**2 Спецификация требования языка ImageScript**

Язык ImageScript – это интерпретируемый язык программирования. Как и все остальные другие языки программирования, язык ImageScript содержит стандартное множество встроенных типов данных, таких как числа, массивы и строки, а также основные элементы управления, такие как условия, циклы, блоки команд.

В языке ImageScript – предусматриваются следующие графические средства :

– предварительная обработка, сегментация и распознавание изображений;

– встроенная поддержка линейных преобразований изображений;

– загрузка и сохранения изображение в распространенных форматах.

Язык ImageScript – это стековый язык программирования. Данные представлены в виде объектов. Простые объекты хранятся в стеке, а сложные – в виде структур в виртуальной памяти, указатели на которые хранятся в стеке. Интерпретатор ImageScript управляет следующими стеками [9,1]:

* стек операндов, который содержит объекты, являющиеся операндами операторов или результатами выполнения операндов;
* стек выполнения, который содержит указатели на исполняемые объекты, находящиеся в промежуточной стадии выполнения;
* стек контекста (графический стек), который хранит объекты текущего графического состояния.

**2.1 Модель данных**

Все данные языка ImageScript, существуют в форме объектов. У каждого объекта есть тип и значение.

Объекты бывают простые и сложные. Простой объект является константой, его тип значения, значение и атрибуты неизменно соединены вместе и не могут быть изменены. У сложных объектов значения хранятся отдельно от самых объектов [9].

**2.1.1 Типы данных и объекты**

Объекты в ImageScript бывают простые (не имеющие внутренней структуры) и сложные (или составные). Ниже приведены виды простых и сложных объектов в ImageScript:

Таблица 2 – Типы данных и объекты

|  |  |
| --- | --- |
| Простые объекты | Сложные объекты |
| boolean  integer  null  operator  real | array  string |

Числа в языке ImageScript бывают:

– знаковые целые (integer), такие как 123,-76;

– вещественные (real), такие как .002, 4.7

Для логических выражений поддерживаются логические объекты со значениями true и false.

Объекты операторы (operator) представляют некоторые встроенные в язык действия.

Любой токен, состоящий из стандартных символов, который нельзя рассматривать как число (например, "abc", "1A") является именем или именным объектом (name).

Массивом (array) являются последовательность объектов ImageScript, заключенная между "[" и "]", например "[ 123/abc (xyz) ]".

Похожий синтаксис есть у процедур (procedure), которая является исполняемым массивом. Процедура- это последовательность объектов, заключенная между "{" и "}".

**2.2 Структура памяти интерпретатора**

Данные в языке ImageScript представлены в виде объектов. Важное место в языке занимают стеки и виртуальная память, поскольку все объекты хранятся на стеках, а виртуальная память предоставить место для хранения значений сложных объектов [9].

**2.2.1 Стеки**

Интерпретатор ImageScript управляет стеками, которые представляют исполняемое состояние программы:

– стек операндов содержит произвольные объекты ImageScript, являющиеся операндами или результатами некоторых исполняемых операторов;

– cтек исполнения содержит исполняемые объекты, находящиеся в промежуточной стадии исполнения;

– графический стек хранит только объекты графического состояния [10].

**2.2.2 Управления памятью**

В языке ImageScript предусмотрено хранилище для значений сложных объектов- виртуальная память. Сложный объект имеет ссылку на свое значение. Виртуальная память подразделяется на локальную и глобальную. Основным отличием является то, что операторы сохранения save и восстановления restore состояния памяти влияют только на виртуальную память. Для освобождения памяти, выделенной для объектов, которые уже не используются, используется сборщик мусора.

