текущего ПВВ вычисленным значением в БУФСВВ*
6. *Отсутствие новых ПВВ означает, что исходное СВВ стало простым ПВВ.*
   - *Удаление хвостовых круглых скобок*
   - *Вычисленное значение выводим на печать.*

2.4.3. Алгоритм модуля “Замена шифров параметров на значения”

Обозначения переменных

БУФСВВ – буферная строка для формирования окончательного результата СВВ (содержит текущее состояние вычисляемого СВВ)
АТ– адрес текущий (смещение текущего символа в строке БУФСВВ относительно начала строки)
СПИС_ПАР – список параметров (шифры и значение)
ОЧЕРЕДЬ – промежуточная очередь

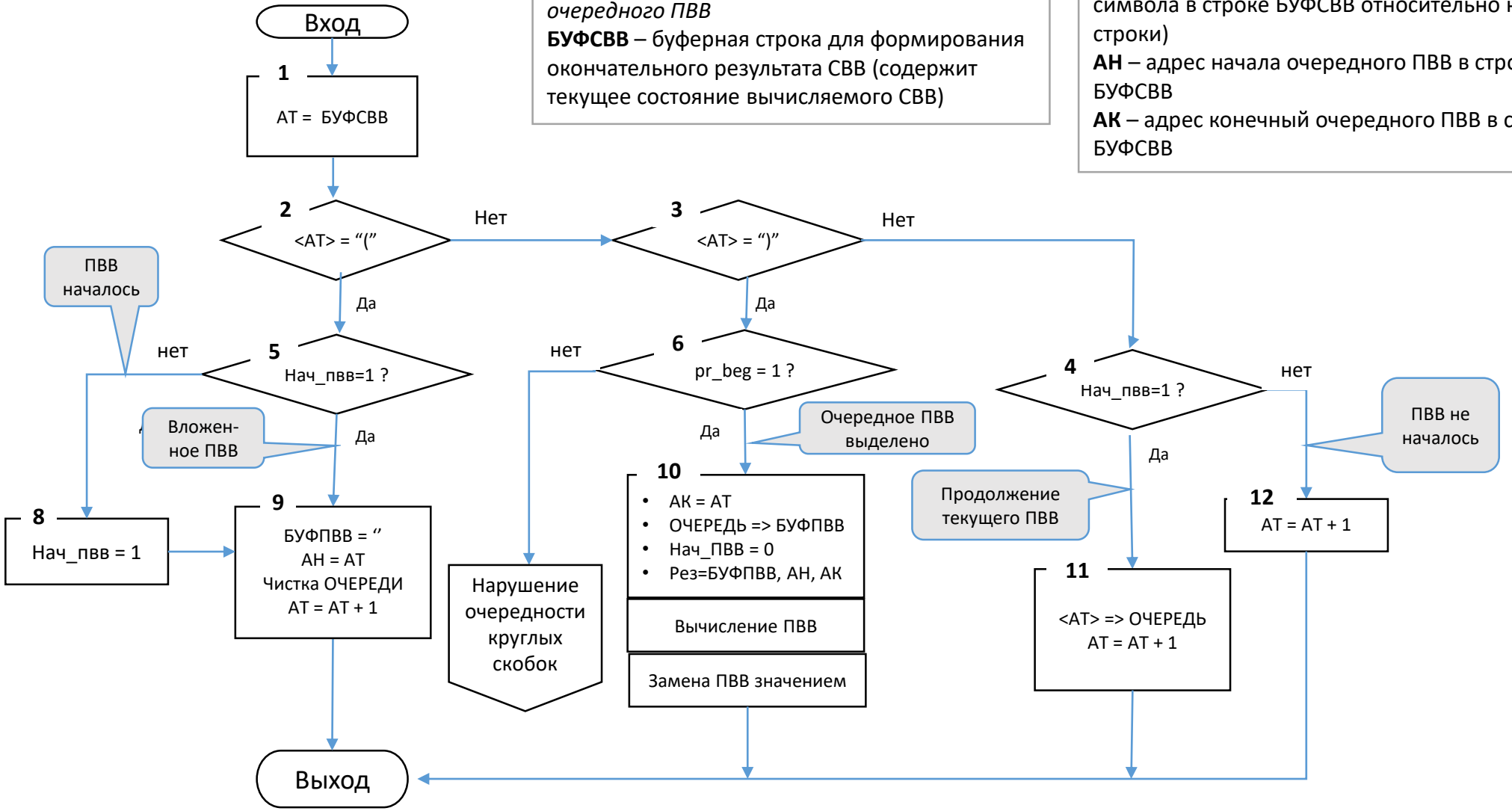


2.4.4. Алгоритм модуля “Поиск и вычисление ПВВ”

2.4.4.1. Блок-схема

Входные и выходные данные:
БУФПВВ *– буферная строка для хранения очередного ПВВ*
БУФСВВ – буферная строка для формирования окончательного результата СВВ (содержит текущее состояние вычисляемого СВВ)

Обозначения переменных
АТ– адрес текущий (смещение текущего символа в строке БУФСВВ относительно начала строки)
АН – адрес начала очередного ПВВ в строке БУФСВВ
АК – адрес конечный очередного ПВВ в строке БУФСВВ



2.4.4.2. Описание алгоритма “Поиск и вычисление ПВВ”

1. *Настройка на начало строк БУФСВВ и БУФПВВ*
2. *<u>Проверка содержимого текущего символа БУФСВВ</u>:*
   - *Если АТ != “(” , то продолжим поиск, перейдя на п.п. 3*
