УДК 519.85
DOI: 10.20535/SRIT.2308-8893.2017.3.09

ИНТЕЛЛЕКТУАЛЬНАЯ ВИЗУАЛЬНАЯ 3D+ ПОЛИГЛОТ-КОНЦЕПЦИЯ ПРОГРАММИРОВАНИЯ БЕЗ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ

И.В. ВЕЛЬБИЦКИЙ

Аннотация. Предложена простая, интеллектуальная и математически строгая концепция программирования ориентированными графами из только горизонтальных дуг, на которых сверху и снизу записываются тексты на любых языках — русском, английском, китайском, математическом и т.д. Математический язык строго определен как часть элементарной математики, которая задает выражения и формулы для записи условий и действий на графических дугах новой концепции и выполнения их на компьютере. Традиционные языки программирования не используются, но их библиотеки (интеллект) сохраняются с возможностью доступа к ним из графических программ с использованием соответствующих функций с параметрами или без. Любая программа из этой библиотеки может быть автоматически переведена в единую новую графическую оболочку. Такой граф является интеллектуальным полиглотом, он может быть 3D+, имеет стандарт ISO 8631/1989, и только один эффективно используется на протяжении всего жизненного цикла процесса разработки, выполнения и эксплуатации программ. В новой концепции используется единый математический (графический) метод для записи алгоритмов, данных, программ и сетевых графиков их разработки, который имеет гораздо лучшие (до 100+ раз) характеристики по наглядности, простоте, компактности, скорости их ввода в компьютер и эффективности полученного в результате компьютерного кода. Приведена история возникновения и проверки новой концепции, описание ее сути, реализованной графической среды программирования и мнение некоторых экспертов и пользователей о преимуществах ее использования.

Ключевые слова: программирование, интеллектуальное, визуальное, полиглот, 3D+, ориентированные графы, только горизонтальные дуги, нагрузка только дуг, любой язык (русский, английский, китайский, математический и т.д.), цвет, чертеж, доказательный стиль, оптимизация, самодокументирование, документипрвание мотивации, кибербезопасность, большие данные.

ВВЕДЕНИЕ

В 1960-х годах нами впервые был введен термин «технология программирования» [1,2] как интуитивное осознание того факта, что главным является не только язык (в то время Алгол-60), а и как его использовать и писать на нем *грамотно* и *культурно*, *легко* и *просто*. В настоящее время технология программирования включена во все IT-программы обучения. Но программирование стало еще более *сложным, дорогим*, *недоступным* и *непонятным*. Сейчас после появления новой нагруженной по дугам графической концепции программирования стало ясно, что причина этому является старая неизменная с 1947 г. и *устаревшая* неграфическая концепция программирования.

В этой концепции используются машинно-ориентированные и неестественные для человека операторы типа: **if-then, else, for, while, goto ..., метки**, блочные скобки типа **begin-end, {-} ..., отступы** (лесенки), большинство знаков препинания и т.д., которые **все** *исключены* из новой концепции программирования (рис. 1) и которых слишком много в существующих текстах программ. Такие операторы сложны, маломощны, эмпиричны (строго не определены) и обеспечивают только примитивно-ремесленническую технологию работы. Для *нейтрализации* их недостатков пользователь тратит слишком много усилий, создавая огромное количество новых языков, методов и сред программирования, которые якобы упрощают, а на самом деле делают программирование неунифицированным, еще более сложным, запутанным, непонятным и недоступным для всех.

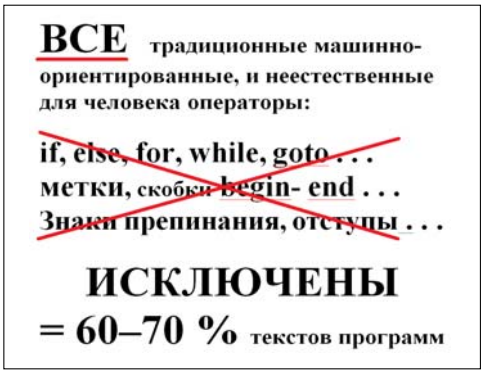

Рис. 1. Существующая с 1947 г. концепция программирования

Программирование в графах* формируется в 70-х в течении 15 лет при разработке больших систем управления основных ракет бывшего Советского Союза [3] как технологический процесс формального документирования их разработки для быстрого внесения постоянных исправлений и улучшений, с одной стороны, и работ Дейкстры [4], который впервые показал математическую *нестрогость* и *избыточность* традиционной концепции программирования, с другой стороны. В 2016 году окончательно сформирована новая, *очень простая* графическая концепция программирования *единая(!)* для задания алгоритмов, программ, данных и сетевых графиков на всем жизненном цикле разработки и эксплуатации программ. Проведен ее всесторонний *анализ*, опытная *реализация(!)* и *эксплуатация* новой графической среды программирования (ГСП) [5–8].

Новая концепция впервые предлагает не писать, а *рисовать* программы в графах, нагруженных *по горизонтальным дугам* символами, выражениями и функциями из элементарной математики. Рисовать *во много раз* проще, быстрее, нагляднее и компактнее, чем писать программу в существующей концепции (в специальных текстовых языках) программирования. При этом используется та же клавиатура, мышь или сенсорный экран и обеспечивается соответствие получаемого рисунка стандарту ISO 8631/1989 [5]. Такие графы и записи на них являются полиглотами, понятными без сложных определений. Они сливаются с известными, строгими математическими принципами представления и обработки информации, которые приближаются к принципам работы мозга. Поэтому *все* специальные языки программирования типа Фортран, C++, JAVA, Python становятся маломощными и ненужными, но сохраняются их библиотеки и специализированные полезные про-

* В математике в теории графов есть только **два** эквивалентных типа графов, которые могут быть использованы в программировании. Это граф, нагруженный по вершинам (например, известные блок-схемы, автоматы Мура), и граф, нагруженный по дугам (сетевые графики, автоматы Мили).

граммы типа **Oracle Database, разработки Websites** и т.д. Любая программа из библиотеки на любом языке может быть автоматически переведена в новую графическую форму представления. Благодаря этому упрощается переход на новую концепцию и сохраняется и многократно преумножается опыт программирования.

Новая концепция позволяет записать любой алгоритм, любую программу любым известным методом от структурного до глубокого **интеллектуального анализа информации, обеспечивая кибербезопасность, с использованием** уже готовых специализированных программ и библиотек, которые могут быть объединены в единую графическую **гипербиблиотеку.** Новая концепция вводит в программирование доказательный и безошибочный стиль, самодокументирование и документирование мотивации принимаемых решений, позволяет легко, быстро, **наглядно** и **компактно (!)** нарисовать логическую схему проблемы и решить ее. Принципы новой концепции так просты и естественны, что программирование будет **доступно** не только программистам, а станет элементом всеобщей грамотности и культуры общества.

СУТЬ НОВОЙ КОНЦЕПЦИИ ПРОГРАММИРОВАНИЯ

В новой концепции для записи любых алгоритмов, данных и программ используется только *одна* направленная влево или вправо горизонтальная дуга графа, так называемая *логическая R-схема* или *аксиома* (рис. 2). На этой дуге сверху записано *Условие*, а снизу — *Действия*, которые выполняются, если Условие «истинно». Для записи на дуге *Условие* и *Действия* в одну или несколько строк используются любые естественные языки (английский, русский, китайский и т.д.), символы, выражения и функции элементарной математики, соответствующие фрагменты из любых языков программирования для записи *Условия* и *Действий* и, прежде всего, функций обращения к библиотекам этих языков (рис. 3).