**2.3 Арифметические операции**

Оператор в ImageScript – это слово, заставляющее интерпретатор выполнять те или иные действия. Он эквивалентен командам или процедурам в других языках программирования. Когда интерпретатор встречает слово в ImageScript – программе, он просматривает свой внутренний словарь и пытается определить, является ли это слово именем оператора. Если оно в словаре найдено, то выполняются все связанные с ним действия, а затем переходит к следующему слову в исходном файле. Итак, перечислим операторы вычислений :

– sub – вынимает два числа из стека, вычитает их и возвращает их разность;

– add – вынимает два числа из стека, складывает их и возвращает их сумму;

– mul – вынимает два числа из стека, умножает их и возвращает их произведение;

– div – вынимает два числа из стека, делит их и возвращает их частное;

– idiv – вынимает два целых числа из стека, делит их и возвращает целое частное.

– mod – вынимает два числа из стека, делит их и возвращает остаток от деления.

Также есть стандартный набор функций, преобразующих одно число в другое. Они просто изменяют последнее число в стеке:

– abs – абсолютное значение;

– neg – изменение знака числа;

– ceiling – ближайшее целое сверху;

– floor – ближайшее целое снизу;

– round – ближайшее целое с меньшим расстоянием;

– truncate– дробная часть числа;

– sqrt – квадратный корень;

– atan – берет два числа из стека и возвращает арктангенс их отношения в градусах;

– cos – косинус числа в градусах;

– sin –синус числа в градусах;

– exp – берет два числа и возвышает предпоследнее в степень последнего;

– log – десятичный логарифм;

– ln – натуральный логарифм.

Таблица 3 – Пример выполнение команды

|  |  |
| --- | --- |
| Операции | Результаты |
| 5 27 add | 32 |
| 10 2 sub | 8 |
| 16 sqrt | 4 |
| 8 2.2 mul | 17,6 |

В таблице 3 показано примеры арифметических операции языка ImageScript.

**2.4 Действие над стеком**

Стеком называется хранилище данных, в котором можно работать только с одним элементом, тем который был добавлен в стек последним. Стек должен поддерживать следующий операции:

– push – добавить в конец стека новый элемент;

– pop – извлечь из стека последний элемент;

– back – узнать значение последнего элемента ( не удаляя его);

– size – узнать количество элементов в стеке;

– clear – очистить стек;

– dup – дублирует верхний элемент в стеке.

Таблица 4 – Операции над стеком

|  |  |
| --- | --- |
| Операции | Результаты |
| 6 8 12 clear | – |
| 7 dup | 7 7 |
| 16 9 pop | 16 |

В таблице 4 показано действия над стеком языка ImageScript.

**2.5 Процедуры и переменные**

Словарь (dictionary) - это таблица, которая связывает между собой пары объектов. ImageScript – словарь связывает объект, именуемый *ключом*, с другим объектом – *значением* этого ключа. Интерпретатор языка ImageScript может искать по ключу в словаре и получать его значение, если такой ключ есть в таблице. ImageScript всегда имеет два словаря: системный и пользовательский. *Системный словарь* объединяет имя каждого встроенного в язык оператора с соответствующим ему действием. *Словарь пользователя* ассоциирует имена с процедурами и переменными, определенными в программе.

Когда интерпретатор встречает имя он сначала просматривает словарь пользователя, а затем системный. Если имя в словаре найдено, то выполняются соответствующие ему действия: либо объект помещается в стек, либо выполняются некоторый набор операторов. Если имя не найдено в словаре (например на стадии отладки программы), то выдается сообщение об ошибке.

Чтобы определить *переменную* в ImageScript ее имя и значение нужно занести в текущий словарь. Это делается с помощью оператора def, как в следующем примере:

/ed 5 def (2.5.1)

Косая черта перед именем переменной показывает, что интерпретатору следует поместить это имя в стек как литерал и не пытаться сразу же искать его в словаре. Вслед за именем в стек заносится число 5. И наконец def берет оба эти объекта из стека и помещает их в текущий словарь. Второй элемент стека (ed) становится ключом, с которым ассоциировано значение первого элемента (5).