   - *Если АТ == “(” , то начинается очередное ПВВ. Переходим к п.п. 5*
3. *<u>Проверка содержимого текущего символа БУФСВВ</u>:*
   - *Если АТ != “)” , то продолжим поиск, перейдя на п.п. 4*
   - *Если АТ == “)” , то заканчивается очередное ПВВ. Переходим к п.п. 6*
4. *<u>Проверяем признак НАЧ_ПВВ:</u>*
   - *Если признак НАЧ_ПВВ != 1 , значит ПВВ не началось. Переходим к п.п.12*
   - *Если признак НАЧ_ПВВ == 1 , значит продолжается текущее ПВВ. Переходим к п.п.11*
5. *<u>Проверяем признак НАЧ_ПВВ:</u>*
   - *Если признак НАЧ_ПВВ != 1 , значит началось вложенное ПВВ. Текущее ПВВ завершаем и переходим к вложенному (п.п.9)*
   - *Если признак НАЧ_ПВВ == 1 , значит началось очередное ПВВ. Переходим к п.п.8*
6. *<u>Проверяем признак НАЧ_ПВВ:</u>*
   - *Если признак НАЧ_ПВВ != 1 , значит нарушена очередность круглых скобок. Т.е. ПВВ не началось, а “)” указывает, что ПВВ закончилось.*
   - *Если признак НАЧ_ПВВ == 1 , значит очередное ПВВ выделено. Переходим к п.п.10*

3. Написание программы в операторах языка программирования

3.1. Операторы языка PYTHON (ПИТОН), которые используются в нашей программе

Как мы убедились, что прежде чем писать программу в операторах языка программирования, нужно проделать огромную подготовительную работу по осмыслению поставленной задачи и продумывания алгоритмов реализации этой задачи.