Рис. 2. Аксиома новой концепции программирования

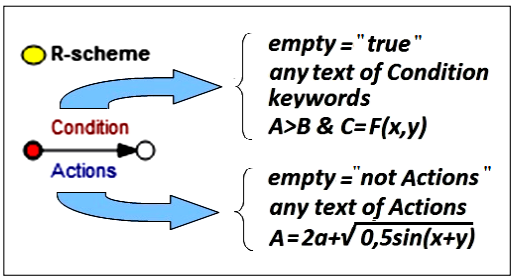

Рис. 3. Запись *Условия* и *Действий* на дуге в одну или несколько строк

Из вершины графа может исходить любое число дуг влево и/или вправо. Для записи графов типа «петли» (рис. 4, слева) используются либо пустая дуга, обозначающая безусловный переход без выполнения каких-либо *Действий*, либо специальная двойная горизонтальная дуга типа «знак равно» между вершинами (рис. 4, справа). Дуги, исходящие из любой вершины, анализируются последовательно сверху вниз. Выполняется первая дуга, у которой *Условие* «истинно». При этом выполняются все *Действия* под этой дугой и осуществляется переход по

стрелке дуги в новое состояние графа (алгоритма, данных или программы). Если все дуги, исходящие из вершины, имеют «ложные» *Условия*, то осуществляется переход по стрелке последней просматриваемой дуги в соответствующее новое состояние графа (программы) без выполнения *Действий*.

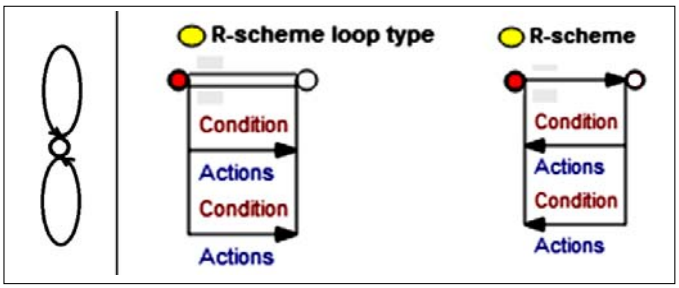

Рис. 4. R-схемы графов типа «петли»

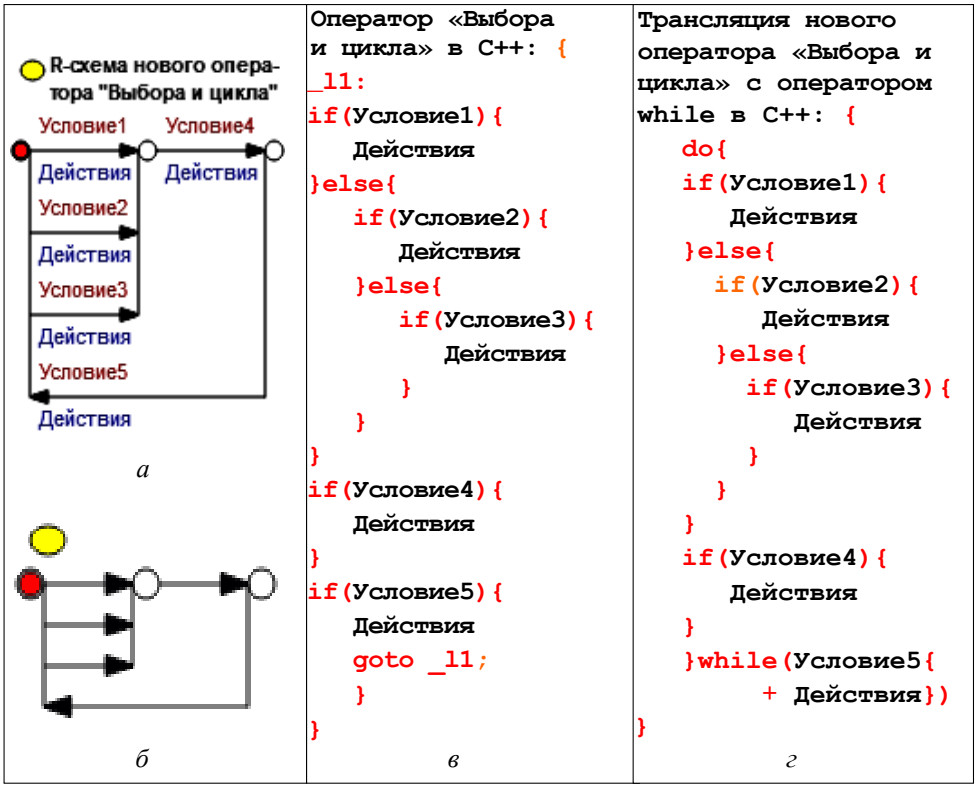

Рис. 5. Запись нового оператора «Выбора и Цикла» в графах и в C++

Пример определения нового оператора: «трех (может быть любое число) Условий и Цикла» показан на рис. 5. На рис. 5, *а* показана его R-схема (**11** строк), на рис. 5, *б* — его *абстрактная* R*-схема (**4** строки) или запись R-схемы (алгоритма, программы) без деталей реализации, без записей на дугах, а на рис. 5, *в* — эквивалентная традиционная запись в C++ (**24** строки), где красным отмечены *лишние* символы для R-схемы на

рис. 5, *а*. А на рис. 5, *в* их **76** или с отступами и переводами строк **217** (**73%**). Запись R-схемы в **2**(24/14), а R*-схемы в **6**(24/4) раз компактнее записи их в C++ или R=**2** и R*=**6** соответственно. Ввод одного (из 10) оператора C++ (рис.5, *в*) в компьютер занимает в среднем **10**(100/10) кликов клавиатуры, а ввод дуги R-схемы (рис.5, *а*) занимает **0,8** (4/5), что в **13** (10/0,8) раз меньше, eR=**13** и потому быстрее. Порядок цифр преимуществ новой концепции сохраняется при использовании в соответствующем трансляторе R-схем в ГСП оператора **while** на выходе (рис.5, *г*). Максимальные известные в настоящее время RR*-схемы программ имеют характеристики: R=**133**, R*=**400** и eR=**578**. Это значит, что они имеют компактность в 133 и 400 раз больше (в 133 и 400 раз имеют меньше строк), а скорость ввода в компьютер быстрее в 578 раз по сравнению с традиционной записью этих же программ в C++

Ввод каждой горизонтальной дуги графа выполняется одним нажатием мыши или клавиатуры или пальца по сенсорному экрану. При одном нажатии может быть введено сразу несколько новых горизонтальных дуг. Вертикальные дуги и вершины графа не вводятся в компьютер и рисуются в ГСП автоматически, что намного упрощает и ускоряет ввод графических операторов (дуг) по сравнению с вводом существующих операторов традиционных языков программирования.

Такой граф имеет имя, которое записывается сверху около желтого эллипса, рис. 2–7. В математике это имя соответствует имени некоторой функции, заданной графически в новой концепции программирования. Для реальных графических программ [9, 10] это имя функции может быть с параметрами или без них, рис. 6, 7. *Программа* в новой концепции задаетя любым числом взаимосвязанных по имени таких графов. Первый граф (R-схема) такой программы называется *аксиомой.* Первая слева вершина аксиомы программы всегда выделяется **красным**. В аксиоме залается архитектура соответствующей ООП программы: данные и соответствующие им методы. Если данные простейшие, как на рис. 6, 7, то они задаются традиционным способом: сверху дуги записывается ключевое слово соответствующего класса данных, а под дугой список его идентификаторов. Если данные логически сложные, то они задаются графом, у которого на дуге задаются сами данные (их синтаксис), а снизу семантика (методы) их обработки. Граф задает логическую структуру данных любой сложности..

Графическая ООП-программа, которая в **7** раз **компактнее,** чем традиционная запись этой программы в языке Delphi [9], показана на рис. 6, другая ООП-программа, которая в 4 раза более компактна, чем традиционная запись этой программы в C++, R=4, показана на рис. 7 [10, с. 489–491]. Это значит, что в этих программах Условия и Действия записаны на языке Delphi и C++ соответственно. На рис. 6 и 7 первая R-схема (аксиома с **красной** точкой) задает формулу, архитектуру определения соответствующих ООП, а именно, Данные (FIELDS для Delphi и privat для C++) и связанные с ними методы. Далее приведено определение четырех Методов для Delphi и 5+1 public-функции для C++ соответственно.