Процедура в ImageScript – это набор операторов сгруппированных под общим именем. Имя процедуры является ключом в словаре, а набор операторов ассоциируется как его значение. Когда имя процедуры появляется в программе, то выполняется связанный с ним набор операторов. Процедуры в ImageScript определяются точно так же, как и переменные, с тем только отличием, что набор операторов процедуры должен быть заключен в фигурные скобки. Следующий пример описывает процедуру fact , который вычисляет факториал.

Таблица 5 – Пример вычисление факториала на языке ImageScript

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | /fact{ | |  | begin | |  | /n exch def | |  | n 2 lt | |  | { 1} | |  | {n 1 sub fact n mul } | |  | ifelse | |  | end | |  | } def | |

Использование процедур сокращает длину программы, улучшает ее читабельность, а главное дает возможность расширить средства языка за счет введения в него новых слов. Таким образом обычный процедурный язык программирования можно поднять до сверхвысокого уровня.

**2.6 Условные переходы**

**2.6.1 Оператор if**

В ImageScript операторы сравнения следуют за сравниваемыми величинами:

*Оператор Значение.*

Таблица 6 – Операторы сравнения

|  |  |
| --- | --- |
| Eg | = (равно) |
| Gt | > (больше) |
| Ge | >= (больше или равно) |
| Ne | <> (не равно) |
| It | < (меньше) |
| Le | <= (меньше или равно) |

Синтаксис оператора if выглядит следующим образом : .

Результат операции сравнения или логической операции – логическое значение true или false. Оператор if берет из стека логический объект и выполняемый массив. Если значение логического объекта true, то выполняются операции, записанные в этом массиве.

**2.6.2 Оператор ifelse**

Позволяет выбрать в зависимости от условия выполнение одной или другой последовательности операторов. Его формат выглядит следующим образом:

(2.6.2.1)

Если результат выполнения условия true, выполняется последовательность {proc1}, иначе {proc2}.

**2.6.3 Логические операторы**

Логические операторы в ImageScript работают как с целыми числами, так и с логическими. Имеется четыре основных логических операции: *and, or, not, xor* (из которых можно построить все остальные булевские функции.

Таблица 7– Логические операторы

|  |  |
| --- | --- |
| Оператор | Описание |
| & | Логическое AND (И) |
| | | Логическое OR (ИЛИ) |
| ^ | Логическое XOR (исключающее ИЛИ) |
| ! | Логическое унарное NOT (НЕ) |

Логический оператор ! инвертирует (меняет на противоположный) булево состояние : *!true == false и !false == true.*

**2.6.4 Циклы**

В языке ImageScript имеется три основных циклов : простой цикл, индексируемый цикл и условный цикл.

*Простой цикл* реализуется с помощью оператора repeat [9], который берет из стека два операнда: счетчик цикла и повторяемую процедуру. Например, для очистки стека можем использовать следующий цикл:

5 {pop} repeat (2.6.4.1)

Здесь все достаточно прозрачно, непривычно, может быть, только то, что в качестве одного из аргумента выступает повторяемая процедура.

*Индексируемый цикл* реализуется с помощью оператора for. Оператор for берет из стека четыре операнда : начальное значение счетчика цикла, его приращение, конечное значение счетчика цикла и повторяемую процедуру[2]. Следует учитывать, что непосредственно перед выполнением этой процедуры for помещает в стек текущее значение счетчика и, если он не используется, то его следует оттуда явным образом удалить.

*Условный цикл* строится из двух операторов : loop и exit. Оператор loop повторяет выполнение процедуры до тех пор пока в ней не встретится оператор exit, который заканчивает циклическое выполнение. Если в повторяемой процедуре нет оператора exit, то цикл будет бесконечным.

**2.7 Массивы**

ImageScript работает с одномерными массивами (векторами), которые определяются как набор объектов (возможно разного типа), заключенный в квадратные скобки. Например,

(2.7.1)

Первый элемент этого массива является строка. Такие структуры в других языках , кроме АПЛ, обычно называют записями.

