Смольянинов Александр Владимирович

Возможности среды визуального программирования BLS для использования в учебном процессе

Сегодня все больше становится людей, которые не только используют установленные на компьютерах программы, но и вовлечены в той или иной степени в процесс создания программ. В сложившейся ситуации необходимы инструментальные средства, позволяющие быстро и без особых усилий создавать необходимые программы. Такие средства должны быть ориентированы на массовое применение, при этом необходимо обеспечить эффективное обучение методам и способам разработки программ.

Для определения, чему учить, как учить и с помощью каких средств, целесообразно рассмотреть процесс разработки программ, выделить принципиальные положения, могущие составить предмет изучения, и определить желаемые свойства инструментальных и обучающих средств.

ПРОЦЕСС РАЗРАБОТКИ ПРОГРАММ И ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА

Разработку программ (программирование) нужно, прежде всего, должным образом организовать, а затем сделать методы разработки достоянием как можно большего числа заинтересованных лиц.

Организация разработки должна начинаться с ответа на вопрос, какая программа нужна. Современный ответ постулирует, что нужна программа, удовлетворяющая первоочередным критериям качества. Такая программа должна полностью решать поставленную задачу, обеспечивать наглядное взаимодействие с пользователем, иметь четкую и наглядную структуру и быть разработанной систематическим способом.

Ключевым этапом в разработке качественных программ является построение хорошо структурированного алгоритма. Четкая и наглядная структура алгоритма, а следовательно, и программы порождается использованием определенных конструкций для представления частей алгоритма и строгих правил их компоновки в рамках систематического способа разработки. Набор конструкций представляет структурный и модульный подходы к программированию, а в качестве систематического способа разработки выступает нисходящее проектирование алгоритмов и программ. Точное следование этим методам гарантирует и построение и надежность алгоритма (программы). Для такого алгоритма сопоставление его конструкциям фрагментов программы, а в итоге и получение самой программы всегда

возможно и является чисто технической задачей - кодированием конструкций алгоритма на выбранном языке по известным правилам.

Основная трудность алгоритмизации заключается в том, что составление алгоритма - это творческий процесс, который полностью невозможно автоматизировать. В лучшем случае можно лишь создать условия для разработчика, в которых этот процесс будет стимулироваться и реализовываться наиболее эффективно.

В настоящее время понято и принято в качестве руководства к действию, что необходимым (а иногда и достаточным) условием этого является наглядность представления алгоритма и программы на всех этапах их существования. Наглядность должна быть такой, чтобы разработчик ясно видел из каких элементов составлены алгоритм и программа, каким сущностям решаемой задачи они соответствуют, какова логика взаимодействия элементов, как алгоритм и программа соотносятся друг с другом.

Подводя итог, можно утверждать, что учить целесообразно методам, способам и приемам структурного и модульного программирования, акцентируя методику на разработку алгоритмов и последующее получение на их основе программ на языке программирования. При этом основу разработки должны составлять наглядные представления. Инструментальное средство программирования, реализующее такую методологию, может и должно использоваться как средство обучения. В такой среде обучение должно осуществляться путем конструктивной деятельности по разработке алгоритмов и программ в условиях, которые обязывают использовать изучаемые конструкции и правила построения.

ВИЗУАЛЬНОЕ ПРОГРАММИРОВАНИЕ

Стремление сделать разработку программ наглядной и, за счет этого, эффективной привело к формированию специального направления, получившего название "визуальное программирование". Этот термин обозначает совокупность методов и средств создания программ путем манипулирования на экране компьютера наглядными образами сущностей решаемой задачи, данных и процессов их обработки.

Визуальное программирование поддерживает различные виды деятельности по созданию программ - от проектирования до сборочного производства. Сводя разработку к компоновке наглядных представлений, которые в итоге задают программные компоненты, визуальное программирование позволяет создавать прикладные программы практически "без программирования".

Перспективность визуального программирования как технологии определяется его инструментальными и когнитивными свойствами. Оно позволяет использовать в творческом процессе разработки программ неязыковое образное мышление автора, раскрепощаемое в среде наглядных и узнаваемых представлений. Эта особенность обуславливает целесообразность применения визуального программирования в учебном процессе как в качестве средства обучения, так и в качестве предмета изучения [1].

Наибольшее развитие и широкое распространение получили инструментальные средства, позволяющие визуально программировать интерфейс с пользователем или визуально собирать программу из специальным образом подготовленных компонент. Визуальные средства разработки абстрактных алгоритмов развиты еще недостаточно, хотя и представляют большой интерес как средства для обучения алгоритмизации и программированию и как средства разработки программ конечными пользователями.