3.1.1. Общие знаки

инструкция	пример	примечание
# примечание	# “пггпрщожлэс ча”	
= присвоить	Альфа= ‘abc’ beta=97	
== равно		
!= не равно		
< меньше		
> больше		

3.1.2. Стандартные функции

инструкция	пример	примечание
print (строка ! число)	print(‘rez=‘, rez)	Отладочная печать
Len (<строка>)	dlina = Len(<строка>)	Длина строки
str (<число>)	a = 24 a_sym = str(a) ‘24’	Преобразование INTEGER в символьное представление
int (<строка>)	a = ’24’ a_int = int(a) 24	Преобразование символьного представления числа в INTEGER
zfill (<символьное число, длина>)	a = ‘3’ a = zfill (a, 2) a = ‘03’	Добавление левых нулей в символьное представление числа
eval (<символьная формула (ПВВ)>)	znach = eval (‘3+4*2 -8/2’) znach = 7	Вычисление символьного представления ПВВ
pass		Пустое действие

3.1.3. Структуры данных

инструкция	пример	примечание
Строка ctk = ‘<последовательность символов>’	ctk1=‘Гена, ’ + ‘Саша, ’ + ‘Владик ‘ ctk1=‘Гена, Саша, Владик ‘ ctk1[5]=‘С’ ctk1[0:2]=‘Ге’ ctk1[2:4]=‘на’	1. *Склеивание* 2. *Индексирование* *(начинается с 0-го индекса)* • 5-й символ в строке • Начиная с 0-го символа первые 2-а символа • Из 4-х первых символов 2-а последних
Список a = [<список элементов через запятую>]	a = [374, ‘Саша’, 148, ‘внук’] a.append(‘Влад’) a = [374, ‘Саша’, 148, ‘внук’, ‘Влад’] a.remove(‘148’) a = [374, ‘Саша’, ‘внук’, ‘Влад’] a.Insert(2, ‘bbb’) a = [374, ‘Саша’, ’bbb’, ‘внук’, ‘Влад’]	• Список • Вставить элемент в конец списка • Удалить элемент • Вставить элемент перед элементом индекс, кот. Указан

Продолжение 3.1.3.

инструкция	пример	примечание
Словарь slovar = {‘<name1>’:<zn1>, ‘<name2>:<zn2>, …}	slovar{‘par1’:23, ‘par2’:2, ‘par3’:8} slovar[‘par2’] 23	• извлечение
Очередь ochered = [‘<элемент1>’, ‘<элемент2>’, …]	ochered = [‘aaaa’, ‘бббб’, ‘ввв’] ochered.append(‘ccc’) ochered = [‘aaaa’, ‘бббб’, ‘ввв’, ’ccc’] ochered.pop(0)	• Первым пришел, первым ушел • Добавить элемент • Удалить последний элемент
Стек ctek = [‘<элемент1>’, ‘<элемент2>’, …]	ctek = [‘aaaa’, ‘бббб’, ‘ввв’] ctek.append(‘ccc’) ctek = [‘aaaa’, ‘бббб’, ‘ввв’, ’ccc’] ctek.pop(0) ctek = [‘aaaa’, ‘бббб’, ‘ввв’]	• Первым пришел, последним ушел • Добавить элемент • Удалить последний элемент

3.1.4. Средства управления логикой

инструкция	пример	примечание

If <условие>:
elif: <действие 1>
elif: <действие 2>
…
else: <действие>
	If x < 0:
	x == 0
	elif x==0:
	print(‘ноль’)
	elif x==1:
	print ‘единица’
	else:
	print ‘больше’
		Условие выполнено
		
		- “-
		
		- “-
		
		Иначе

Цикл:
while <условие>:
<действие 1>
else:
<действие 2>
	a = 80
	b = 0
	while b < a :
	b=b+1
	else:
	print(‘b=‘, b)
		Пока (while) <условие> истинно, должно выполняться <действие 1> .
		Иначе (else) должно выполняться <действие 2>.
		Дополнительные инструкции:
		BREAK – прекратить
		CONTINUE – продолжить

Продолжение 3.1.4.