Массив может быть также определен с помощью оператора *array*, который берет из стека число и создает массив такой длины. Например,

(2.7.2)

Эта строка оставит в стеке массив из 10 элементов. Его элементы – NULL объекты.

Для работы с элементами массивов служат операторы *put* и *get*.

Оператор put берет из стека три аргумента : массив, индекс элемента в массиве и объект. Он помещает в объект в массив в позицию, заданную индексом.

У оператора get два аргумента: массив и индекс. Он возвращает в стеке элемент массива с заданным индексом. Например, после выполнения строки

[0, 1, 2, 3, 4, 5] 3 get

в вершине стека будет число 2.

Для работы с массивом также необходимо оператор length, возвращающий длину массива.

Еще два оператора, *load* и *store* облегчают загрузку и сохранение сразу всего массива. Так оператор *load* берет в качестве аргумента массив, заносит в стек по очереди все его элементы, а затем заносит туда сам массив. Оператор *store* выполняет обратную функцию.

**2.8 Встроенные графические команды для анализа и обработки изображений**

Язык ImageScript содержит также встроенные графические команды для стандартного анализа и обработки изображений.

**2.8.1 Двумерное прямое дискретное косинусное преобразование**

Функция V=iCos2(S) осуществляет двумерное прямое дискретное косинусное преобразование матрицы S и помещает результат в матрицу V. Матрицы S и V имеют одинаковый размер.

Двумерное прямое дискретное косинусное преобразование осуществляется формулой:

где

**2.8.2 Обратное двумерное дискретное косинусное преобразование**

Функция V=iDicos2(S) осуществляет обратное двумерное дискретное косинусное преобразование матрицы S и помещает результат в матрицу V. Матрицы S и V имеют одинаковый размер [6].

Обратное двумерное дискретное косинусное преобразование осуществляется формулой:

где

**2.8.3 Вывод изображения на экран**

Функция iShow(I,n) выводит на экран полутоновое изображения I, используя при выводе n уровней серого.

**2.8.4 Масштабирование изображение**

Функция iScale реализует масштабирование изображений путем линейной интерполяции пикселей с помощью аппаратных расширений. Синтаксис данной функции выглядит следующим образом:

iScale(P,V,row,col), где P – исходное изображение,V– масштабируемое изображения , rol – горизонтальный размер входного изображения, col – вертикальный размер входного изображения.

**2.8.5 Построение гистограммы**

Функция IMG\_histogram реализует гистограммный анализ. Входные данные изображения должно быть в диапазоне [0,255]. Формат входных и выходных данных изображения должно быть приведено в Q16.0. Синтаксис данной функции выглядит следующим образом :

ihistogram(P,V, size),

где P – исходное изображение,V– выходные данные гистограммы, size – размер изображения.

**2.8.6 Свертка изображений**

Функция U=iCon2(A,B) вычисляет свертку матриц А и B. Размер матрицы U определяется следующим образом [1] : если

[ma,na]=size(A),

[mb,nb]=size(B) и [mu,nu]=size(U), то

mu=max([ma+mb-1,ma,mb]) и

nu=max([na+nb-1,na,nb])

**2.8.7 Ранговая фильтрация**

Функция В= iOrder2(A,order,domain) создает полутоновое изображение B, каждый пиксель которого формируется следующим образом. Пиксели исходного полутонового изображения A, соответствующие ненулевым элементам маски фильтра domain, сортируются по возрастанию. Пикселю изображения B, соответствующему центральному элементу маски, присваивается значение с номером order в отсортированном множестве. Операция применяется нерекурсивно для всех положений маски. Фильтрацию такого вида называют порядковой или ранговой [1].

**2.8.8 Медианная фильтрация**

Функция B=medfil2D(A,[m,n]) создает полутоновое изображение B, каждый пиксель которого формируется следующим образом. Пиксели исходного полутонового изображения A, соответствующие всем элементам маски фильтра размера mx.gif (833 bytes)n, составляют упорядоченную последовательность Q. Пикселю B(r, с), где r и с – координаты текущего положения центрального элемента маски, присваивается значение медианы последовательности Q. Операция применяется нерекурсивно для всех положений маски.