РАБОТА В СРЕДЕ BLS

BLS как учебная среда

Среда выступает в двух ипостасях. С одной стороны, это инструментальное средство для разработки алгоритмов и программ на языке Паскаль (стандарт ANSI). С другой стороны, она является учебной средой, которая предлагает современные рациональные и надежные способы и приемы создания качественных алгоритмов и программ. Работа в среде позволяет легко освоить методы структурного и модульного программирования и оценить преимущества визуальных методов разработки.

Среда обеспечивает реализацию основных этапов создания программы: проектирования, алгоритмизации и получения текста на языке Паскаль. Она может использоваться для изучения способов и приемов построения алгоритмов решения задач любого происхождения. Для успешной работы в этом направлении достаточно элементарных навыков логического мышления. В среде можно квалифицированно разрабатывать программы средней сложности на языке Паскаль. Для этого необходимо знание языка. Однако в среде можно эффективно осуществлять начальное обучение программированию, постепенно вводя необходимые сведения о языке для обеспечения выполнения алгоритма. Среда предоставляет широкие возможности для создания и реализации разнообразных методик обучения алгоритмизации и программированию.

Разработка алгоритма: основные действия

Основой для создания программы в среде является алгоритм, который представляется блок-схемой (схемой). Поэтому разработка алгоритма сводится к построению схемы, которая автоматически сохраняется в файле.

В режиме создания схемы среда автоматически строит пустую (с точки зрения решаемой задачи) схему, содержащую только символы "начало" и "конец". Дальнейшая разработка выполняется в режиме структурного программирования. Это означает, что на каждом шаге разработки может быть построена только "простая" схема.

Смысл слова "простая" применительно к схеме состоит в том, что такая схема имеет только один вход и только один выход, который должен быть обязательно реализован. Такие требования не накладывают ограничений на внутреннюю сложность схемы и обеспечивают ее практическую полезность. Схема формируется только из основных конструкций структурного программирования, задающих управление действиями программы. Это конструкции следования, решения и повторения. Для практических целей используется полный набор их вариантов. Конструкции, которые можно использовать для построения схемы, сведены в вертикальное меню в правой части экрана (см. верхний и нижний рисунки на 3-ей странице обложки).

Для построения схемы используется только одно проблемное действие - суперпозиция конструкций, которое обеспечивает включение одной конструкции в состав другой в качестве элемента. Включение реализуется тремя действиями разработчика схемы: выбором места в схеме, в которое надо включить, выбором включаемой конструкции в меню и щелчком левой клавиши мыши (или нажатием клавиши INS), выполняющим размещение конструкции в схеме.

Выбор места осуществляется проблемным курсором, который может указывать либо конструкцию схемы (тогда он называется "блок"), либо место на линии потока передачи управления (тогда он называется "разрыв"). Виды курсоров и

их названия представлены на рисунках 3-ей страницы обложки. Основные действия по построению схемы могут быть заданы либо с помощью мыши, либо с помощью клавиш.

Профессиональный уровень разработки алгоритма предполагает указание для каждого действия его назначения. Это можно выполнить, снабдив конструкцию схемы пояснением (комментарием). Подготовка комментария осуществляется в специальном окне, которое предоставляет для этого основные возможности традиционных текстовых редакторов. При переходе из этого окна к схеме комментарий автоматически связывается с конструкцией схемы. На верхнем рисунке 3-ей страницы обложки изображена схема, конструкции которой снабжены комментариями.

Любая конструкция схемы может быть удалена. Для этого необходимо выбрать конструкцию проблемным курсором и нажатием клавиши DEL удалить ее. Если конструкция была снабжена комментарием, то она будет удалена из схемы, но не будет потеряна для разработки. Она будет помещена в архив схемы для возможного последующего использования.

Написание текста программы

Подготовка текста программы осуществляется полуавтоматически. На каждом шаге построения алгоритма с его схемой автоматически связывается шаблон программы - ее незаконченный вариант на языке Паскаль (стандарт ANSI), в котором представлены шаблоны для всех использованных конструкций управления. Этот шаблон в общем случае не может быть выполнен.

Если целью разработки является построение алгоритма, то степенью законченности программы можно не интересоваться. Если цель - получение программы, то в шаблон необходимо включить на языке Паскаль те элементы программы, которые не могли быть заданы автоматически как результат манипулирования конструкциями. Формирование текста осуществляется отдельно для каждой конструкции в специальном окне. Участки в шаблоне конструкции, требующие заполнения, заданы вопросительным знаком. Его замена на требуемое содержание осуществляется обычными для текстовых редакторов способами. Результат автоматически включается в шаблон и связывается со схемой (см. верхний рисунок на 3-ей странице обложки).