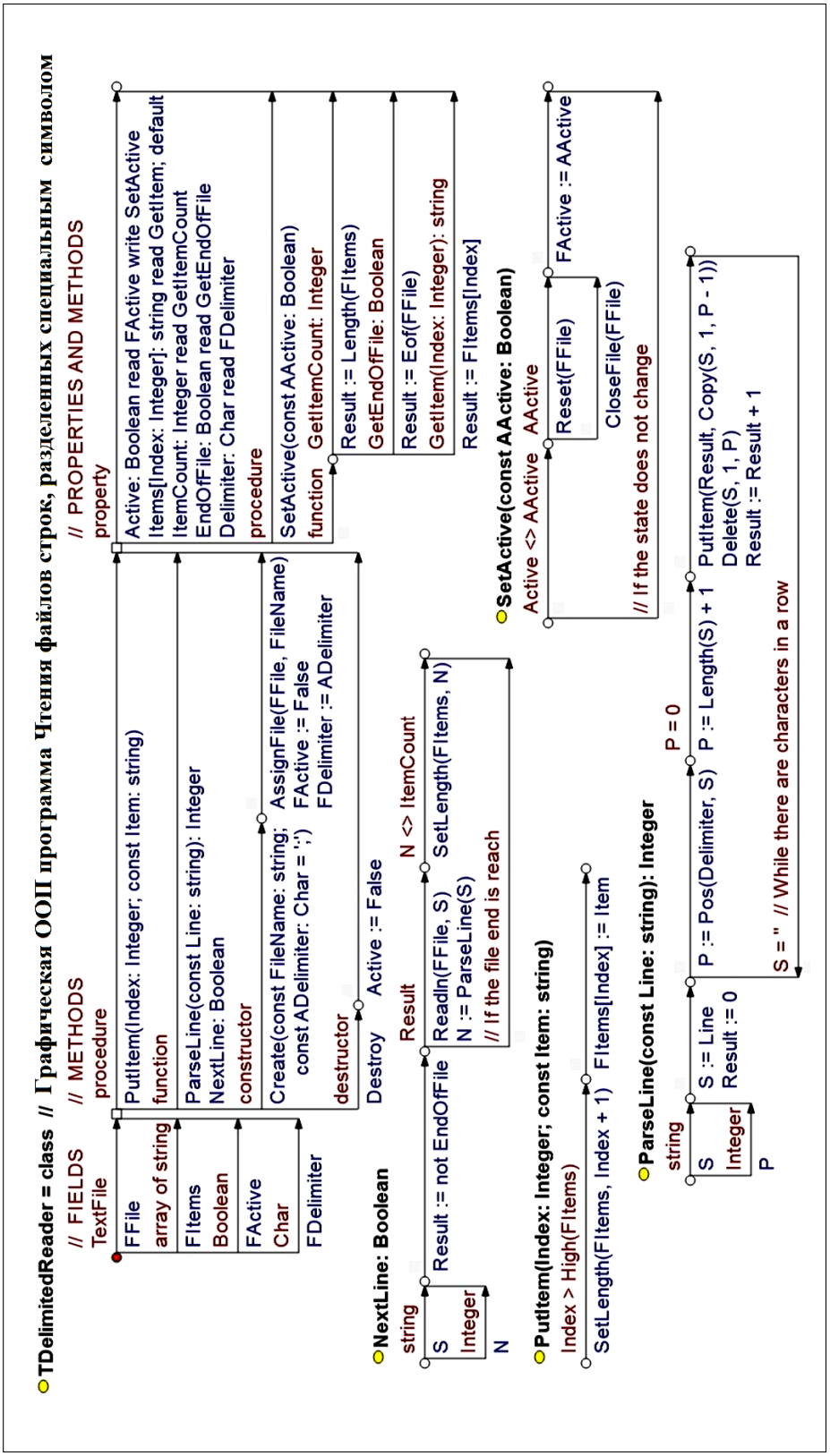

Рис. 6. R-схема ООП-программы

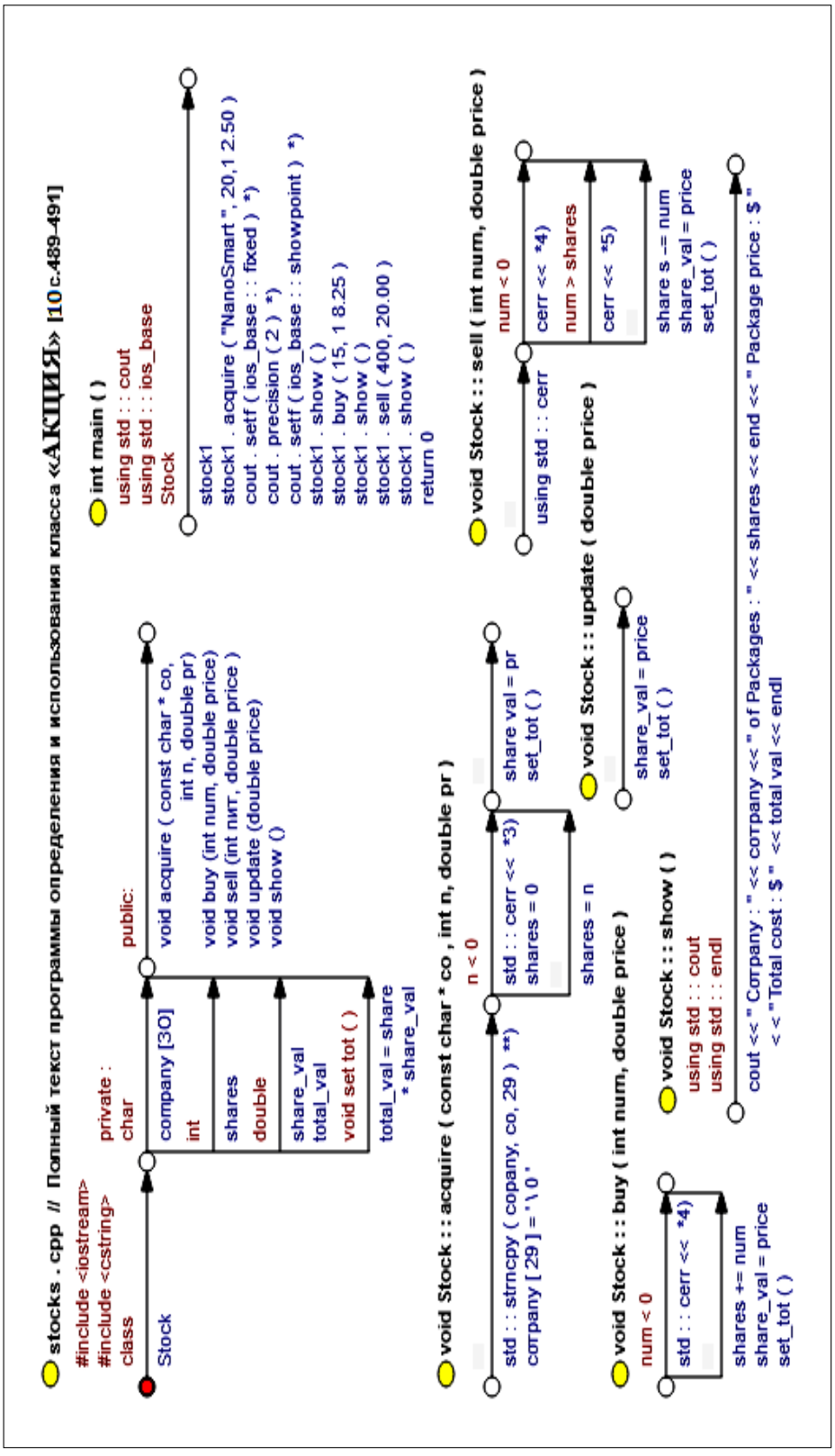

Рис. 7. R-схема ООП-программы «Определения и использования класса АКЦИЯ»

ГДЕ следующие обозначения показаны на рис. 7:

*) // формат #.## с 2-мя знаками после точки и показом нулей в итоге

**) // усечь со для помещения в company

*3) «Количество пакетов не может быть отрицательным; для» << « установлено в 0.\n»

4) «Количество приобретаемых(продаваемых) пакетов не может быть отрицательным.» 6

5) «Вы не можете продать больше того, чем владеете!» 6

*6) << «Транзакция прервана.\n»

Показанная на рис.7 графическая ООП-программа в C++ иллюстрирует способ записи комментариев и макроопределений после префикса **ГДЕ**, используемого в математике и потому более естественный, компактный и эффективный по сравнению с записью их в традиционной концепции программирования.