инструкция	пример	примечание
for n in range(n_beg, n _end): <действие>(n) …	**stroka('abdkgtpe')** **for n in range(0, len(stroka)):** **print(stroka[n])**	• Инструкция FOR перебирает элементы произвольной структуры (например, символьной строки) по индексу [n], указывающей на порядковый номер символа в строке (т.е. смещение относительно начала строки) • Диапазон значений индекса от 0 до последнего символа строки заданного встроенной функцией len(stroka), вычисляющей автоматически длину строки • В результате будет распечатана вся строка

3.1.5. Модули

инструкция	пример	примечание
<файл>.py	book.py	Программный модуль
--name--		Глобальная переменная, содержащая имя модуля и всех входящих
Import <имя модуля>	Import def_book	Объявление имени модуля, который содержит требуемые входяящие модули
def <имя модуля>	Import def_book def_razdel1	Вызываемый модуль может быть в головном модуле или в объявленном через IMPORT

3.2. Тексты программы

3.2.1. Модуль “Вычисление СВВ”

```
global ak
bufpvv=''
# 1:
isxctk='(40+({par1}-(8+({par2}*4+{par3}/2)))'
print('ИСХСВВ = ',isxctk)
bufcvv=isxctk
# 2:
an=0
ak=0
adn=0
rez=''
#==== Поиск и замена параметров =======
import def_primer_kniga
bufcvv=def_primer_kniga.poisk_par(bufcvv)
        print('bufcvv=',bufcvv)
```

```
# 3:
#======= Поиск и вычисление ПВВ ==============
rez=0
while rez != '#':
    import def_primer_kniga
    rez=def_primer_kniga.poisk_pvv(bufcvv,bufpvv)
    print('rez=',rez)
    #print(' ------ Формировае АН, АК ----------------')
    dln=len(rez)-6
    bufpvv=rez[0:dln]
    dln=len(rez)-3
    an=rez[dln-2:dln]
    an=an.zfill(2)
```

Продолжение 3.2.1.

```
dln=len(rez)
ak=rez[dln-2:dln]
#'---------------- 4. Вычисление ПВВ ------------'
import def_primer_kniga

rez=def_primer_kniga.vichisl(bufcvv,bufpvv,an,ak)
print('rez=',rez)


#---- 5. Замена ПВВ вычисленным значением ------
import def_primer_kniga
rez=def_primer_kniga.zamena_pvv(bufcvv,bufpvv,
    an,ak)
bufcvv = rez
print('БУФСВВ=',bufcvv)
```

```
#---------- 6. Удаление хвостовых ")" ------------
ddd=len(bufcvv)-1
if bufcvv[0] != '(':
        while bufcvv[ddd]==')':
                sym=bufcvv[ddd]
                ddd=ddd-1
        else:
                rez=bufcvv[0:ddd+1]
                print('rez=',rez)
                break
 else:
        pass
        rez=""
else:
   pass
```

3.2.2. Модуль “Замена шифров параметров на значения”

```
def  poisk_par( bufcvv ):
        print('=== 4. П о и с к   п а р а м е т р о в ===')
         print('БУФСВВ=',bufcvv)
        slovar={'par1':23, 'par2':2, 'par3':8}
        print('slovar=',slovar)
        # 1:
         at=bufcvv[1]
        dlina=len(bufcvv)
        # 2:
         text=[]
        och=[]
        for n in range(0,dlina):
                # 3:
        if bufcvv[n]=="{":
                # 4. Выделение шифра параметра:
                par=(bufcvv[n+1]+bufcvv[n+2]+bufcvv[n+3]+bufcvv[n+4])
                #print('par=',par)
                # 5, 6. Поиск и замена шифра на значение (запись в
                очередь):
                och.append(slovar[par])
                #print('och_{=',och)
elif bufcvv[n]=="}":
                # Удаление остатков шифра парамаетра из очереди:
                och.pop()
                och.pop()
                och.pop()
                och.pop()
  else:  # 2 - <AT> != "{"}, 7 - БУФСВВ не исчерпался,
                 # 8 – AT=AT+1:
                  och.append(bufcvv[n])
                  pass
```