Текст программы на каждом шаге автоматически запоминается в специальном файле, может быть просмотрен в среде или использован автономно.

Выполнение программы

На каждом шаге разработки среда позволяет выполнить алгоритм (схему). В действительности выполняется программа, сопоставленная схеме. Если программа не закончена (шаблон не заполнен), то выполнение невозможно, и об этом выводится сообщение в поле индикации ошибок. Расположение этого поля показано на нижнем рисунке 3-ей страницы обложки. Если программа может быть выполнена, то она исполняется и по завершении происходит возврат в среду.

Обеспечение обозримости алгоритма: группирование

При разработке количество конструкций в схеме может увеличиться настолько, что схема не сможет полностью поместиться на экране. В результате теряется наглядность изображения алгоритма.

Для сохранения схемы в пределах экрана предоставляется возможность группировки конструкций. Группа представляется в схеме прямоугольником со сторонами, заданными пунктирной линией. Если группа выбрана проблемным курсором, то в пределах этого прямоугольника

изображаются все конструкции, составляющие группу (см. рис.1). При этом другие части схемы могут выйти за пределы экрана. Если курсором выбрана конструкция вне группы, то схема преобразуется, и детальное изображение группы заменяется на новое - на прямоугольник, ограниченный пунктирной линией и содержащий внутри знак разрыва. На рис.1 сверху и снизу по отношению к выделенной курсором группе видны символы группы, содержащие знак разрыва.

Для группировки конструкций в среде реализовано два способа. Первый позволяет объединить в группу конструкции, существующие в схеме. Второй - создавать группу из новых конструкций.

При первом способе выбирается одна из конструкций схемы, которая подлежит включению. Затем в меню выбирается символ "Группа конструкций", и с помощью клавиш со стрелками последовательно друг за другом включаются конструкции, расположенные или ниже выбранной или выше ее. Второй способ состоит в указании на линии потока места для создаваемой группы, создании в этом месте пустой группы и наполнении ее конструкциями из меню или из архивов. На рис.1 изображен также кадр визуальной помощи, показывающий в картинках, как нужно действовать в этом случае. Количество групп и глубина их вложения друг в друга практически не ограничены.

Разработка "сверху-вниз": детализация

Среда обеспечивает нисходящее ("сверху-вниз") построение алгоритмов и программ. Для этого в схему включаются символы "Детализируемый процесс". Каждый такой символ означает, что реализующий этот символ фрагмент алгоритма определен вне изображенной схемы.

Для задания фрагмента, соответствующего такому символу, нужно установить проблемный курсор на символ, а затем либо щелкнуть левой клавишей мыши, либо нажать клавишу Enter. Происходит как бы вход внутрь этого символа и разработчик получает возможность создать фрагменты схемы и программы, реализующие символ. На рис. 2 показана ситуация модернизации детализированного фрагмента, состоящей во включении новой конструкции в разрыв линии потока. Кадр визуальной помощи показывает, как это осуществить.

Детализация выполняется по общим правилам построения схемы и формирования соответствующих программных фрагментов. Количество детализаций и их глубина практически не ограничены.

Разработка "снизу-вверх": использование архивов

Средствами среды можно выполнять восходящее ("снизу-вверх") построение алгоритмов и программ. При этом алгоритм формируется из уже построенных и доступных фрагментов. Простейший вариант восходящей разработки реализуется выбором конструкций из меню и включением их в схему. Более мощный вариант

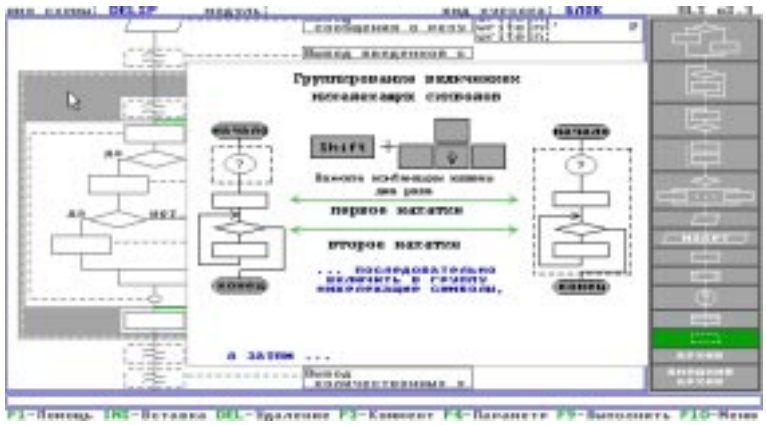

Рисунок 1.

