

Министерство образования и науки Российской Федерации
Федеральное агентство по образованию
Ярославский государственный университет им. П. Г. Демидова
Кафедра компьютерной безопасности

**И. В. Лаврентьев
В. А. Никулин
Н. Б. Чаплыгина**

Средства отладки в Visual C++

Методические указания

*Рекомендовано
Научно-методическим советом университета для студентов,
обучающихся по специальностям Математика
и Прикладная математика и информатика*

Ярославль 2010

УДК 519.72
ББК 3 973.2–018.2я73
Л 13

*Рекомендовано
Редакционно-издательским советом университета
в качестве учебного издания. План 2009/10 года*

Рецензент
кафедра компьютерной безопасности Ярославского
государственного университета им. П. Г. Демидова

Лаврентьев, И. В. Средства отладки в Visual C++ :
Л 13 метод. указания / И. В. Лаврентьев, В. А. Никулин,
Н. Б. Чаплыгина; Яросл. гос. ун-т им. П. Г. Демидова. –
Ярославль : ЯрГУ, 2010. – 51 с.

Методические указания разработаны для ознакомления со специальными средствами, помогающими правильно организовать и ускорить процесс отладки, а также освоить режим отладки приложений в интегрированной среде MS Visual C++.

Предназначены для студентов, обучающихся по специальностям 010101.65 Математика, 010501.65 Прикладная математика и информатика (дисциплины «Информатика», «Практикум по ЭВМ», блок ЕН), очной формы обучения.

УДК 519.72
ББК 3 973.2–018.2я73

© Ярославский
государственный университет
им. П. Г. Демидова, 2010

Под отладкой понимается процесс выявления и исправления ошибок в программе. Каждый программист, начинающий или опытный, так или иначе делает ошибки. Практически даже самая простая программа нуждается в исправлениях, прежде чем начнет выполняться так, как задумал ее создатель. Этап отладки наиболее трудоемкий в жизненном цикле программы и требует немалых интеллектуальных затрат, на него уходит не менее половины времени создания программы. Он заключается в устранении логических ошибок, обнаруженных во время тестирования работающей программы, т. е. после устранения ошибок, не позволяющих создать исполняемый модуль. Во время отладки продолжается процесс изучения задачи и уточняется алгоритм ее решения, и зачастую процесс отладки в корне меняет взгляд на выбор алгоритма решения и тогда программа переписывается заново.

При создании программы для решения некоторой поставленной задачи программисту следует предварительно разработать список тестовых примеров для обнаружения возможных ошибок в работе программы. Таким образом, процесс тестирования и процесс отладки программы тесно связаны, т. е. в процессе отладки непременно используется этап тестирования. После обнаружения ошибки ее устранение производится в два этапа: определяется возможная причина и локализуется место ошибки, а затем ошибка исправляется. И в этот момент можно использовать другие специально созданные тестовые примеры, которые направлены не на проверку работы всей программы, а нацелены на поиск данной ошибки.

Поиск обнаруженной ошибки, установление ее причин могут быть осуществлены аналитическими методами, порой без использования программных средств, путем тщательного анализа хода выполнения программы. Как вспомогательное средство в этом случае может быть использовано направленное тестирование, которое может дать дополнительную информацию об ошибке. Аналитический метод наиболее предпочтителен, так как требует четкого представления о логике выполнения программы и может выявить другие ошибки, которые себя пока не проявили. Если же этот метод не дает результатов, то для локализации та-

ких «трудных» ошибок прибегают к методу «грубой силы», инспектированию программы с помощью автоматических средств. Зачастую программисты сразу прибегают к таким средствам, минуя аналитический этап, так как применение автоматических средств не требует значительного внимания и такого умственного напряжения, как использование аналитического метода. Однако применение автоматических средств не освобождает совсем от анализа конкретных ситуаций, возникающих в ходе выполнения программы. Использование автоматических отладочных средств без анализа всего хода исполнения алгоритма, взаимосвязей его частей таит в себе опасность получения новой ошибки, может быть, совсем иной природы, взамен только что исправленной.

Причины одних логических ошибок, обнаруженных в ходе выполнения, для программистов, имеющих некоторый опыт, выясняются довольно быстро, для поиска других требуется существенная работа, и в этом может помочь среда программирования Microsoft Visual C++, предоставляя удобные автоматические инструменты для поиска ошибок. К таковым автоматическим отладочным средствам относятся получение последовательности выполненных операторов программы или ее фрагмента, установление контрольных точек останова, предоставление возможности инспектирования значений переменных или областей памяти в процессе выполнения программы. Предлагаемые методические указания знакомят с составом и архитектурой отладочных средств в интегрированной среде разработки приложений Microsoft Visual C++ Express Edition, а также рассмотрены шаги построения программ: от исходного кода до исполняемого файла. Вся терминология и обозначения могут быть в точности перенесены и на среду разработки MS Visual Studio .NET.

I. Этапы создания программ на C/C++

Каждая программа на языке C++ есть последовательность препроцессорных директив, описаний и определений глобальных функций, переменных, пользовательских типов. Препроцессорные директивы управляют преобразованием текста программы до ее компиляции. Определения вводят функции и объекты. Объекты необходимы для представления в программе обрабатываемых данных. Функции содержат потенциально возможные действия программы. Описания уведомляют компилятор о свойствах и именах тех объектов и функций, которые определены в других частях программы (например, ниже по ее тексту или в другом файле).

Задача препроцессора – преобразование программы до её компиляции. Правила препроцессорной обработки устанавливает сам программист с помощью директив `#define` и `#include`, которые предписывают заменить некоторое имя, встречающееся в тексте программы, на постановочную строку замены и вставить фрагмент текста из указанного файла соответственно. Список заголовочных файлов, которые содержат интерфейсы уже созданных объектов: констант, макросов, функций, используемых для подключения директивой `#include`, есть в директории по умолчанию

C:\Program Files\Microsoft Visual Studio .NET 2003\Vc7\include.

При просмотре содержимого этой папки можно увидеть имена заголовочных файлов: с расширением `.h` и без него, что позволит правильно указать имя в директиве `#include`. В свою очередь, каждый такой файл можно открыть и познакомиться с его содержанием, узнать, описания каких функций в нем содержатся.

Препроцессор сканирует исходный текст программы в поисках строк, начинающихся символом `#`. Такие строки воспринимаются препроцессором как команды, которые определяют действия по преобразованию текста программы.

После выполнения препроцессорной обработки в программе не остается ни одной препроцессорной директивы. На этом этапе программа представляет собой набор описаний и определений имен и функций.