МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Ереванский филиал федерального государственного бюджетного образовательного учреждения высшего образования
«РОССИЙСКИЙ ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ Г.В. ПЛЕХАНОВА»

Курсовая работа

на тему

Использование языка программирования FreeBASIC для вывода динамических данных в блоках информационных систем

по дисциплине: Проектирование информационных систем

Группа: ЕР-ДКИ-401

Студент: Бабушкин В.В.

Подпись *Дата*

Руководитель работы: Прохоренко Е.Б.

Подпись *Дата*

Ереван 2016

Содержание

ВВЕДЕНИЕ

Информационные системы (ИС) являются важнейшим звеном в процессе обмена информации без которого развитие технологии не представляется возможным. Именно ИС обеспечивают сбор, хранение, обработку, поиск, выдачу информации, необходимой в процессе принятия решений задач из любой области. Они помогают анализировать проблемы и создавать новые информационные продукты, позволяющие получить быстрый доступ к упроавлению колоссальными информационными ресурсами.

Разнообразие ИС дает возможность автоматизации практически любой области человеческой деятельности. Например, специализированные ИС позволяют увеличить продуктивность и производительность работы инженеров и проектировщиков, многопрофильные офисные ИС упрощают работу сотрудников любого организационного уровня, экспертные системы или системы обработки знаний, вбирают в себя знания, необходимые инженерам, юристам, ученым при разработке или создании нового продукта. Информационно-решающие системы осуществляют все операции переработки информации по определенному алгоритму, и возлагая на себя ответственность за проведение сложных математических вычислений значительно упрощают моделирование физических, экономических и т.д. процессов.

Часто при исследовании процессов нередко приходится моделировать ситуации, когда значение результативного признака в текущий момент времени t формируется под воздействием факторов, действовавших в прошлые моменты времени. Динамические данные в подобных системах могут зависить от начальных условий и также от значений параметров системы в предшедствующие моменты врмени. В данной работе мы исседовали самый простейший случай информационно-вычислительной системы, основанной на алгоритме модели случайного блуждания, в котором изменение параметров системы на каждом шаге не зависит от предыдущих и от времени. Мы рассмотрели процесс вывода изменения динамических перемненных, таких как направление движения и положение точки в пространстве в течение времени, а также возможность считывание начальных параметров из текстового файла на языке программирования FreeBasic. Конечным результатом стала разработка графического интерфейса, наглядно демонстрирующего процесс случайного блуждания в двумерном пространстве, который считывает входные параметры из текстового файла и в процессе выполнения создает лог-файл с значениями динамических переменных на каждый момент времени.

1 Определение и классификация информационных систем

1.1 Понятие информационной системы, выполняемые функции

Для более точного определения ИС сначала обратимся к понятию "системы" – как совокупности дискретных элементов с четко определенными выполняемыми функциями и тесно взаимодействующих друг с другом. Результатом такого тесного взаимодействия является восприятие системы как целого, гармонично функционирующего механизма. Примерами систем могут служить как и механические ситемы, например, автомобиль, так и биологические, например, человеческий организм. В зависимости от глубины дисекртизации системы можно выделить как и отдельные примитивные элементы, имеющих определенное функциональное назначение, так и совокупности из простых взаимосвязанных элементов, так называемых подсистем. Именно состав, упорядоченность и принципы взаимодействия элементов, иными словами, структура системы, является определяющим фактором для ее свойств и характеристик.

Что касается информационных систем (ИС), то их основным предназначением является сбор, хранение, обработку, поиск, выдачу информации, необходимой в процессе принятия решений задач из любой области. Они помогают анализировать проблемы и создавать новые информационные продукты. Иными словами, ИС является одним из составляющих звеньев процессов создания, сбора, обработки, накопления, хранения, поиска, распространения и потребления информации, так называемых информационных процессов. Именно эта последовательность своег рода "блоков" лежит в основе работы любой ИС, а именно, различают следующие этапы обработки информации в ИС

- ввод информации из внешних или внутренних источников;
- обработка входной информации и представление ее в удобном виде;
- вывод информации для представления потребителям или передачи в другую систему;
- обратная связь — это информация, переработанная людьми данной организации для коррекции входной информации.

Для реализации информационных процессов используются различные информационные процедуры, которые можно условно разделить на три категории в зависимости от метода обработки информации:

- с четко определенным и неизменным алгоритмом – так называемые полностью формализованные информационные процедуры;
- неформализованные информационные процедуры, при выполнении которых создается новая уникальная информация, причем алгоритм переработки исходной информации неизвестен ;

- плохо формализованные информационные процедуры, при выполнении которых алгоритм переработки информации может изменяться и полностью не определен;

В целом любая ИС может быть анализирована, построена и управляема на основе общих принципов построения систем, при этом она будет вполне динамичной и эволюционирующей с течением времени. Как на входе так и на выходе ИС мы будем иметь дело с информацией – входная информация задает начальные параметры задачи а на выходе мы получаем данные, на основе которых принимаются решения. Иными словами, ИС это система обработки информации, в которой человек пока еще остается существенным звеном.

1.2 Классификация информационных систем

В зависимости от предназначения и целей использования информационной системы различают несколько различных видов ИС. Например, по характеру представления и логической организации хранимой информации информационные системы подразделяются на:

- *фактографические*, хранящие данные в виде информационных объектов, т.е. множества экземпляров одного или нескольких типов структурных элементов. Каждый информационный объект отражает сведения по какому-либо факту, событию отдельно от всех прочих сведений и фактов. Как правило это четко структурированные системы с ярко выраженной иерархией;

- *документальные*, в отличие от фактографических, не являются структурированными, но могут устанавливать логическую взаимосвязь между документами. В таких системах единичным элементом информации является нерасчлененный на более мелкие элементы документ;

- *геоинформационные*, в которых данные организованы в виде отдельных информационных объектов, привязанных к общей электронной топографической основе (электронной карте) и употребляются для организации информационных процессов с пространственно-географической компонентой;

Если классифицировать по фуннкциональному признаку, определяющему предназначение, цели и задачи ИС, то различают *автоматизированные системы (АС)*, *системы поддержки принятия решений (СППР)*, *информационно-вычислительные системы (ИВС)*, *информационно-справочные системы (ИСС)* и *системы обучения.*