разработки "снизу-вверх" предусматривает использование архивов, в которых хранятся и из которых выбираются требуемые достаточно сложные фрагменты. Можно использовать архив схемы, которая разрабатывается, или архивы других схем (внешние архивы). Вид архива определяется выбором в меню конструкций символа "архив", если нужно использовать архив разрабатываемой схемы, или символа "внешний архив". Доступ к архиву схемы осуществляется непосредственно, а для обращения к внешнему архиву необходимо указать его имя. При просмотре архива выбирается требуемый элемент и он обычным способом включается в схему. Нижний рисунок на 3-ей странице обложки иллюстрирует построение схемы модуля с использованием архива.

Модернизация схемы

Для внесения изменений в схему необходимо выбрать пункт "Модернизация схемы" в меню режимов работы среды и задать имя модернизируемой схемы. Для внесения изменений доступны те же самые действия, которые использовались при разработке алгоритма.

Модульное программирование

Среда позволяет выполнять разработку модульных программ. В качестве модулей выступают процедуры и процедуры-функции языка Паскаль. Среда предоставляет возможность разработать модуль и разместить его вызов в другой программной единице: программе и процедуре. Модуль создается с помощью тех же действий и приемов, что и программа. Отличие имеется только в начале построения. Нужно выбрать режим создания модуля, задать имя модуля и далее разработать модуль. Для задания вызова модуля нужно в вызывающем алгоритме разместить символ "Предопределенный процесс". Нижний рисунок на 3-ей странице обложки демонстрирует разработку модуля с использованием архива.

Визуальное программирование ввода-вывода

Среда предоставляет возможность автоматизированного программирования процессов ввода и вывода числовой и текстовой информации. Разработчик программы должен разместить в алгоритме символы "Ввод-Вывод". Для каждого из них он должен выбрать в меню конструкций символ "Макет ввода-вывода" и перейти в режим построения макета. Это означает, что ему нужно, как в текстовом редакторе, построить образ экрана, включив в него все тексты и числовые значения в том виде и в той последовательности, в которой они должны появляться при исполнении этого элемента алгоритма. Затем нужно поочередно выделить в построенном макете поля, в которые будет производиться либо ввод, либо вывод и связать их с нужными переменными программы (см. рис. 3). В результате будет автоматически построен фрагмент текста

Рисунок 2.

программы, обеспечивающий при ее выполнении заданную форму информации на экране, содержащую вычисленные программой значения.

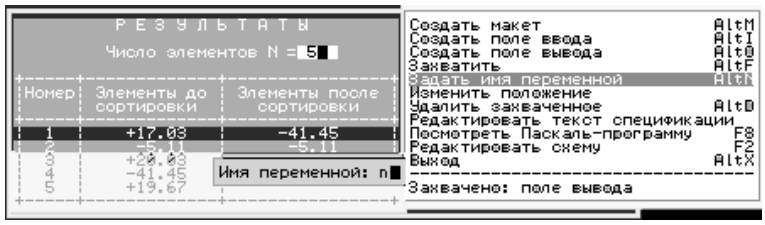

Рисунок 3.

Визуальная помощь

Среда оснащена системой контекстно-зависимой визуальной помощи, которая, используя наглядную форму представления информации, показывает на одном или нескольких кадрах, как нужно выполнить выбранное действие (см. рис. 1 и 2).

Аппаратное и программное обеспечение функционирования среды BLS

IBM-совместимый компьютер (начиная с серии 286) с минимальными характеристиками:
- оперативная память от 512 Кб;
- графический адаптер EGA или VGA с 256 Кб памяти;
- жесткий диск со свободным местом около 1 Мб.

Дополнительные внешние устройства:
- манипулятор типа "мышь", совместимый с IBM Mouse;
- печатающее устройство, имеющее систему управляющих кодов совместимую с EPSON, IBM Graphic Printer или IBM ProPrinter.

Операционная система MS DOS версии 3.0 и выше, а также в MS Windows в оконном и полноэкранном режимах.

Дополнительное программное обеспечение:
- русификатор клавиатуры и экрана в альтернативной кодировке, обеспечивающий графические режимы работы (например, cyrillic.com);
- пакетный компилятор "tpc.exe" и системная библиотека "turbo.tpl" системы Turbo Pascal версии 4.0 и выше для выполнения разрабатываемых программ.

Литература

1. Смольянинов А.В., Калмычков В.А. Визуальное программирование в учебном процессе. - В кн.: Современные технологии обучения. Сб. научн.-метод. тр., Вып. 3.- СПб.: СПбГЭТУ, 1997.- с. 8-19

Смольянинов Александр Владимирович, доцент кафедры математического обеспечения и применения ЭВМ Санкт-Петербургского государственного электротехнического университета, кандидат технических наук.

НАШИ АВТОРЫ