– Разбиение изображения на квадратные блоки («разлив»), искажение-искажение изображения в случае потери цифрового сигнала или повреждения носителя (царапины на DVD, замятие DV-ленты).

Список литературы

1. *Третьяков С.А.* CAN на пороге нового столетия // Мир компьютерной автоматизации. 1999. №2 (32). С.45-53.
2. Robert Bosch GmbH «CAN Specification Version 2.0», 1991. (42). С.100-110.

РЕКУРСИВНЫЙ АЛГОРИТМ

Юнусова Л.Р.[1], Магсумова А.Р.[2]

[1]Юнусова Лилия Рафиковна – магистрант;
[2]Магсумова Алия Рафиковна – магистрант,
направление: информатика и вычислительная техника,
магистерская программа: технология разработки программного обеспечения,
кафедра информационных систем,
отделение информационных технологий и энергетических систем,
Высшая инженерная школа
Набережночелнинский институт
Казанский федеральный университет,
г. Набережные Челны

Аннотация: *в статье рассматривается и анализируется понятие рекурсия, и его существующие методы. Описывается использование рекурсии, и как он представляется в программировании.*

Ключевые слова: *рекурсия, программирование, программа, подпрограмма.*

Рекурсия - это метод определения класса объектов или методов, который сначала определяет одну или несколько (обычно простых) баз или методов, а затем назначает основанное на них правило для создания прямой или косвенной ссылки на эти базовые случаи.

Другими словами, рекурсия - это способ определения объекта или действия сам по себе с использованием ранее определенного частного определения. Используйте рекурсию, когда вы можете выделить самоподобие задачи.

Рекурсивный алгоритм (процедура, функция):

– Если его определение содержит прямые или косвенные обращения к одному и тому же алгоритму, алгоритм называется рекурсией;

– Рекурсивная функция - одно из математических улучшений интуитивной концепции вычислимой функции.

Адаптивный рекурсивный алгоритм - алгоритм, который благодаря рекурсии учитывает определенные индивидуальные особенности, которые решают проблемы из определенной области.

Рекурсивная база - это предложение, которое определяет некоторые начальные условия или условия во время завершения. Обычно в этом предложении пишите простой случай, и вы можете получить немедленный ответ, даже если вы не используете рекурсию.

Рекурсивный шаг - это правило, тело которого должно содержать вызов определенного предиката в качестве дочерней цели.

Подпрограмма - это одна из вещей в рекурсивной функции.

Основное правило рекурсии: перед рекурсивным вызовом должна быть проверка, возвращаемая из рекурсии. Существуют следующие типы рекурсии:

– Прямая рекурсия - напрямую вызывать алгоритм (функцию, процедуру, метод) из текста самого метода;

– Косвенная рекурсия - существует несколько циклических последовательностей вызовов алгоритма;

– Линейная рекурсия - если подпрограмма выполняется только один раз для той же самой подпрограммы, то эта рекурсия называется линейной;

– Ветвь рекурсии - если каждый экземпляр подпрограммы может вызывать себя несколько раз, рекурсия называется нелинейной или «ветвью»;

– Бесконечная рекурсия (на самом деле это соглашение, потому что, когда память компьютера заполнена, программа выдаст ошибку и / или завершит работу в аварийном режиме).

Необходимо рассмотреть особенность рекурсивной программы, например, функции, используемые для генерации чисел Фибоначчи. Каждый рекурсивный уровень в функции Фибоначчи удваивает количество вызовов, поэтому количество рекурсивных вызовов, которые должны вычислить n-е число Фибоначчи, равно 2n.

Поэтому, если возможно, следует избегать рекурсивных программ, таких как программы, которые вычисляют числа Фибоначчи, что привело бы к экспоненциальному увеличению числа вызовов.Любая проблема, которая может быть решена рекурсивно, также может быть решена итеративно (а не рекурсивно). Рекурсивный алгоритм в программировании реализован в механизме так называемого рекурсивного процесса. Рекурсия - это подпрограмма, на которую прямо или косвенно ссылаются другие подпрограммы, и на которую можно ссылаться вместе с другими фактическими параметрами. В современных системах программирования стек обеспечивает правильную работу подпрограмм, особенно рекурсивных подпрограмм.

Аппаратный стек находится в ОЗУ, а указатель стека содержится в специальной паре регистров SS: SP для программиста. Аппаратный стек расширяется в направлении убывающего адреса с указателем на первый свободный элемент.

PASCAL, C и C ++ используют стек для размещения локальных переменных процесса и других программных блоков. Стек делится на сегменты, которые представляют собой блоки смежных ячеек. Каждый вызов подпрограммы использует фрагмент стека, длина которого зависит от вызывающей подпрограммы. Каждый раз, когда процесс активируется, стек выделяет память для своих локальных переменных, а когда процесс завершается, память освобождается. Поскольку вызовы процедур всегда строго вложены, в верхней части стека всегда есть память, содержащая локальные переменные для текущего активного процесса.

Поэтому, как правило, когда процедура A вызывает процедуру B, происходит следующее:

1. Фрагмент нужного размера размещается сверху стопки. Включает в себя следующие данные:

а) фактические параметры, указывающие на вызов процедуры;

б) пустая ячейка локальной переменной, определенной в процессе Б;

С) Обратный адрес, адрес команды, выполняемой в программе A, выполняется после того, как программа B завершила свою работу.

Если B является функцией, фрагмент стека b содержит указатель на ячейку в фрагменте стека a, в которую следует поместить значение функции (адрес значения).

2. Управление первым оператором, переданным в программу B.

3. После завершения шага B переведите управление на шаг A, используя следующую последовательность шагов:

а) получить адрес возврата с вершины стека;

b) Если B - функция, ее значение сохраняется в ячейке, указанной указателем на адрес значения;

C) извлечь фрагмент стека процесса b из стека и поместить фрагмент процесса a поверх стека;

d) Программа a возобновляется командой, указанной в адресе возврата.

Та же последовательность операций выполняется, когда подпрограмма вызывает себя, то есть в случае рекурсии.

Такой подход позволяет легко реализовать рекурсивный процесс. Когда процедура вызывает себя, новая память в стеке выделяется для всех ее локальных переменных, а вложенные вызовы представляются их собственными локальными переменными. Когда вложенный вызов завершается, он освобождает пространство памяти, занимаемое его переменными в стеке, и представление предыдущего уровня локальных переменных становится актуальным.

Список литературы

1. *Божко В.П., Власов Д.В., Гаспариан М.С.* Информационные технологии в экономике и управлении. Учебно-методический комплекс. М.: ЕАОИ. 2008 г. С.30-41
2. *Колмогоров А.Н.* Теория информации и теория алгоритмов. М.: Вильямс, 2017. 240 с.
3. *Коротков М.А., Степанов Е.О.* Основы теории алгоритмов. М.: Вильямс, 2016. 174 с.
4. *Мальцев, А.И.* Алгоритмы и рекурсивные функции: моногр. / А.И. Мальцев. М.: Вильямс, 2016. 346 с.
5. *Мейерс С.* Эффективный и современный C++: 42 рекомендации по использованию C++11 и C++14 -- М. Вильямс, 2016. 304 с.
6. *Третьяков С.А.* CAN на пороге нового столетия // Мир компьютерной автоматизации. 1999. №2 (32). С.45-53.
7. Robert Bosch GmbH «CAN Specification Version 2.0», 1991. (42). С.100-110.