Другой подход к классификации ИС по уровням управления различает между *ИС оперативного* и *тактического* уровня, *ИС специалистов*, *ИС офисной администрации* и *стратегическими ИС. ИС оперативного уровня* обрабатывая данные о сделках и событиях, отвечают на запросы о текущем состоянии и отслеживают поток сделок в фирме способствуя оперативному управлению. *ИС тактического уровня* позволяют сравнивать текущие показатели с прошлыми, составлять отчеты и получать доступ к

архивам. *ИС специалистов* помогают специалистам, работающим с данными, повышают продуктивность и производительность работы инженеров и проектировщиков, интегрируя новые сведения в организацию и помогоют в обработке бумажных документов. *ИС офисной администрации* используются в основном административными сотрудниками, например, бухгалтерами, секретарями, клерками для обработки данных, повышения эффективности их работы и упрощения канцелярского труда а также для поддержания связи с покупателями, заказчиками и другими организациями. *ИС обработки знаний* и экспертные системы, вбирают в себя знания, необходимые инженерам, юристам, ученым при разработке или создании нового продукта. Их работа заключается в создании новой информации и нового знания. Так, например, существующие специализированные рабочие станции по инженерному и научному проектированию позволяют обеспечить высокий уровень технических разработок. *Стратегические ИС* — обеспечивают поддержку принятия решений по реализации перспективных стратегических целей развития организации.

По степени автоматизации ИС разделяются на ручные, автоматические и автоматизированные. В отличие от взаимно противоположных ручных и автоматических ИС, автоматизированные системы предполагают участие в процессе обработки информации как человека, так и компьютера.

По характеру использования информации различают *информационно-поисковые системы*, которые производят ввод, систематизацию, хранение, выдачу информации по запросу пользователя; *информационно-решающие системы* – осуществляют все операции переработки информации по определенному алгоритму; *управляющие информационные системы*–вырабатывают информацию, на основании которой человек принимает решение; *советующие информационные системы* – вырабатывают информацию, которая принимается человеком к сведению и не превращается немедленно в серию конкретных действий. Управляющие ИС способны работать с большими данными, а советующие ИС нацелены на работу со знаниями и могут использовать алгоритмы машинного обучения и искусственного интеллекта.

ИС также могут быть классифицированы по сфере применения на *ИС организационного управления* предназначенные для автоматизации функций управленческого персонала, *ИС управления технологическими процессами* для автоматизации функций производственного персонала, *ИС автоматизированного проектирования* для автоматизации функций инженеров-проектировщиков, конструкторов, архитекторов или дизайнеров, *интегрированные (корпоративные) ИС* для автоматизации всех функций фирмы, которые охватывают весь цикл работ от проектирования до сбыта продукции.

По способу организации групповые и корпоративные информационные системы подразделяются на системы на основе архитектуры файл-сервер, архитектуры клиент-сервер, многоуровневой архитектуры и на основе интернет/интернет-технологий.

1.3 Автоматизированные информационные системы

Автоматизированная информационная система (АИС) — система, состоящая из персонала и комплекса средств автоматизации его деятельности, реализующая информационную технологию выполнения установленных функций. Иными словами, это организованная совокупность средств, методов и мероприятий, используемых для регулярной обработки информации для решения задачи. Основным отличием АИС от ИС лежит в процессе, протекающим в АИС. Если автоматизируемый процесс связан в основном с обработкой информации, то такая система является автоматизированной информационной системой. Таким образом, АИС является частным случаем ИС.

Автоматизированные информационные системы могут быть классифицированы по направлению вида деятельности на производственные, административные, финансовые и учетные, и системы маркетинга.

Структуру информационной системы составляет совокупность отдельных ее частей, называемых подсистемами. Структура любой информационной системы может быть представлена совокупностью обеспечивающих подсистем, среди которых обычно выделяют информационное, техническое, математическое, программное, организационное и правовое обеспечение.

Информационное обеспечение — совокупность единой системы классификации и кодирования информации, унифицированных систем документации, схем информационных потоков, циркулирующих в организации, а также методология построения баз данных. Назначение подсистемы информационного обеспечения состоит в своевременном формировании и выдаче достоверной информации для принятия управленческих решений.Технологическое и организационное воплощение информационного обеспечения осуществляется в форме документационного управления, информационной и экспертно-аналитическая службы.

Для создания информационного обеспечения необходимо ясное понимание задач и целей предназначения АИС, представление о потоке информации и взаимодействии между различными компонентами АИС, представление о системе документооборота и возможности внесения изменений приводящих к более эффективному использованию документационных ресурсов также достигаемой путем внедрения упрощенной системы классификации и кодирования, способностью к концептуально-логическому моделированию взаимодействия информационных объектов и взаимосвязи информации а также сохранение информации на машинных носителях, что предполагает наличие технического обеспечния у разработчика.

Под *техническим обеспечением* понимается не только наличие вычислительных устройств, но и устройств сбора, накопления, обработки, передачи и вывода информации, передачи данных и линий связи, оргтехники и устройств автоматического съема информации а также расходные материалы и аксессуары. Техническое обеспечение может быть как *централизованным*, с центральной ИС на большонм компьютере (сервере), способной орабатывать запросы клиентов, так и *децентрализованным* (*распределенным*), предполагающим организацию подсистем на компьютерах клиен-

тов, интергирующимися в единую ИС. Однако наиболее оптимальным будет использование *частично-децентрализованной* организации технического обеспечения, когда общие для функциональных подсистем базы данных хранятся на распределенных сетях, состоящих из состоящих из персональных и больших компьютеров.

Для бесперебойного функционирования технических средств и взаимодействия между элементами ИС также необходимо предоставить *математическое и программное обеспечение*, представляющее собой совокупность математических методов, моделей, алгоритмов и программ для реализации целей и задач ИС. Примерами Математического и програмного обеспечения могут служить средства моделирования процессов управления, типовые алгоритмы управления, методы математического программирования, математической статистики, теории массового обслуживания и др.

Програмное обеспечение можно разделить на *общесистемные* и *специальные* программные продукты, а также *техническую документацию*. В качестве *общесистемных програмных продуктов* выступают программы для типовой обработки информации, расширяющие функциональные возможности ПК а также позволяющие управлять процессо обработки данных. Как правило, такие продукты более ориентированы на пользователя и обладают хорошо разработанным и удобным пользовательским интерфейсом.

Специальное программное обеспечение состоит ис комплекса программ, "заточенных" под конкретную ИС. В его состав входят пакеты прикладных программ, реализующие разработанные модели разной степени адекватности, отражающие функционирование реального объекта.