Продолжение 3.2.2.

```
                    # 7 - БУФСВВ исчерпался (да):
print('och = ',och)
dl=len(och)
#print(dl)
buf=[]
buf1=[]
 # 9 :
 # Перепись содержимого очереди в список
 buf1
 for m in range(0,dl):
            buf=och.pop(0)
             buf1.append(buf)
```

```
#  Преобразование списка buf1 в строку rez:
for k in range(0,dl):
            rez=str(rez)+str(buf1[k])


# Возврат строки REZ в главную программу:
bufcvv=rez
return bufcvv bgtv
```

3.2.3. Модуль “Поиск и вычисление ПВВ”

```
adn=0
buf_adn=0
def poisk_pvv(bufcvv,bufpvv):
        global buf_adn
        global och_
        global adn
        och_=[]
        rezul1=''
        rezul=''
        bufpvv=''
        # 1:        Начало
        at = bufcvv[0]
        adn = bufcvv[0]
        pr_beg=0
        sym=''
        m=0
for n in range(0,len(bufcvv)):
                # 2:        <AT> = ‘)’ ?
                If  bufcvv[n]=="(“
                        # 5:
                        If pr_beg==0:
                        # 9 - ПВВ началось
                        pr_beg=1
                        an=def_begpvv(bufcvv,n,
                        pr_beg,m,bufpvv,och_)
                        rezul1=''
                        an_s=str(an)
                        rezult=an_s.zfill(2)
                else:
```

Продолжение 3.2.3.

```
            # 8 - Вложеная ПВВ
            an=def_begpvv(bufcvv,n,pr_beg,m,bufpvv,och_)
            rezul1=''
            an_s=str(an)
            rezult=an_s.zfill(2)
# 3 - <AT> = “)”
elif bufcvv[n]==")":
            # 6:
            if pr_beg==1:
                        # 10 - ПВВ выделено
                        rezult=def_endpvv(bufcvv, pr_beg,n,m,
                        bufpvv,och_,an)
            else:
                        # Нарушение очередности круглых скобок
                        pass
 else:
            # 4==1:
            If pr_beg
                        # 11 - Продолжение ПВВ
                        def_prod(bufcvv,pr_beg,n,m,bufpvv,och_)
                        return result
            else:
                        #12: - ПВВ не началось
                        pass
#================ '(' ==================================
 def def_begpvv(bufcvv_,n_,prbeg,m_,bufpvv,och_):
            adn=n_
            buf_adn=adn
            buf=''
            for k in range(0,len(och_)):
                        buf=och_.pop(0)
            if prbeg==1:
                        # вложенная ПВВ
                        bufpvv=‘’
```

Продолжение 3.2.3.

```
                m_=0
        else:    # начало ПВВ
                prbeg=1
                bufpvv=''

                ad=str(adn)
                ad=ad.zfill(2)
                return ad

#================ ')'============================
def def_endpvv(bufcvv_,prbeg,n_,m_,bufpvv,och_,an):
        #print('------------ def_endpvv ---------------------------')
        m_=0
        dln=len(och_)
        buf=[]
        buf1=[]
        rez=''
if prbeg==1:
        prbeg=0
        ak=n_
        # Перепись содержимого очереди в буфер
        for c in range(0,dln):
                buf=och_.pop(0)
                buf1.append(buf)
        # Преобразование списка в строку
        for k in range(0,dln):
                rez=str(rez)+str(buf1[k])
                bufpvv=rez
        prbeg=0
else:
        prbeg=0
result=‘’
```