На рис. 8, 9 показаны *абстрактные* R*-схемы программ. *Абстрактная* R*-схема — это запись ее R-схемы без деталей реализации, без записей на дугах, т.е. максимально компактно с изображением только математической логики сути программы. Это новое теоретичекое понятие в программировании для компактной записи, *анализа*, графической *сертификации и синтеза (оптимизации)* программ по различным критериям: времени, памяти и т.д. Логическая R-схема и абстрактная R*-схема новой концепции являются новой **единой** графической оболочкой (эталоном, шаблоном) для записи программ и библиотек программ на всех языках. Это открывает новые возможности объединения интеллекта, сосредоточенного в библиотеках всех существующих языков программирования, и эволюционного их развития вместе с человеком, соответствующим коллективом разработчиков и общества в целом. Такой возможности НЕТ в традиционном программировании.

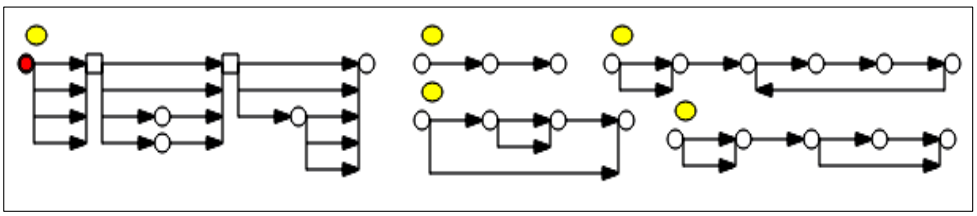

Рис. 8. R*-схема записи программы без деталей реализации и в **36** раз компактнее записи ее в Delphi

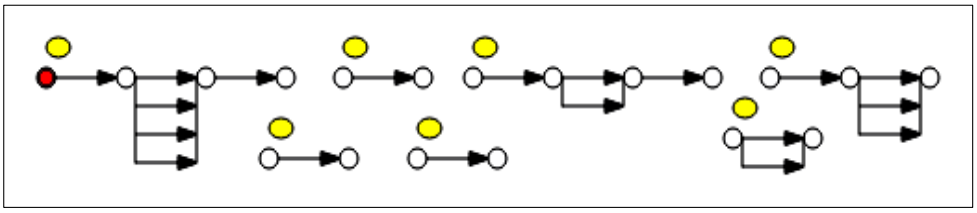

Рис. 9. R*-схема записи программы без деталей реализации и в **32** раза компактнее записи ее в C++

Оба примера на рис. 6–9, иллюстрирующие организацию ООП в различных самых современных ООП-ориентированных языках — C++ и Delphi, записаны программистами, которые использовали все тонкости и особенности этих языков для эффективной записи соответствующих программ. Графическая запись этих программ свидетельствует об эффективности (преимуществе) RR*-схем, т.е. простоты, понятности, наглядности, компактности и самодокументированности графов программ. Такое преимущество [10] подтверждает общую эффективность предлагаемой новой графической полиглот-концепции программирования. Новая графическая концепция позволяет также *объективно* (!) и количественно сравнить два новых языка, выяснить, какой язык лучше, например, Delphi, Java или C++, дополнительно сравнить количественные характеристики абстрактной R*-программы из $10\times10=100$ дуг на рис.10, которая компактнее C++ в $R=15$, $R^*=45$ раз и вводится в компьютер в $eR=138$ раз быстрее, а R-схема из 100x100 дуг имеет $R=133$, $R^*=400$ и $eR=578$. Новая концепция является *полиглотом,* потому что используемые в ней математика и графика интернациональны и лучше воспринимаются человеком, чем существующие искусственные языки программирования.

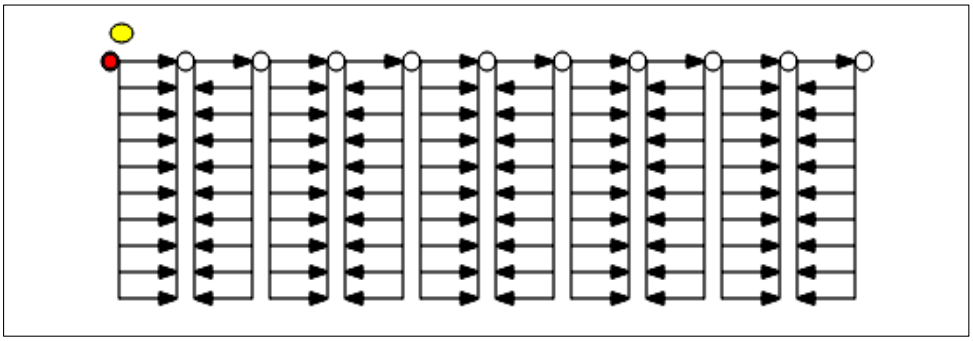

Рис. 10. Ввод R-схемы (программы) из $10\times10=100$ дуг: $R=15$, $R^*=45$, $eR=138$ по сравнению с C++

Для задания структур и классов данных, методов ООП, функций настройки в новой концепции используются *ключевые* слова типа: **int, array, class, table, function**, **#include <iostream> ... ,** которые записываются на дуге сверху и всегда истинны (см. рис. 6, 7). Эти дуги обычно *безальтернативны*, независимы друг от друга и могут выполняться *параллельно в* соответствии с ключевым словом на дуге. Используемая в них информация является, как правило, вспомогательной и служит для нейтрализации недостатков традиционной концепции программирования и упрощения процессов трансляции в ней. В графическом программировании эту информацию можно просто получить из контекста R-схемы и потому со временем может быть исключена из графического программирования полностью или частично, как это произошло в математике. На рис. 6, 7 она сохранена. Принцип сохранения изображен на рис. 7 при определении нового класса «АКЦИЯ» и традиционной записи этой же информации в языке C++ (рис. 11) [10, с. 489].

```
// stocks.cpp – полный текст программы
#include <iostrem>
#include <cstring>
class Stock   // объявление класса
{
private :
        char company[30];
        int shares;
        double share_val;
        double total_val;
        void settot () { total_val = shares * share_val;
public :
        void acquire (const char * co, int n, double pr);
        void buy (int num, double price);
        void sell ( int num, double price);
        void update (double price);
        void show ();
};                          и т о г о : R=(17/13)  R*=(17/4)=4,3
```

Рис. 11. Листинг10.3. Полный текст программы stocks.cpp

Сложные структуры данных могут непосредственно задаваться в графическом виде R-схем (рис. 12). В этом случае на дуге сверху записывается элемент логики или синтаксиса данных, а снизу – соответствующая семантика [2]. RR*-схемы эфективно используются для записи алгоритмов «от данных», например, трансляторов (рис. 12). Пример задания в R-схемах табличных данных показан на рис. 13.