Техническая документация на разработку программных средств должна содержать описание задач, задание на алгоритмизацию, экономико-математическую модель задачи, контрольные примеры.

И, наконец, для полноценного функционирования ИС также должно быть уделено достаточно внимания для разработки *организационного* и *правового* обеспечений. Первое представляет собой совокупность методов и средств, регламентирующих взаимодействие работников с техническими средствами и между собой в процессе разработки и эксплуатации ИС. В спектр задач *организационного обеспечения* входит анализ существующей системы управления организацией и выявление задач, подлежащих автоматизации; подготовка к проектированию ИС технико-экономическое обоснование эффективности; разработка управленческих решений по составу и структуре организации, методологии решения задач, направленных на повышение эффективности системы управления.

Что касается *правового обеспечения* ИС, то оно преследует цель укрепления законности и включает в себя совокупность правовых норм, определяющих создание, юридический статус и функционирование информационных систем, регламентирующих порядок получения, преобразования и использования информации.

2 Язык программирования FreeBASIC

2.1 Краткий обзор языка программирования FreeBASIC

FreeBASIC это компиллируемый язык программирования высокого уровня, основанный на компиляторе с открытыми исходными кодами (FreeBASIC Compiler), позволяющий писать программное обеспечение под DOS, Windows, Linux и MacOS. Компиллятор распространяется на условиях GNU GPL, стандартная библиотека — на GNU LGPL. По синтаксису FreeBASIC наиболее близок с языком программирования QuickBASIC корпорации Microsoft, его компилятор также совместим с QuickBASIC, так как изначально язык развивался как альтернатива и замена QuickBASIC, но быстро превратился в мощный, кроссплатформенный, свободный инструмент.

В FreeBASIC были добавлено множество расширений и возможностей для соответствия современным требованиям, стандартам и совместимостью с библиотеками и API написанными на C/C++. Начиная с версии 0.17 пользователь может выбрать три режима компиляции (опция -lang): режим совместимости с QuickBASIC (qb), старыми версиями FreeBASIC (deprecated), и стандартный режим, включающий в себя все последние изменения и возможности.

Язык FreeBASIC был разработан в 2004 году Andre Victor T. Vicentini и поддерживается так называемой The FreeBASIC Development Team. Изначально был написан на Visual Basic, но в последствии его исходные коды были переписаны на FreeBASIC. Возможности FreeBASIC значительно уступают такому популярному языку как C++, однако это компенсируется простым, удобным и более легким для изучения синтаксисом и может служить промежуточным звеном на пути к изучению более комплексных языков программирования. Как и C++, FreeBASIC является объектно-ориентированным языком, что позволяет ознакомиться с концепциями ООП на примере более простого и интуитивно понятного кода.

Исходный код программы пишется в редакторе и после компилируется. В качестве наиболее удобной среды программирования, поставляемой вместе с компиллятором можно использовать FBIDE, однако также доступны множество других сред разработки ПО, распространяемых как на коммерческой основе так и находящихся в открытом доступе.

Многие команды языка несут в своих названиях большой смысл делающим понятным большинство команд. Так например: Print - печатать Sleep - усыпить программу Draw - рисовать Goto - перейти на нужную строчку End - конец программы. В FreeBASIC исползьуются ограничения на наименование переменных, стандартные для большинства языков программирования. Например, имя переменной не должно начинаться с цифры а также не должно иметь сходство со встроенными операторами или командами языка.

FreeBASIC вбирает в себя, по мере своего развития, все наилучшее от языков Basic и C++. На данном этапе развития он уже обладает впечатляющим набором возможностей, таких как:

- Поддержка Unicode (как исходники, так и строки)
- Множество встроенных типов переменных общих для большинства языков программирования (Byte, UByte, Short, UShort, Integer, UInteger, LongInt, ULongInt, Single, Double, String, ZString, WString)
- обработка ошибок
- Типы данных определяемые пользователем (бесконечная вложенность, Union, тип поля (array, function, bit fields))
- Пространства имён (namespace)
- Перечислимый тип (Enum)
- Новые возможности при работе с массивами (до 2 ГБ размером, Redim Preserve)
- Указатели (указатели на любые типы данных, неограниченная косвенная адресация)
- Перегрузка функций и операторов
- Необязательные аргументы функций
- Встроенный ассемблер (ассемблерные инструкции в исходном коде программы)
- поддержка функций языка C и C++
- мощные Препроцессоры
- Typedefs
- разбиение выполнения на несколько потоков
- классы и объекты
- Конструкторы и деструкторы классов

Вместе с FreeBASIC в комплект входят также наиболее распространенные библиотеки (с подключаемыми заголовочными файлами):OpenGL, GTK+, SDL, Allegro, ODE, Newton, BASS, Fmod, FreeImage, Zlib, sqlite, MySQL, PostgreSQL, Lua, LibXML, и многие другие.

FreeBasic сейчас находится в активной стадии развития. Его функциональные возможности увеличиваются с каждой новой версией.

2.2 Ввод-вывод данных в языке программирования FreeBASIC

Ввод - это те данные, которые получает программа из вне (от пользователя, из файла, и пр.). Вывод - это извлекаемые данные из программы в определенное место , например на экран, в файл, на принтер и пр. Input и Output (ввод и вывод) часто сокращают как I/O, или IO. FreeBASIC имеет множество методов ввода и вывода. Самой базовой командой выводящая данные на экран является `PRINT`. Для ввода данных в программы используется команда `INPUT`, записывающая данные, введенные пользователем в соответствующие переменные в программе.

Наконец, FreeBASIC способен работать с файлами, а именно, открывать файл для прочтения данных и присвоения этих данных переменным в программе, а также для записи величин переменных программы в файл. Для создания , открытия файла в FreeBASIC есть функция `OPEN`. Она имеет как бы один параметр в котором сочетаются несколько параметров с ключевыми словами, которые разделены пробелами. Никаких разделяющих запятых нет.
Синтаксис:
`OPEN`(имя файла, `FOR` {`INPUT` | `OUTPUT` | `APPEND`}, `AS` дескриптор)
`OPEN`(имя файла, `FOR BINARY`, `ACCESS`{`READ` | `WRITE` }, `AS` дескриптор)
`OPEN`(имя файла, `FOR RANDOM`, `ACCESS`{`READ` | `WRITE` }, `AS` дескриптор [размер буфера])
где "име файла" – название файла с указанием полного пути к нему, следующий параметр `FOR` задает режим работы, который может быть как текстовым так и двоичным. В текстовом режиме нужно указать тип открытия или создания:

- `INPUT` - только для чтения (если файл не будет найден, функция вернет код ошибки с помощью функции `ERR`.)
- `OUTPUT` - только для записи(если файла не существует, то создаст его. Если файл есть, то перезапишет)
- `APPEND` - для записи в конец файла (предыдущая записанная информация в файле сохраняется)

Дополнительно можно указать режим кодировки с помощью ключевого слова `ENCODING` и строки с нужной кодировкой для открытия файла: ASCII или Unicode. По умолчанию кодировка ASCII.