Медианой упорядоченной последовательности Q(i), где i=1...N, называется величина Q((N + 1)/2), если N - нечетное, и (Q(N/2) + A((N+2)/2))/2, если N - четное.

**2.8.9 Сегментация методом разделения**

Функция **S=** iQtdecomp**(I)** осуществляет сегментацию полутонового изображения I методом разделения и помещает результат в разреженный массив S. Разреженный массив S конструируется следующим образом. Элементам матрицы S(r, с), соответствующим координатам левых верхних углов блоков на исходном изображении I, присваиваются значения, определяющие размеры каждого блока. Таким образом, большинство элементов матрицы равно нулю. И поэтому для хранения квадро-дерева применяется разреженный массив, который эффективно использует память, когда большинство элементов массива равно нулю. Для данной функции критерием однородности блока является равенство всех пикселей блока друг другу[2].

**2.8.10 Выделение границ**

Функция **B=iEdge(I,method)** предназначена для выделения границ на исходном полутоновом изображении I. Данная функция возвращает бинарное изображение B такого же размера, как исходное I. Пиксель B(r, с) равен 1, если пиксель I(r, с) принадлежит границе. Для обнаружения границ может использоваться несколько методов. Применяемый метод задается в параметре method в виде одной из следующих строк: 'sobel', 'prewitt', 'roberts', 'log', 'zerocross', 'canny'. Если параметр method при вызове функции опущен, то по умолчанию он полагается равным 'sobel'.

**2.8.11 Двумерное преобразование Фурье**

Функция A= iFFT2 (B,m,n) вычисляет двумерное преобразование Фурье матрицы mxn, при необходимости дополняя нулями или усекая исходную матрицу B. Возвращается матрица A размера mxn.

**2.8.12 Поворот изображения**

Функция **D=iRotate(S, angle, method)** создает изображение D, соответствующее повернутому исходному изображению S, используя один из предопределенных методов интерполяции [3], который задается во входном параметре method в виде одной из следующих строк:

''nearest" - использовать значение ближайшего пиксела (установлено по умолчанию, и данный параметр может быть опущен при вызове функции);

''bilinear" - использовать интерполяцию по билинейной поверхности;

''bicubic" - использовать интерполяцию по бикубической поверхности;

Угол поворота angle задается в градусах. Положительные значения данного параметра соответствуют повороту против часовой стрелки, а отрицательные - по часовой стрелке.

Тип и формат представления данных результирующего изображения совпадают с типом и форматом исходного.

**2.8.13 Конвертирование из RGB в НСV**

Функция **HSV=**iRGB**(RGB)** создает полноцветное изображение, значения пикселов которого представлены в цветовой системе HSV, из исходного полноцветного изображения в цветовой системе RGB. Результирующее изображение имеет формат представления данных double.

Широкое распространение в компьютерной графике и цифровой обработке изображений получила цветовая система HSV (hue - цветовой фон, saturation - насыщенность, volume - светлота) [3]. Часто для названия этой же системы используется аббревиатура HSB (hue, saturation, brightness - яркость).

**2.8.14 Конвертирование из HSV в RGB**

Функция **RGB=** iHCV **(HSV)** создает полноцветное изображение, значения пикселов которого представлены в цветовой системе RGB, из исходного полноцветного изображения в цветовой системе HSV. Исходное изображение должно иметь формат представления данных double. Результирующее изображение также имеет формат представления данных double.

**2.8.15 Конвертирование из YCbCr в RGB**

Функция **RGB=**iYCbCr**(YCbCr)** создает полноцветное изображение, значения пикселов которого представлены в цветовой системе RGB, из исходного полноцветного изображения в цветовой системе YCbCr. Формат представления данных исходного и результирующего изображений совпадают.