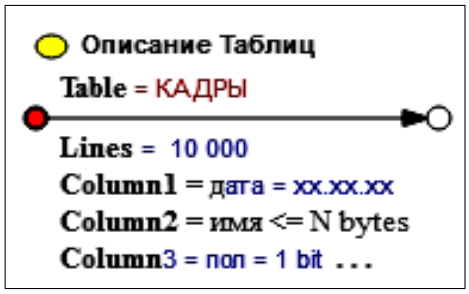

Рис. 12. Задание структур данных

Рис. 13. Задание табличных данных

Вершины и дуги графа могут иметь различную *конфигурацию* и *цвет,* (рис. 4–15). На рис. 6, 8 квадратной вершиной отмечены дуги, которые могут выполняться параллельно, а на рис. 16 вершиной в виде параллелограмма изображен принцип 3D и многомерное выполнение графических программ, а прямоугольная вершина (типа поплавок, на рис. 14), например, может запоминать число обращений к ней и иметь порог чувствительности в своей работе аналогично работе мозга у человека, чего нет в традиционном программированиии. На рис. 4, 12, 17, 18 используется конфигурация дуги в виде знака «равно» для наглядного изображения

графа типа «петли» и т.д. Цвет вершин широко используется для реализации светофоров в программе (рис. 14, 15), а цвет в дугах — для наглядного обозначения, например, маршрутов автоматической генерации тестов программ (рис. 14) и т.д.

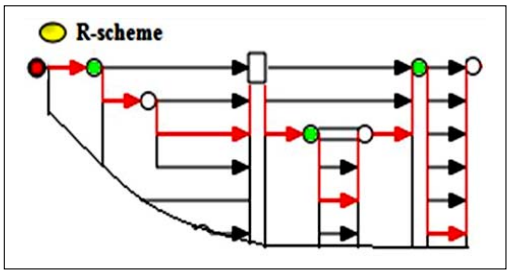

Рис. 14. Задание табличных данных

Цвет вершин и дуг может легко осуществляться (задаваться) из прграммы, например, функция **GREEN= green(...)** на рис. 15 *c*, *d*. Это новая возможность, отсутствующая в традиционном программировании или сложно моделирующаяся традиционными средствами. В новой концепции впервые появилась возможность естественно и просто документировать мотивацию принимаемых решений, программировать параллельно различные части проекта программ, автоматически генерировать функционально-полный набор тестов, проводить отладку программ параллельно с разработкой и т.д. На рис. 15 продемонстрирована еще одна новая (отсутствующая в традиционном программировании) возможность *автоматической оптимизации* R*-схем по их не оптимальной, но полученной в результате более простой процедуры проектирования R-схемы: «**SPIDER**» **Cycle ➔**.

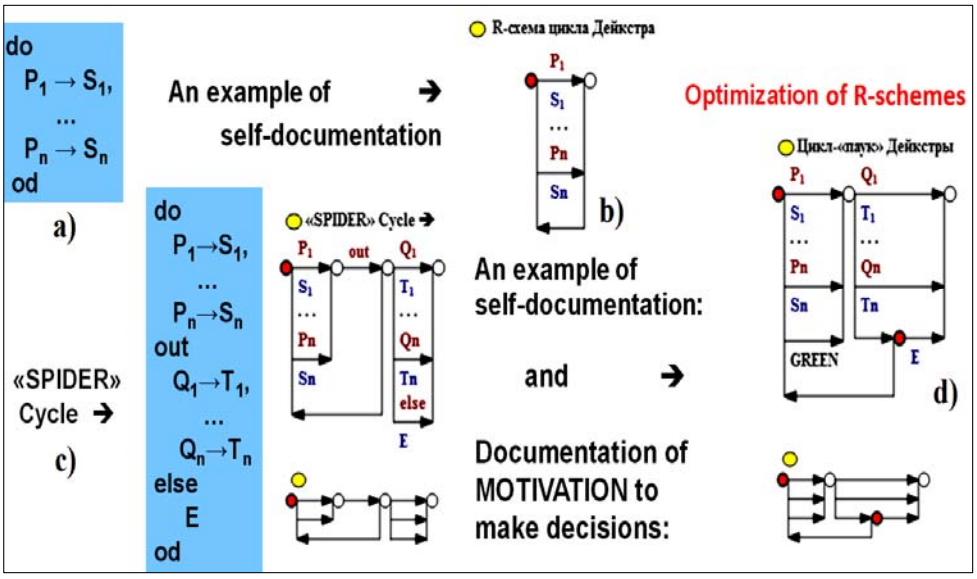

Рис. 15. Охраняемые команды Дейкстра и оптимизация графических программ с использованием «*светофора*» на последней вершине R-схемы

В теории построения аппаратуры самого компьютера на логических интегральных схемах используются автоматы Мили и Мура, которые зада-

ются графами, нагруженными по дугам (рис. 17, 18), что открывает новые большие перспективы построения компьютеров с новой гибкой архитектурой, задаваемой соответствующими графическими программами (R-схемами) и с аппаратной реализацией соответствующих библиотек. Все это позволяет значительно упростить и ускорить программирование и увеличить *уровень киберзащищенности, технологичности и автоматизации* разработки графических программ в таких компьютерах.