Двоичный режим работы имеет только два параметра: `BINARY` для записи или чтения двоичных стандартных типов данных (`BYTE`, `INTEGER` и пр.); и `RANDOM` для работы с двоичными данными нестандартных типов данных. Дополнительно в двоичном режиме можно указать доступ к файлу с помощью ключевого слова `ACCESS` с параметрами: `READ`– только чтение и `WRITE` – только запись. Если ничего не указывать, то по умолчанию и то и другое.

Дескриптор файла необходим для функций позволяющих считывать информацию из файла или записывать ее в файл и является идентификатором файла. Записывается дескриптор с помощью символа ♯ и цифры (либо переменной, олицетворяющей эту

цифру). Последний параметр размер буфера для чтения или записи. По умолчанию буфер равен 128 байт.

Для записи в файл или чтения из него, который открыт в текстовом режиме, применяются команды `PRINT ♯` (запись) и `INPUT ♯`(чтение). При работе с файлами эти функции используются практически так же как в консоли. Функция `PRINT` при работе с файлами, дополнительно имеет один параметр в начале. Этим параметром является дескриптор файла, тот же что и для функции `OPEN`. В остальном никаких отличий нет, если конечно принять во внимание что информация выводится не в консоль, а в файл.

Однако, для того чтобы прочитать строку с запятыми придется использовать `LINE INPUT`, так как `INPUT` разделит строку на аргументы. Так же можно применять для записи в файл функцию с расширенным форматированием `PRINT ♯ USING`.

Для определения положения указателя в файле используется функция `EOF` (End of file). Если она возвращает ненулевое значение (-1), то указатель достиг конца файла. Это часто бывает полезным, например при прочтении всего файла.

При работе с большим кол-вом файлов, все время нужно помнить, какие номера дескрипторов задействованы. При создании или открытии нового файла, нужно точно знать: свободен ли определенный номер дескриптора. Чтобы не держать эту информацию в голове и не рыться лишний раз в своем исходном тексте, можно просто и легко получать действительно свободный дескриптор с помощью функции `FREEFILE`.

2.3 Форматированный вывод с помощью `PRINT ♯ USING` и `LPRINT`

Функции `PRINT ♯ USING` и `LPRINT` позволяют добиться форматированного вывода информации в файл или в командную строку. В отличие от команды `PRINT`, которая позволяет слитный вывод с помощью оператора ";" и табулированный вывод с помощью оператора "," , команда `PRINT USING` позволяет записывать значения с использованием расширенного форматирования. У этой команды два параметра:

- любая строка, содержащая символы нужного форматирования,
- список аргументов.

Принцип ее действия таков:

Берется строка из первого параметра как основа, далее в зависимости от специальных символов форматирования, размещенных в ней, размещаются аргументы из второго параметра, которых может быть несколько. Специальные символы подразделяются на 2 группы:

Для работы со строковыми типами аргументов

`!` – печатает первый символ строки аргумента,

`\\` – печатает пару символов строки аргумента, кол-во печатываемых символов может изменяться кол-вом пробелов между `\\`,

`&` – печатает всю строку.

Для работы с числовыми типами аргументов

`♯` - резервирование места для одной цифры из аргумента п печать. Кол-во цифр регулируется кол-вом этого символа,

`&` – печатает все число,

`.` – ставится точка между `♯` для определения дробных чисел,

`,` – ставится между символами `♯` для разделения числа в группы по три цифры,

`+` – ставится тандемом с `♯` для вывода знака числа,

`^` – ставится между символами `♯` и дает возможность записи, экспоненциального представления числа. Число при этом округляется.

Если нужно в строке использовать один из выше описанных символов не для форматирования, а просто как часть строки, то перед этими символами нужно ставить знак нижнего подчеркивания `_` например `_!`. Если при использовании форматирующих символов, места для всех цифр не хватает, то перед числом выводится знак `%`.

Команда `SPC` позволяет разделять слова нужным кол-вом пробелов. У нее всего один параметр: число нужных пробелов.

Функция `LPRINT` используется для записи текстовой информации на принтер по умолчанию. FreeBASIC использует специальный номер файла -1 для печати с помощью `LPRINT`. Это номер файла может быть безопасно закрыт с помощью `CLOSE` -1. При следующем использовании `LPRINT` автоматически откроет его по мере необходимости.

Функция `CLOSE` позволяет закрывать открытые дескрипторы. Это надо делать в обязательном порядке. Если же открыт только один файлом, то номер дескриптора после функции `CLOSE` указывать не обязательно.

3 Работа с файлами и вывод динамических данных на примере модели случайного блуждания

3.1 Случайное блуждание как математическая модель

Для демонстрации возможностей файлового ввода/вывода в языке FreeBASIC мы исследовали самый простейший случай информационно-вычислительной системы, основанной на алгоритме модели случайного блуждания.

Случайное блуждание это стохастический процесс при котором объекты, движущиеся произвольным образом, постепенно удаляются от точки отправления совершая определенное количество шагов в дискретные моменты времени. При этом предполагается, что изменение на каждом шаге не зависит от предыдущих и от времени. Классический пример — простое случайное блуждание на квадратной решетке, моделирующий своего рода "реальный"процесс – допустим пьяный матрос стартует из бара в начале координат, проходит один квартал в случайно выбранном направлении (на север, запад, юг или восток), после чего повторяет процедуру на каждом перекрестке, независимо от того, что происходило ранее. Очевидно, что его среднее расстояние от бара будет зависеть от времени. Изначально может показаться что наиболее вероятное положение матроса - это у двери бара. Однако чем больше число "шагов"тем дальше матрос будет удаляться от бара. В результате получается следующая зависимость между количеством шагов N и средним расстоянием от бара r:

$$\langle r^2 \rangle = N \tag{1}$$

С течением времени - увеличением числа шагов - ширина распределения растет, кривая "расплывается" – матрос оказывается все дальше и дальше от бара.