Продолжение 3.2.3.

```
        rezult=''
        rezult=(str(bufpvv)+','+str(an)+','+str(ak).zfill(2))
        #print('rezult=',rezult)
        return rezult

#============= Продолжение ======================
 def def_prod(bufcvv_,prbeg,n_,m_,bufpvv,och_):
        #print('------------- def_prod ----------------------------')
        if prbeg==1:
                # Запись символа в очередь
                och_.append(bufcvv_[n_])
                #print('och_=',och_)
                 m_=m_+1
        else:
                prbeg=1
        return
```

```
#============ 7. Вычисление ПВВ ===============
def vichisl(bufcvv,bufpvv,an,ak):
print(bufpvv, an, ak)
        aaa=eval(bufpvv)
        #print('aaa=',aaa)
        rez=int(aaa)
        return rez

#============ 8. Замена ПВВ значением =============
 def zamena_pvv(bufcvv,bufpvv,an,ak):
        an_i=int(an)
        ak_i=int(ak)
        # Перепись БУФСВВ в очередь д АТ=АН
        for n in range(0,an_i):
                och_.append(bufcvv[n])
```

Продолжение 3.2.3.

 # Запись вычисленного значения в очередь,

начиная с АН:

```
aaa = eval (bufpvv)   # формирование значения ПВВ из
                        символьного  представления
                        формулы
znach=int(aaa)
#print('znach=',znach)
och_.append(znach)
```

 # Запись в очередь хвоста БУФСВВ

```
for m in range(ak_i,len(bufcvv)):
    och_.append(bufcvv[m])
```

 # Перепись очереди в БУФСВВ

```
buf1=[]
ddd=len(och_)
for v in range(0,ddd):
        buf=och_.pop(0)
        buf1.append(buf)
        #print(bufcvv)
        #print('bufcvv=',buf1)
```

 # Преобразование списка в строку

```
rez=''
for k in range(0,ddd):
        rez=str(rez)+str(buf1[k])
bufcvv=rez
return bufcvv
```

3.2.4. Отладочная печать

ИСХСВВ = (40+({par1}-(8+({par2}*4+{par3}/2)))

====== П о и с к п а р а м е т р о в =====

БУФСВВ= (40+({par1}-(8+({par2}*4+{par3}/2)))

slovar= {'par1': 23, 'par2': 2, 'par3': 8}

bufcvv= (40+(**23**-(8+(**2***4+**8**/2)))

====== П о и с к П В В =====

rez= 2*4+8/2,11,19

---------- Вычисление ПВВ-1 ----------

2*4+8/2 11 19

rez= **12**

---------- Замена ПВВ-1 значением ----------

БУФСВВ= (40+(23-(8+**12**)))

====== П о и с к П В В-2 =====

rez= 8+12,08,13

---------- Вычисление ПВВ-2----------

8+12 08 13

rez= **20**

---------- Замена ПВВ-2 значением ----------

БУФСВВ= (40+(23-**20**)))

====== П о и с к П В В-3=====

rez= 23-20,04,10

---------- Вычисление ПВВ-3 ----------

23-20 04 10

rez= **3**

---------- Замена ПВВ-3 значением ----------

БУФСВВ= (40+**3**)))

Продолжение 3.2.4.

====== П о и с к П В В-4 =====

rez= 40+3,00,05

---------- Вычисление ПВВ-4 ----------

40+3 00 05

---------- Замена ПВВ-4 = СВВ значением ----------

БУФСВВ = **43**)))

---------- Удаление хвостовых круглых скобок ------

СВВ = **43**

4. Объектно-ориентированное программирование

Многие задачи требуют наличия сценария диалога между программой и человеком. Этот процесс создания диалога решили систематизировать, создав **классы** отдельных визуальных объектов, таких как: *линия, кнопка управления, окно для текста, таблица БД, окно для создания древовидной структуры* и т.д.

В некоторых платформах в классы оформлены не только визуальные операторы, но и большинство других программных объектов, таких как средства работы в сети, средства доступа к интернету и т.п.. Лидером такого подхода к программированию стала кросс-платформа QT для языка C++.

Каждый класс программных объектов содержит набор свойств и параметров, характеризующих размер, координаты, цвет и другие свойства. Т.е. все многообразие экземпляров одного класса отличаются, например, координатами размещения их на экране, размером и цветом.