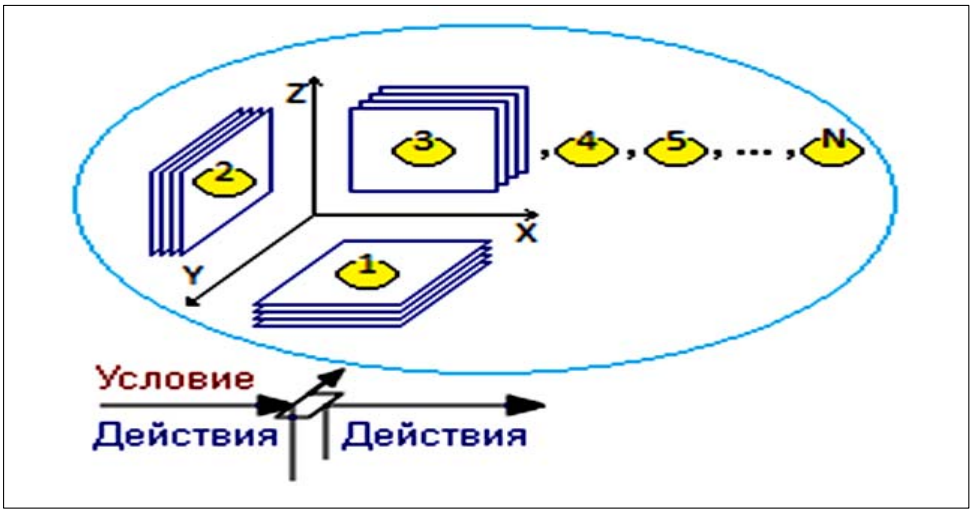

Рис. 16. Организация 3D программирования

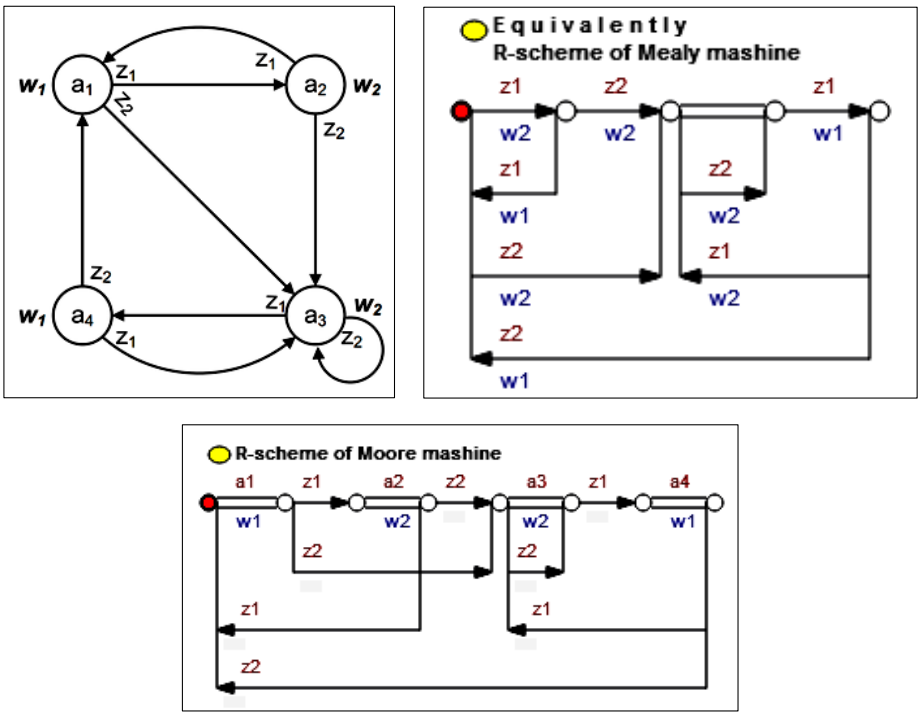

Рис. 17. R-схема автоматов Mealy and Moore

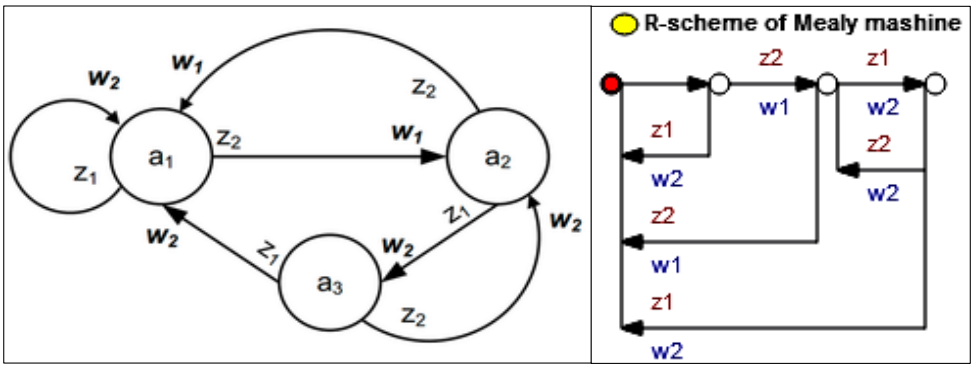

Рис. 18. R-схема автомата Mealy

На всем жизненном цикле новой концепции чертеж проекта программы совпадает с самой программой, документацией и *сетевым графиком* разработки (рис. 19). Руководитель работ **впервые** может около каждой работы (проекта программы) на дуге R-схемы записать в особых скобках имя исполнителя, расчетное время ее выполнения и *проконтролировать* качество выполняемых работ. Такого нет в существующем программировании и промышленности, где, например, чертеж (документация) автомобиля *отличается* от самого автомобиля и сетевого графика его разработки.

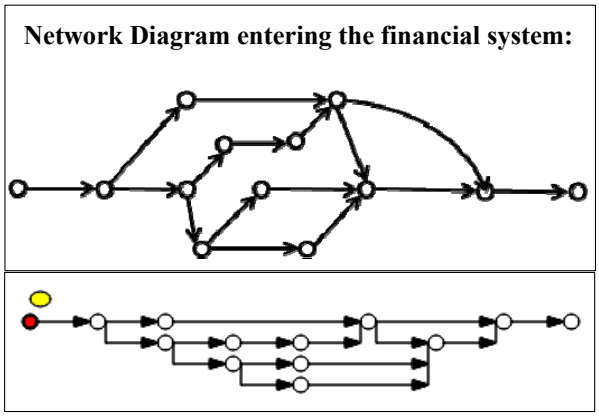

Рис. 19. Запись в R*-схеме сетевого графика ввода в эксплуатацию финансовой системы

Графическая форма записи имеет большие резервы по развитию в будущем. Для записи вершин, дуг и записей на них может использоваться цвет. Вершины и стрелки дуг могут иметь разную конфигурацию, а дуги быть двойными, волнистыми, пунктирными и т.д.

НОВАЯ КОНЦЕПЦИЯ И ГРАФИКА В СУЩЕСТВУЮЩЕМ ПРОГРАММИРОВАНИИ

Человек всегда стремится найти графический образ любым своим действиям. Известно огромное число графических способов записи программ в виде графов, нагруженных по вершинам: блок-схемы, UML, Workflow, SDL, ДРАКОН, Google BLOCKLY и др. (см. рис. 20а, 21, 22 (слева)). Для таких

графических программ всегда на заключительных этапах используются традиционные языки программирования, которые вручную (не автоматически) преобразуют графическую запись программы в обычную текстовую запись в этом языке.

R-схемы — это графы, нагруженные по дугам (рис. 20, *б*, *в*, 21, 22 (справа)), которые имеют количественные и качественные преимущества. Они *выполнимы*(!) на компьютере на всем жизненном цикле (без использования языков-посредников), более просты, универсальны (не содержат сложных профилей), эффективны (более мощны), наглядны и компактны. Для их записи и обработки в компьютере не нужны существующие языки программирования.

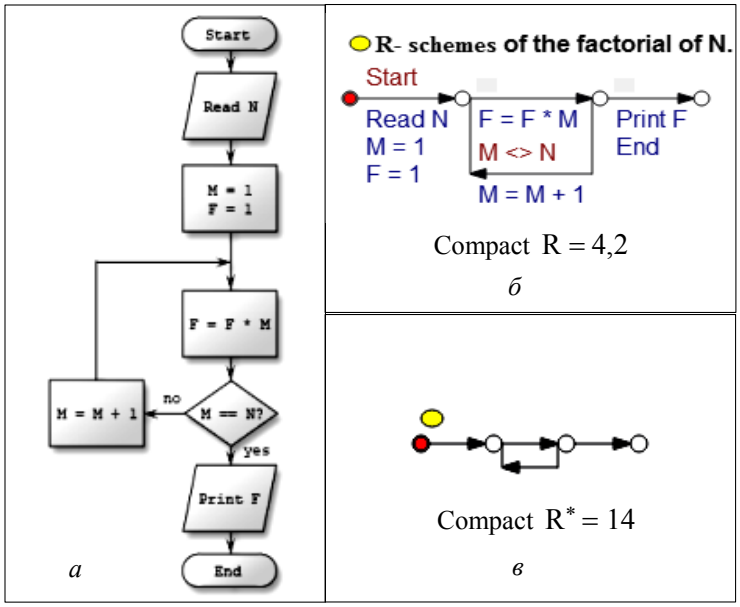

Рис. 20. Пример записи алгоритма вычисления Факториала числа *N* в блок-схемах (*а*) и в RR*-схемах (*б*, *в*)

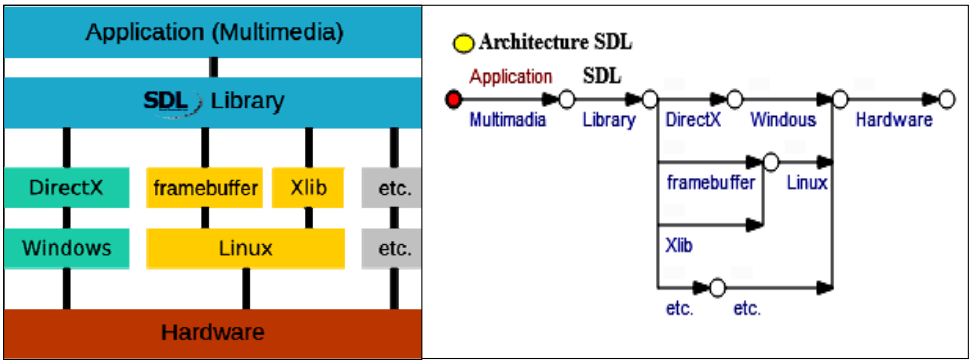

Рис. 21. R-схема Workflow ахитектуры SDL

Таким образом, в новой концепции для всех языков – естественных, математики и программирования – предлагается единая графическая оболочка. Запись программ и их проектов в этой оболочке, а также их трансляция и интерпретация становятся проще. Программа (R-схема) впервые является