Информационно-вычислительные системы основанные на алгоритмах случайного блуждания широко используются в физике, биологии, медицине, социологии и других областях науки. Например, случайное блуждание может быть применено для моделирования таких физических процессов, как броуновское движение и других диффузных процессов а также для симуляции передвижения, распространения и распределения популяции животных или микроорганизмов.

3.2 Чтение исходных параметров ИС

Информационно-вычислительной система моделирующая случайное блуждание считывает начальные параметры, заданные в файле `parameters.txt`. В качестве начальных параметров мы выбрали количество шагов (переменная `numberOfSteps`), координаты начальной позиции на решетке (`startPositionHorizontal` и `startPositionVertical`, причем в качестве начала отсчета выбран верхний левый угол решетки) и скорость движения матроса (`speed`) – то есть время необходимое для совершения одного шага в миллисекундах. На рис. 1, рис. 2 и рис. 3 представлена графическая симуляция

траектория движения пьяного матроса, подсчитанная информационно-вычислительной ситемой для различного числа шагов.

Информация, записанная в файл `parameters.txt` выглядет следующим образом:

```
numberOfSteps,10000
startPositionHorizontal,30
startPositionVertical,30
speed,100
```

В самой программе мы сначала задаем имя файла с параметрами:

```
dim As String readfile = ''parameters.txt''
dim As Integer reader = Freefile
```

Затем мы проверяем существует ли такой файл в пакете с программой – если нет, то присваиваем переменным значения по умолчанию, если да – то читаем значения из файла:

```
'first check whether the parameters file exists
If dir(readfile)='' '' Then:
      ' if ''parameters.txt'' file does not exist - use default values
      speed = 100'default value
      startPositionHorizontal = 2 '2 squares left by default
      startPositionVertical = 5 '5 squares down by default
      numberOfSteps = 10000 'default value
'if ''parameters.txt'' file exists - read the variables from the ''parameters.txt''
 Else:
      Open readfile For Input As reader
      While Not Eof(reader)
            count +=1
            Redim Preserve varName(count), varValue(count)
            Input #reader, varName(count), varValue(count)
      Wend
      Close reader
EndIf
```

Для чтения файла мы обьявляем имя файла `readfile` как строковую константу и получаем свободный дескриптор `reader` для этого файла (целочисленная переменная) с помощью функции `FreeFile`. Затем открываем его для только чтения с помощью функции `Open`, сообщая ей параметры `Input` (только для чтения) и наш дескриптор. Затем, пока дескриптиор не указывает на конец файла, мы увеличиваем переменную `count`, изначально равную -1 и указывающую на ячейку массивов `varName` (строковые данные) и `varValue` (целочисленные данные), в которые мы записываем имена переменных и их величины соответственно. Так как мы используем массивы переменной длины, то нам необходимо обращаться к функциям `Redim` и `Preserve` для изменения размеров массива и перезаписи нового массива в ту же область памяти, в которой хранился сатрый массив. После выхода из цикла мы закрываем файл сообщая функции `Close` его дескриптор.

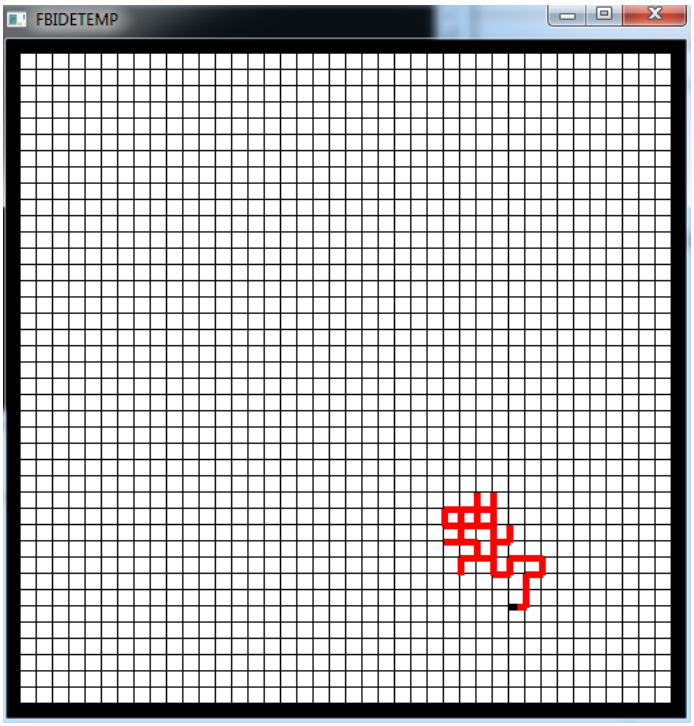

Рис. 1: Траектория случайного блуждения, симулированная информационно-вычислительной системой для $N = 100$ шагов.

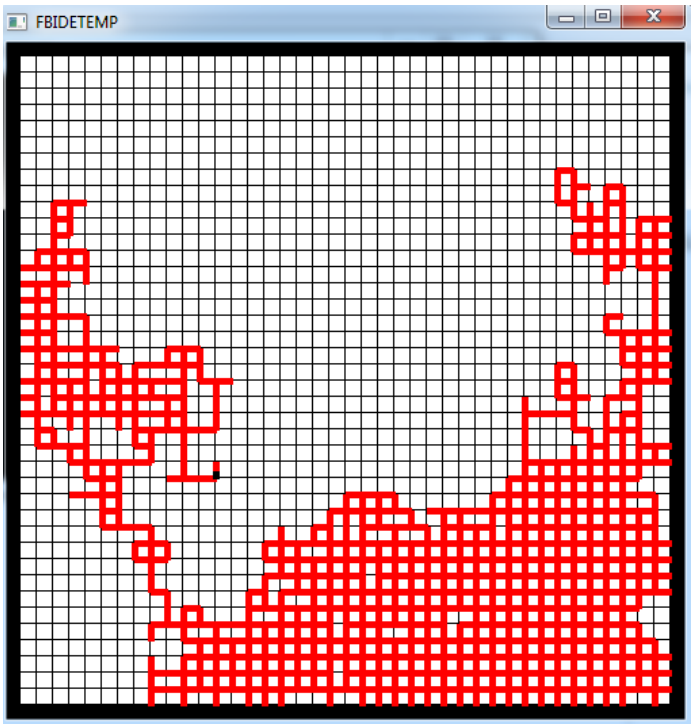

Рис. 2: Траектория случайного блуждения, симулированная информационно-вычислительной системой для $N = 2500$ шагов.