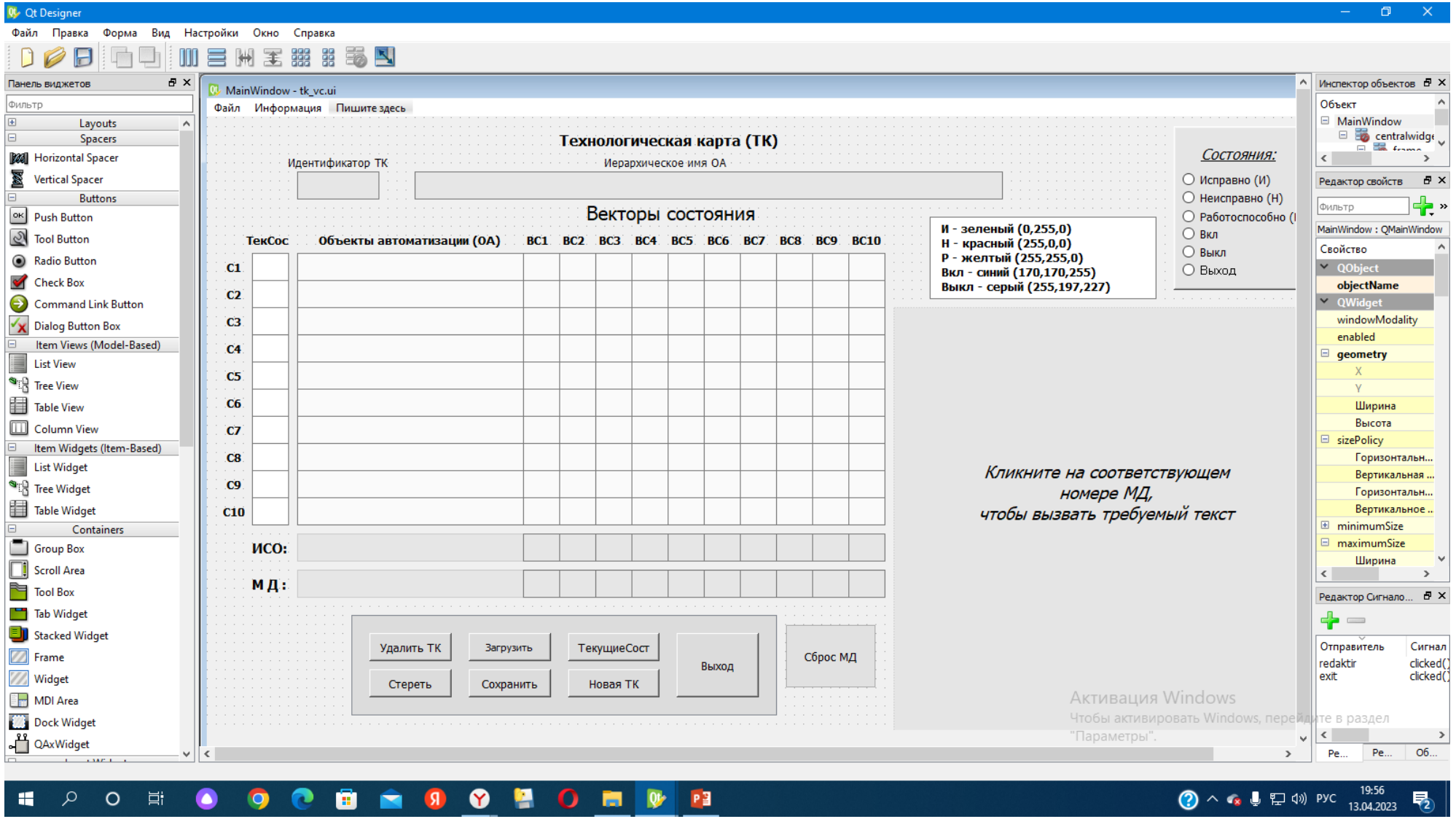
Для конструирования сценариев диалога разрабатываются специальные средства проектирования визуальных форм. Наиболее популярным средством создания визуальных форм является **QT-Designer**, используемый в Питоне.