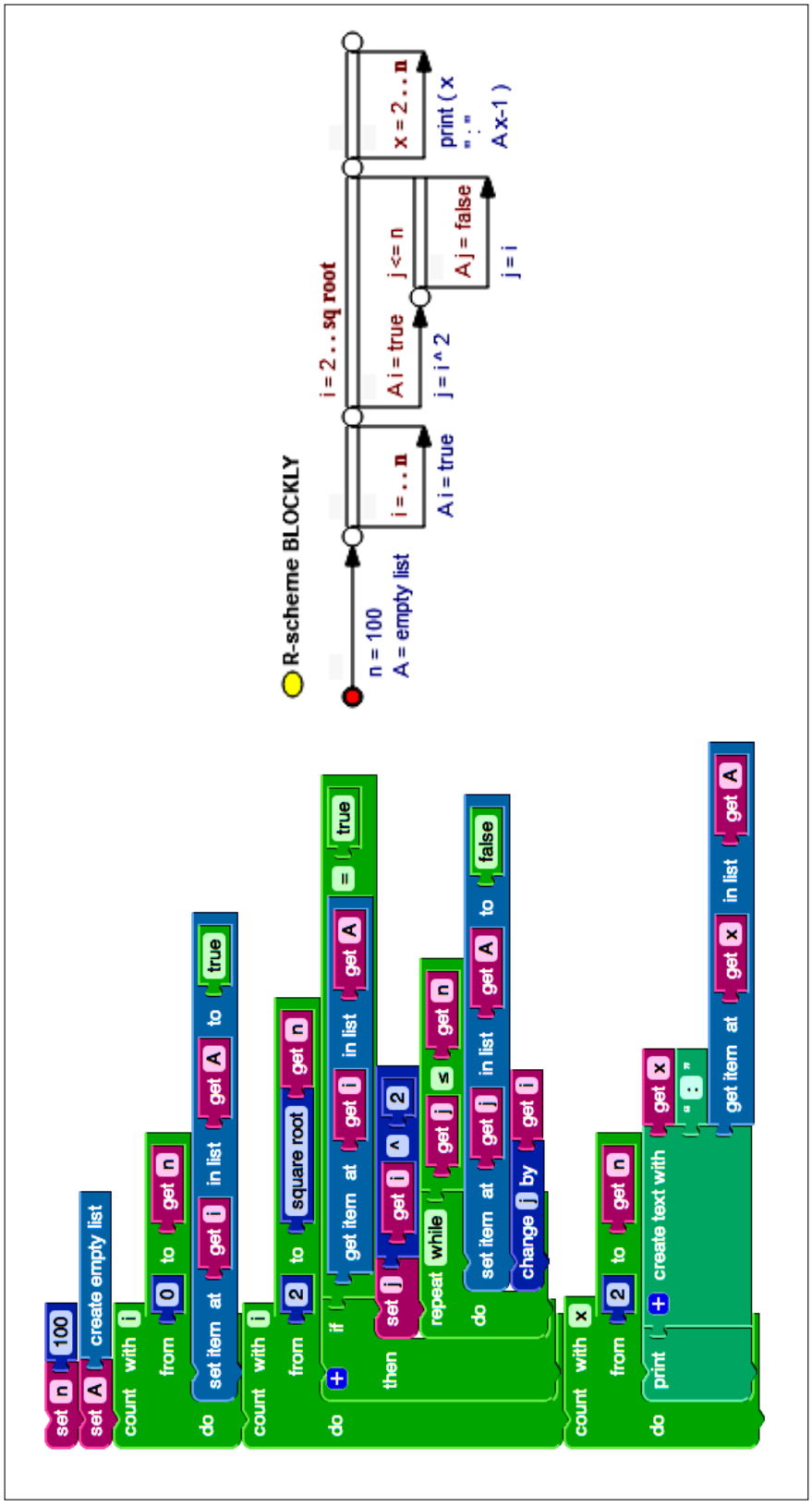

Рис. 22. Пример записи программы в визуальном языке программирования Google Blockly [11] и в R-схеме (компактность R=**3**, R*=**14**)

объектом математики и для человека(!), а не только для компьютера. RR*-схемы идеальны для *обратной* (такого нет в традиционном программировании) трансляции программ на любых языках в новую единую графическую оболочку как эталон. В результате программа из «вещи в себе», понятной только автору (не всегда доступному), становится прозрачной для развития другими, а весь процесс программирования становится простым, эволюционным с сохранением и накоплением опыта в унифицированной графической библиотеке. Поэтому в новой концепции программирования, как и в математике, нет необходимости строить постоянно новые языки и среды программирования, а есть возможность неограниченного совершенствования графической знаковой системы обозначений и обработки информации с сохранением ранее накопленного опыта программирования.

БЕЗОШИБОЧНЫЙ И ДОКАЗАТЕЛЬНЫЙ СТИЛЬ ПРОГРАММИРОВАНИЯ В НОВОЙ КОНЦЕПЦИИ

Традиционная концепция проектирования алгоритма (программы) на компьютере основана на построении вычислительной схемы решения исходной задачи с использованием уже готовых библиотек и с помощью около десяти фиксированных, текстовых, машинно-ориентированных и неестественных для человека, строго не определенных операторов (команд) типа **if, for, while** и т.д. Исходная задача преобразуется под возможность записи алгоритма (программы) в этих операторах. В результате сложного многоступенчатого процесса проектирования алгоритмов и программ исходная задача преобразуется «до неузнаваемости», в нее вводятся новые элементы, меняется связь между ее элементами и мотивацией принимаемых решений, теряется понимание, почему так сделано и как это работает. Для упрощения этого процесса придумывают многочисленные языки, методы и среды, что еще больше запутывает и усложняет процесс программирования. Существующие сложные большие программы, записанные в традиционной концепции программирования, постоянно отлаживаются, поэтому процесс программирования сложный и дорогой.

Новая концепция проектирования основана на построении графической логической R-схемы исходной задачи в естественном языке, терминах и словах формулировки постановки задачи. Для этого используется метод «step by step from logic». В процессе такого построения выявляются и уточняются все основные неточности в постановке задачи. В результате логическая R-схема исходной задачи становится одинаково понятной исполнителю и заказчику и *утверждается* (*подписывается*) ими. Такая R-схема изначально может быть очень простой и недетерминированной (рис. 23).

После этого построение проекта алгоритма и программы осуществляется формальными методами математического вывода в ГСП. Графическая среда программирования следит и направляет человека таким образом, чтобы формальные преобразования исключали из исходной логической R-схемы все неоднозначности понимания и превращали ее в язык *естественный для человека* (как правило, в R-графический язык простейшей математики) и далее автоматически без человека – в язык компьютера. В результате

последовательность преобразований исходной R-схемы автоматически запоминается в ГСП и является формальным доказательством правильности программы и процесса ее разработки. Автоматически получаемая в ГСП документация отличается от традиционной, так как сохраняет мотивацию (историю) принимаемых решений, более компактна, самодокументирована и содержит абстрактные R*-схемы, которые формально *оптимизируются* (преобразуются) в ГСП по различным критериям — времени, памяти и т.д.