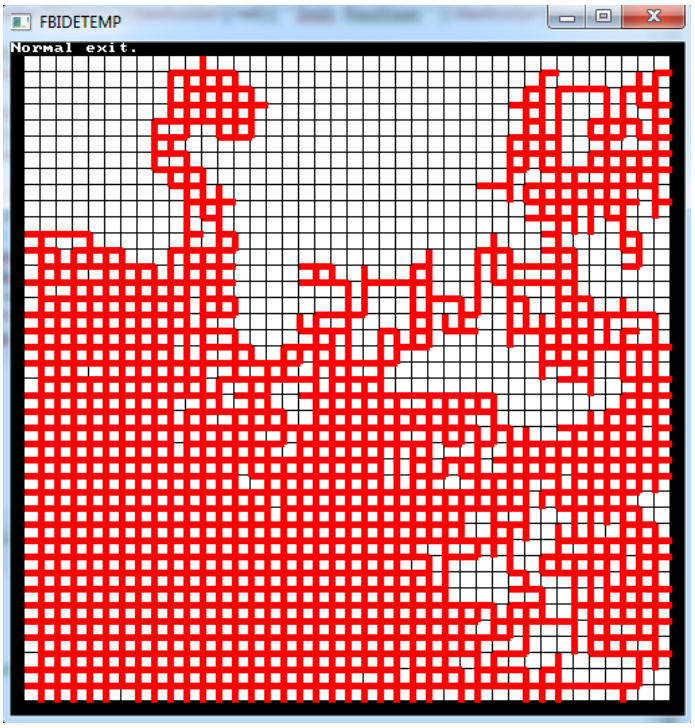

Рис. 3: Траектория случайного блуждения, симулированная информационно-вычислительной системой для $N = 10000$ шагов.

3.3 Вывод динамических данных ИС

Теперь рассмотрим как именно можно вывести динамические переменные информационно-вычислительной ситемы, моделирующей случайное блуждание в файл. Заметим, что в нашем случае следующие данные являются динамическими:

1. `Step` – хранит в себе шаг на данном этапе,
2. `Time` – момент нахождения в данной точке решетки, соответстственно равен 0 для начальной точки
3. `Direction` – направление движения пьяного матроса – принимает 4 значения – вверх, вниз, налево и направо,
4. `InitPosition` – позиция в данный момент времени
5. `NextPosition` – позиция, в которую он перейдет на следующем шаге.

Для записи в файл мы опять определяем имя файла как строковую переменную `printfile`и его дескриптор как целочисленную перемненную `printer`, которую затем получаем от функции `Freefile`:

```
dim As String printfile = ''path.txt''
dim As Integer printer = Freefile
```

Затем, перед началом итерирования по заданному количеству шагов мы объявляем заголовки таблицы и распечатываем их в наш файл использую функцию `Lprint`:

```
Lprint   ''Step'' & Space(10-Len(''Step''))&
         ''Time, msec'' & Space(20-Len(''Time, msec''))&
         ''Direction'' & Space(15-Len(''Direction''))&
         ''Init Position'' & Space(15-Len(''Init Position'')) &
         ''Next Position'' & Space(15-Len(''Next Position''))
```

Мы использовали функции `Space` и `Len` для того чтобы красиво отформатировать столбики в таблице. Функция `Len` возвращает длину строки, передаваемой ей в качестве параметра, а функция `Space` отступает на количество пробелов, переданное ей в качестве параметра. Таким образом, например выражение `Space(10-Len(''Step''))` означает что функция `Space` отступает от последнего символа строки `''Step''` на 10−`Len(''Step'')` $= 10 - 4 = 6$ пробелов, в итоге, следующая строка будет расположена на расстоянии 10 пробелов от начала страницы.

Далее в программе мы опять обращаемся к функции `Lprint` для вывода численных значений переменных в файл. В конце программы мы закрываем файл `printfile` передавая функции `Close` его дескриптор `printer`.

В таблице 1 представлен пример файла с выводом динамических данных.

Step	Time, msec	Direction	Init Position	Next Position
1	0	LEFT	(370,370)	(358,370)
2	99	UP	(358,370)	(358,358)
3	199	UP	(358,358)	(358,346)
4	299	LEFT	(358,346)	(346,346)
5	399	LEFT	(346,346)	(334,346)
....	.....	.....	.....	.....
16	1499	UP	(370,358)	(370,346)
17	1599	RIGHT	(370,346)	(382,346)
18	1699	DOWN	(382,346)	(382,358)
.....	.....	.....	.....	.....
5478	548280	DOWN	(262,466)	(262,478)
5479	548380	RIGHT	(262,478)	(274,478)
.....	.....	.....	.....	.....
9997	1000278	RIGHT	(298,298)	(310,298)
9998	1000378	RIGHT	(310,298)	(322,298)
9999	1000479	UP	(322,298)	(322,286)
10000	1000578	DOWN	(322,286)	(322,298)

Таблица 1: Файл вывода динамических данных Информационной вычислительной системы, моделирующей процесс случайного блуждания с параметрами `numberOfSteps = 10000, startPositionHorizontal = 30, startPositionVertical = 30, speed = 100`.

ЗАКЛЮЧЕНИЕ

При выполнении курсового проекта с целью изучения вывода динамических данных в блоках ИС было произведено тщательное исследование типов информационных систем и способов их классификации. Затем мы представили краткое описание основных функций языка программирования FreeBASIC и возможности, которые он представляет. Далее был детально изучен способ ввода-вывода переменных в языке программирования FreeBASIC.

Для практической демонстрации возможностей языка программирования FreeBASIC при выводе динамических переменных была разработана информационно - вычислительная система, моделирующая самый простой случай стохастического процесса случайного блуждания на двумерной решетке. Мы детально представили те части кода, которые ответственны за ввод и вывод данных а также дали описание всех динамических переменных, инициализируемых на каждом шаге случайного блуждения.

Как и любая информационная система, разработанная нами программа не является совершенной и допускает следующие возможные технические и логические улучшения:

1. мы предполагаем, что пьяный матрос всегда движется в одном из четырех направлений. Было бы интересно рассмотреть случай когда он также мог оставаться неподвижным. В этом случае алгоритм потребует большее число шагов и будет наблюдаться совешенно другое отношение между квадратом расстояния от бара и количеством сделанных шагов.