Размещение визуальных объектов осуществляется перемещением по экрану методом захвата объектов манипулятором мышь. Также управляется процесс задания размеров объекта. Шрифт текста и цвет задаются выбором из меню.

Заданные параметры, используемых в форме экземпляров класса, автоматически регистрируются в тексте программного модуля, отображающего визуальную форму. Каждая визуальная форма оформляется в виде отдельного программного модуля с расширением “.un”.

Визуальная форма с расширением “.un” должна быть перекомпилирована в формат с расширением “.py”. Сложность заключается в том, что в одном модуле формируются автоматически тексты описания экземпляров классов и операторы языка, сформированные программистом. Т.е. программист должен сохранять свои тексты и восстанавливать их при любой модификации визуальной формы.

Пример визуальной формы



5. Системы Управления Базами Данных

Появление Систем Управления Базами Данных (СУБД) было революцией в программировании. Данные были отделены от программы. Их можно разместить на файл-сервере и сделать доступными одновременно с разных терминалов (вынесенных рабочих мест). Появилась возможность работы с большими массивами данных.

Были разработаны Язык Описания Данных (ЯОД) и Язык Манипулирования Данными (ЯМД), которые систематизировали и обеспечили единые правила описания и доступа к данным. Рассмотрим простейшие запросы на этих языках в СУБД SQLITE, используемого в языке программирования PYTHON.

5.1. Язык описания данных (ЯОД)

5.1.1. Создание таблицы

CREATE TABLE <таблица> (<ключ> INTEGER PRIMARY KEY <колонка1> CHAR(255), <колонка2> CHAR(255) …)

<ключ> - первичный индекс записей (не очень хорошее решение, т.к. он не должен повторяться в разных таблицах)

Команда **.tables** отображает список таблиц

Продолжение 5.

5.1.2. Удалить таблицу (DROP TABLE)

Рассмотри последовательность команд:

CREATE TABLES Demo (<индекс> INTEGER PRIMERY KEY, <kolonka1>, <kolonka2>)

CREATE TABLES book (<индекс> INTEGER PRIMERY KEY, <kolonka1>, <kolonka2>)

.tables

Demo, book

DROP TABLE demo

.tables

Book

5.1.3. Изменение названия таблицы

ALTER TABLE <имя_таб> RENAME TO <нов_имя>

5.1.4. Добавить новые столбцы

ALTER TABLE <имя_таб> ADD COLUMN <нов_кол> BOOLEAN

UPDATE <имя_таб>

SET <нов_кол> = True

Продолжение 5.

5.2. Язык манипулирования данными

5.2.1. Фильтрация данных

SELECT (<колонка1>, <колонка2>Б …) FROM <таблица> WHERE <колонкаN> = 1

Ключевые слова:

WHERE - задает условие фильтрации

GROUP BY – группирует записи по значениям, указанных колонок

AND, OR и BETWEEN – ***и, или, в диапазоне*** (расширяет выборку, заданную WHERE

LIMIT – ограничивает кличество значений выборки (например, по заданному условию выборки получено 100 значений, а нужны первые10. Тогда задается LIMIT 10.

5.2.2. Вставить записи

INSERT INTO <таблица> (<индекс>, <колонка1>, <колонка3>) VALUES(?, ?, ?)

data = (1, ‘Митя’, 40),

(2, ‘Влад’, 10),

(3, ‘Саша’, 6)

Продолжение 5.

5.2.3. Изменить запись

UPDATE <таблица>

SET <колонка> = <значение>

WHERE <индекс> = 1 OR <индекс> = 3

5.2.4. Удалить записи

- *Удаление всех записей:*

  DELETE FROM <таблица>

- *Удаление записей согласно условию:*

  DELETE FROM <таблица> WHERE <условие>