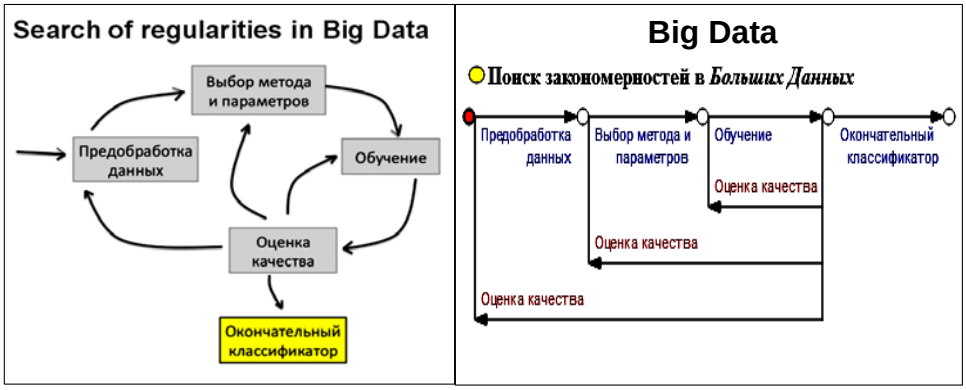

Рис. 23. Пример начала решения в новой концепции сложной проблемы «Поиска закономерностей в BigData»

Естественный язык постоянно совершенствуется и приближается к эффективному профессиональному языку соответствующего коллектива разработчиков проекта графических программ. Язык R-схем такого описания и совершенствования – **один (**см. рис. 2, 3**)** на всем его жизненном цикле и для компьютера, и для заказчика, и для исполнителей проекта, и для его пользователей. Графический и наглядный язык, используемый в основном в процессе математического вывода и во многом автоматически, при минимальных условиях возникновения ошибок (описок), в среде программистов называется безошибочным, а весь процесс, стиль такого программирования доказательным в отличие от программирования в традиционных текстовых языках.

РЕАЛИЗАЦИЯ ГРАФИЧЕСКОЙ СРЕДЫ ПРОГРАММИРОВАНИЯ

В настоящее время *реализована* (лабораторный вариант) ГСП логическими R-схемами, которая включает в себя систему ввода логических R-схем и любых текстов на их дугах на любом языке, формирование и запоминание дерева проекта, универсальный графический редактор, преобразователь R-схем в R* и обратно, компилятор RR* в C++ и т.д. Компилятор RR* в C++ сделан простейшим для демонстрации возможности очень простого преобразования графических программ в любую формальную текстовую структуру или в язык компьютера. Все приведенные примеры реализованы в ГСП. Графическая среда преобразования разработана на C++ как плагин ГСП к *Qt Creator u* состоит из пяти областей, которые делят и формируют экран монитора.

Основную (большую, 90%) центральную часть монитора занимает первая область — рабочее поле (РП), на котором осуществляется разработка всех R-схем проекта.

В оставшихся 10%, сверху в трех строках располагаются три области: меню формирования архитектуры среды, панель инструментов из 14 графических иконок и панель открытых (неограниченное число) имен РП R-схем.

Последняя пятая область занимает слева узкую (менее 5%) полосу экрана монитора для хранения *Дерева* R-схем конкретного проекта. Графическая среда реализована по принципу А. Эйнштейна «Любую работу делай так просто, как это возможно, но не проще этого». Она *намного проще* реализации современных сред и трансляторов с традиционных языков программирования. Например, транслятор графических R-схем *гораздо проще* и *меньше* (ориентировочно в 30+ раз), чем транслятор существующих языков программирования и имеет *более простую и эффективную* инфраструктуру организации вычислений на компьютере. Получаемые компьютерные программы более эффективны по времени выполнения и занимаемой памяти. Главное достоинство реализации новой концепции состоит в том, что сохранился доступ к специализированным инструментам и библиотекам программ традиционных языков программирования.

Реализованная графическая среда обобщила существующий опыт программирования графами, доказывает правильность и эффективность новых принципов программирования и является достаточно полной (по нашим оценкам на 80%) для построения на ее основе коммерческой версии ГСП.

Мнения независимых экспертов и первых пользователей сводятся к такому. Сильная полиглот-концепция программирования проще существующей и может быть использована многими пользователями. Этот новый математический инструмент эффективен для анализа, синтеза и обучения программированию; он увеличивает долголетие и эволюционное использование программ применительно к условиям их эксплуатации; имеет повышенную защищенность от ошибок и высокую отлаженность готовых графических программ. В течении двух недель рисования R-схем в новой среде программирования решается задача, которую без них решить невозможно. Новая концепция — это полезный и продуктивный инструмент не только в программировании, но и везде, где надо найти решение любой логически сложной (запутанной) проблемы. По сравнению с известными Flow-chart, UML. R-схемы имеют ощутимые преимущества. Они выполнимы на компьютере на всем жизненном цикле любых работ, более наглядны, компактны и не загромождены лишними деталями (фигурами). Подходят для визуализации нелинейных алгоритмов (например, классов в C++) или структур данных.

ЗАКЛЮЧЕНИЕ

Предлагаемая новая математическая концепция (культура) программирования отличается простотой, наглядностью, компактностью и возможностью перехода на нее для всех пользователей, а не только для программистов, позволяет включить в программирование доказательный и безошибочный стиль работы, развитые и отработанные веками математические методы

анализа и синтеза. Новая концепция имеет на 1–2 порядка большую компактность записи программ и простоту быстрого ввода информации, особенно через использование сенсорных экранов, что позволяет эффективно применять ее и в малых компьютерных формах (планшетах, Айфонах).

В связи с простотой и наглядностью предложенная концепция может быть включена в систему обязательного начального обучения программированию. Для профессионального программирования эффект от применения новых предложений тем больше, чем логически сложнее и запутаннее решаемая задача.

Главную роль в R-схемах играют ***дуги*** между вершинами. Количество дуг, их направление, 3D-ориентация между различными вершинами и R-схемами не имеют ограничений. Вершина может запоминать количество обращений к ней и иметь порог чувствительности в своей работе. Новая концепция упрощает процесс современного программирования, достаточно обоснована для новых эффективных применений и реализаций новой гибкой, архитектуры аппаратуры компьтеров, которая напрямую управляется выполняемой программой. Это позволит по новому решать проблемы искусственного интеллекта, кибербезопасности, Big Data и суперсложных программ нового поколения.

ЛИТЕРАТУРА

1. *Глушков В.М.* Технология программирования и проблемы ее автоматизации / В.М. Глушков, И.В. Вельбицкий // УСИМ. — № 6. — 1976. — С. 75–93.
2. *Вельбицкий И.В.* Технология программирования / И.В. Вельбицкий // Техника. — 1984. — 279 с.
3. *Сергеев В.Г.* Главный конструктор систем управления ракет и космических комплексов / В.Г. Сергеев. — Х., 2014. — 448 с.
4. *Dijkstra E.* Letters to the editor: go to statement considered harmful / E. Dijkstra // Communications of the ACM. — 1968. — P. 147–148.
5. *Information* technology, Programme constructs and convention for their Representation // Standard ISO/IEC 8631. — 1989.
6. *McHenry W.K.* Technology: A soviet visual programming / W.K. McHenry // Journal of Visual Languages and Computing. —**1**, N 2. — 1990.
7. *Velbitskiy I.V.* Graphical Programming and Program Correctness Proof / I.V. Velbitskiy // IEEE: 10.1109/ CSIT-13.6710368. — 2013. — P. 85–89.
8. *Velbitsky I.* Programming without Programming Languages (New Graphic Poliglot Concept) / I. Velbitsky // Application and Theory of Computer Technology (the British kingdom). —**2**, N.2. — 2017. — P. 26–41. — Available at:
9. http://www.archyworld.com/journals/index.php/atct/article/view/49
10. Valvachev A.N. Programming in Delphi / A.N. Valvachev, K.A. Syrkov, D.A. Syrkov. — 2005. — Available at: http://www.rsdn.ru/article/Delphi/
11. Prata Stephen. C++ Primer Plus, Fifth Edition, SAMS Indiana, 46240 USA, 1184 p., ISBN 5-8459-1 1 27-3 Moscow. 2007.
12. *Cade Metz.* Google Blockly Lets You Hack With No Keyboard. — Available at: https://en.wikipedia.org/wiki/Blockly

Поступила 07.07.2017