2. еще одна модификация предполагает наличие нескольких таких "матросов". В этом случае можно было бы изучить распределение их расстояний от бара в зависимости от количества сделанных шагов а также динамику расстояний между ними с течением времени. Такая система будет манипулировать куда большим количеством динамических переменных.

3. с технической точки зрения можно было бы модифицировать информационно-вычислительную систему так, чтобы она создавала еще один файл, куда она могла бы записывать количество пройденных шагов, начальное и конечное положение матроса и затраченное время. В этом случае анализ данных намного упроститься.

4. также к функциональности данной информационно-вычислительной системы может быть добавлена возможность генерировать графики функции расстояния от бара в зависимости от числа пройденных шагов.

5. наименование файла `''path.txt''` практически неудобно и является плохим примером документирования работы информационной системы. В следующих версиях желательно динамически изменять имя файла так чтобы оно отражало интересующие нас параметры, например, количество шагов. В этом случае будет сгенерировано более чем один файл, однако это упростит дальнейший анализ данных.

Информационно-вычислительные системы основанные на алгоритмах случайного блуждания широко используются в физике, биологии, медицине, социологии и других областях науки. Например, случайное блуждание может быть применено для моделирования таких физических процессов, как броуновское движение и других диффузных процессов а также для симуляции передвижения, распространения и распределения популяции животных или микроорганизмов.

4 Список литературы

[1] Челкак Дмитрий. Случайные блуждания на плоскости и их пределы: простое случайное блуждание, lerw и saw. http://www.mccme.ru/dubna/2013/courses/chelkak.htm.

[2] Станислав Будинов. Сайт для поддержки русскоязычных пользователей freebasic. http://free-basic.ru/fileswork.html.

[3] В.Б. Уткин К.В. Балдин. *Информационные системы в экономике.* Москва, 2008.

[4] Ледовских И. А. *Теория и практика построения информационных систем и баз данных.* Москва, 2010.

[5] Н.Н. Карабутов. *Информационные технологии в экономике.* Экономика М., 2002.

[6] А.А. Хлебников. *Информационные системы в экономике.* Феникс, Ростов н/Д., 2007.

[7] Е.В. Михеева. *Информационные технологии в профессиональной деятельности.* Академия, Москва, 2008.

[8] Г.А. Титоренко. *Информационные системы в экономике: учебник для студентов вузов, обучающихся по специальностям «Финансы и кредит», «Бухгалтерский учет, анализ и аудит» и специальностям экономики и управления (060000).* ЮНИТИ-ДАНА, Москва, 2008.

[9] Е.В. Михеева. *Практикум по информационным технологиям в профессиональной деятельности.* Академия, Москва, 2014.

[10] AndreaSchmidt. Random walks. http://www.mit.edu/~kardar/teaching/projects/chemotaxis(AndreaSchmidt)/random.htm.

[11] Y. D. Liang. *Introduction to programming using Python.* Pearson, Boston, 2013.

[12] Wikipedia. Random walk. https://en.wikipedia.org/wiki/Random_walk.

Дополнительные материалы

Информационно-вычислительная система состоит из двух файлов:

`parameters.txt`

`randomwalk_v2.bas`

которые следует поместить в одну и ту же папку

Содержание файла `parameters.txt`

```
numberOfSteps,10000
startPositionHorizontal,30
startPositionVertical,30
speed,100
```

Содержание файла `randomwalk_v2.bas`

```
declare sub gfx_box(buffer as integer ptr, x1 as integer, y1 as integer,_
x2 as integer, y2 as integer, col as integer,_
scr_size_x as integer)

screenres 500,500,32,1
ScreenSet 1,0

dim as integer screenSizeX = 500
dim as LongInt numOfSquaresHorizontal= 40
dim as LongInt numOfSquaresVertical = 40
dim as LongInt squareSize = 10
dim as LongInt indent = 2
dim as LongInt lineWidth = 3
dim as LongInt x0 = 10
dim as LongInt y0 = 10
dim as LongInt gridCornerX = x0
dim as LongInt gridCornerY = y0
dim as LongInt startX
dim as LongInt startY
dim as LongInt direction
dim as LongInt currentX
dim as LongInt currentY
dim as LongInt minX
dim as LongInt minY
```

```
dim as LongInt maxX
dim as LongInt maxY
dim as LongInt speed
dim as LongInt startPositionHorizontal
dim as LongInt startPositionVertical
dim as LongInt numberOfSteps

dim Start As Double

dim as String stepStr
dim as String timeStr
dim as String dirStr
dim as String initPositionStr
dim as String nextPositionStr

dim as LongInt nextX
dim as LongInt nextY

dim as integer i

'output stream variables
dim As String printfile = "path.txt"
dim As Integer printer = Freefile
Open printfile For Output As #printer
#UnDef Lprint
#Define Lprint Print #printer,

'array to store all nodes' coordinates
dim As LongInt xCoordArray(numOfSquaresVertical)
dim As LongInt yCoordArray(numOfSquaresHorizontal)

dim as LongInt  row, col

'input stream variables
dim As Integer varValue()
dim As String  varName()
dim As String  lname()
dim As Integer count = -1
dim As String readfile = "parameters.txt"
dim As Integer reader = Freefile

'first check whether the parameters file exists
```

```
If dir(readfile)="" Then:
' if "parameters.txt" file does not exist -- use default values
speed = 100 'default value
startPositionHorizontal = 2'2 squares left by default'
startPositionVertical = 5 '5 squares down by default
numberOfSteps = 10000 'default value

Else:
' if "parameters.txt" file  exists -- read the variables from the file
Open readfile For Input As  reader
While Not Eof(reader)
count +=1
Redim Preserve varName(count), varValue(count)
Input #reader, varName(count), varValue(count)
Wend
Close reader

'set the variables from "parameters.txt" file
For idx as Integer = 0 To count
If varName(idx) = "numberOfSteps" Then:
numberOfSteps = varValue(idx)
EndIf
If varName(idx) = "startPositionHorizontal" Then:
startPositionHorizontal = varValue(idx)
EndIf
If varName(idx) = "startPositionVertical" Then:
startPositionVertical = varValue(idx)
EndIf
If varName(idx) = "speed" Then:
speed = varValue(idx)
EndIf
Next
EndIf

'draw the grid
For row=1 To numOfSquaresVertical
For col=1 To numOfSquaresHorizontal
xCoordArray(col-1)= gridCornerX
gfx_box(screenptr, gridCornerX, gridCornerY, gridCornerX+squareSize,_
gridCornerY+squareSize, rgb(255,255,255), screenSizeX)
gridCornerX = x0+col*(squareSize+indent)
Next col
```

```
yCoordArray(row-1)= gridCornerY
gridCornerX = x0
gridCornerY = y0+row*(squareSize+indent)
Next row

minX = xCoordArray(0)
maxX = xCoordArray(numOfSquaresVertical-1)+squareSize

minY = yCoordArray(0)
maxY = yCoordArray(numOfSquaresHorizontal-1)+squareSize

startX = xCoordArray(startPositionHorizontal)
startY = yCoordArray(startPositionVertical)

nextX = startX
nextY = startY

Lprint "Step" & Space(10-Len("Step"))&_
"Time, msec" & Space(20-Len("Time, msec"))&_
"Direction" & Space(15-Len("Direction"))& _
"Init Position" & Space(15-Len("Init Position")) &_
"Next Position" & Space(15-Len("Next Position"))

Start = Timer

For i=1 To numberOfSteps
'direction legend:
'1-left
'2-right
'3 -up
'4-down

'decide which direction to go
randomize(timer)
Randomize

'generate a random integer from 1 to 4
direction = int(4*Rnd+1)

'update X coordinates
If nextX > minX and nextX < maxX Then:
currentX = nextX
```

```
EndIf
'update Y coordinates
If nextY > minY and nextY < maxY Then:
currentY = nextY
EndIf

screenlock

'LEFT
If direction = 1 Then:
dirStr = "LEFT"
nextX = currentX - squareSize - indent
nextY = currentY

gfx_box(screenptr, nextX, currentY -lineWidth/2, currentX,_
nextY+lineWidth/2, rgb(255,0,0),500)
'draw a 'black head' pointing the direction of the movement
gfx_box(screenptr, nextX, currentY -lineWidth/2, nextX+squareSize/2,_
nextY+lineWidth/2, rgb(0,0,0), 500)

EndIf

'RIGHT
If direction = 2 Then:
dirStr = "RIGHT"
nextX = currentX + squareSize + indent
nextY = currentY

gfx_box(screenptr, currentX, currentY - lineWidth/2, nextX,_
nextY + lineWidth/2, rgb(255,0,0), 500)
'draw a 'black head' pointing the direction of the movement
gfx_box(screenptr, nextX-squareSize/2, currentY - lineWidth/2, nextX,_
nextY + lineWidth/2, rgb(0,0,0), 500)

EndIf

'UP
If direction = 3 Then:
dirStr = "UP"
nextX = currentX
nextY = currentY - squareSize - indent
```

```
gfx_box(screenptr, currentX-lineWidth/2, nextY, nextX+lineWidth/2,_
currentY, rgb(255,0,0), 500)
'draw a 'black head' pointing the direction of the movement
gfx_box(screenptr, currentX-lineWidth/2, nextY, nextX+lineWidth/2,_
nextY+squareSize/2, rgb(0,0,0), 500)

EndIf

'DOWN
If direction = 4 Then:
dirStr = "DOWN"
nextX = currentX
nextY = currentY + squareSize + indent

gfx_box(screenptr, currentX-lineWidth/2, currentY, nextX+lineWidth/2,_
nextY, rgb(255,0,0), 500)
'draw a 'black head' pointing the direction of the movement
gfx_box(screenptr, currentX-lineWidth/2, nextY-squareSize/2,_
nextX+lineWidth/2, nextY, rgb(0,0,0), 500)

EndIf

screenunlock

' use this trick to convert to the strings for file output
stepStr = ""& i &""
timeStr = ""& int((Timer-Start)*1000)&""
initPositionStr = "(" & currentX & "," & currentY & ")"
nextPositionStr = "(" & nextX & "," & nextY & ")"

'pause for speed milliseconds to create an impression of animated movement
sleep speed

Lprint stepStr & Space(10-Len(stepStr))&_
timeStr & Space(20-Len(timeStr))&_
dirStr & Space(15-Len(dirStr))&_
initPositionStr & Space(15-Len(initPositionStr)) &_
nextPositionStr & Space(15-Len(nextPositionStr))

'press 'q' to terminate the movement
If Inkey = "q" Then:
exit for
```

```
EndIf

're-color the 'black heads' back to red
If direction = 1 Then:
If nextX+squareSize/2 > minX and nextX+squareSize/2 < maxX Then:
gfx_box(screenptr, nextX, currentY -lineWidth/2, nextX+squareSize/2,_
nextY+lineWidth/2, rgb(255,0,0), 500)
EndIf
EndIf

If direction = 2 Then:
If nextX-squareSize/2 > minX and nextX-squareSize/2 < maxX Then:
gfx_box(screenptr, nextX-squareSize/2, currentY -lineWidth/2, nextX,_
nextY+lineWidth/2, rgb(255,0,0), 500)
EndIf
EndIf

If direction = 3 Then:
If nextY+squareSize/2 > minY and nextY+squareSize/2 < maxY Then:
gfx_box(screenptr, currentX-lineWidth/2, nextY, nextX+lineWidth/2,_
nextY+squareSize/2, rgb(255,0,0), 500)
EndIf
EndIf

If direction = 4 Then:
If nextY-squareSize/2 > minY and nextY-squareSize/2 < maxY Then:
gfx_box(screenptr, currentX-lineWidth/2, nextY-squareSize/2,_
nextX+lineWidth/2, nextY, rgb(255,0,0), 500)
EndIf
EndIf

Next i

Close #printer

'_____
Print "Normal exit."

sleep
```

```
'routine to draw a filled rectangle
sub gfx_box(buffer as integer ptr, x1 as integer, y1 as integer, _
x2 as integer, y2 as integer, col as integer, scr_size_x as integer)
dim as integer x, y, blit_corr

x = (x2 - x1) + 1
y = (y2 - y1) + 1

buffer += (y1 * scr_size_x) + x1
blit_corr = (scr_size_x - x) shl 2

asm
mov edi,[buffer]
mov edx,[y]
mov ebx,[x]'<-|
mov eax,[col]'|
nl_box:'      |
mov ecx,ebx'<--difference = not fetching var W but keep it in a register
rep stosd
add edi,[blit_corr]
dec edx
jnz nl_box
end asm
end sub
```