Московский Государственный Универстит имени М. В. Ломоносова

Реферат

на тему: «Интегрированные среды разработки ПО»

Выполнил: студент гр. 420
Немешаева Алиса

Москва 2020

Содержание

1 История появления интегрированных сред разработки ПО

Интегрированная среда разработки, ИСР (англ. IDE, Integrated Development Environment или Integrated Debugging Environment) — объединение программных средств, используемое разработчиками для создания программного обеспечения (ПО).
Классическая среда разработки включает в себя:

- текстовый редактор;
- компилятор и / или интерпретатор;
- средства автоматизации сборки;
- отладчик.

Иногда IDE может содержать также средства для интеграции с системами управления версиями и различные инструменты для упрощения конструирования графического интерфейса для конечного пользователя. Современные среды разработки часто также включают браузер классов, инспектор объектов и диаграмму иерархии классов — для упрощения систематизации при объектно-ориентированной разработке ПО. Хотя и существуют IDE, предназначенные для нескольких языков программирования — такие, как Eclipse, NetBeans, Embarcadero RAD Studio, Qt Creator или Microsoft Visual Studio, но чаще всего IDE предназначается для одного определённого языка программирования - как, например, Visual Basic, PureBasic, Delphi, Dev-C++.

Первые IDE были созданы для работы через консоль или терминал, которые сами по себе только недавно вошли в употребление на тот момент времени: до того программы создавались на бумаге, вводились в машину с помощью предварительно подготовленных бумажных носителей (перфокарт, перфолент) и так далее.

Dartmouth BASIC был первым языком, который был создан с IDE, и был также первым языком, что был разработан для использования в консоли или терминале. Эта IDE (часть Dartmouth Time Sharing System) управлялась при помощи команд, поэтому существенно отличалась от более поздних, управляемых с помощью меню и горячих клавиш, и тем более графических IDE, распространённых в XXI веке. Однако она позволяла править исходный код, управлять файлами, компилировать, отлаживать и выполнять программы способом, принципиально подобным современным IDE.

Maestro I — продукт от Softlab Munich, был первой в мире интегрированной средой разработки для программного обеспечения в 1975 г. и, возможно, мировым лидером в этой рыночной нише в течение 1970-х и 1980-х годов. Он был установлен у 22000 программистов во всем мире. До 1989 года 6000 копий было установлено в Федеративной Республике Германия. Ныне Maestro I принадлежит истории и может быть найден разве что в Музее Информационной технологии в Арлингтоне.

Одной из первых IDE с возможностью подключения плагинов была Softbench.

Таблица 1.1 — Несколько популярных IDE по годам их появления

Год	Название IDE	Разработчик
1976	Emacs	David A. Moon, Guy L. Steele Jr.
1991	Vim	Bram Moolenaar
1997	Visual Studio	Microsoft
1997	NetBeans	Apache Software Foundation, Oracle, Sun Microsystems
1999	KDevelop	KDE
2001	Eclipse	IBM, Eclipse Foundation
2001	IntelliJ IDEA	JetBrains
2003	XCode	Apple Inc.
2005	Code::Blocks	The Code::Blocks team
2005	Oracle Solaris Studio	Oracle Corporation
2006	CodeLite	Eran Ifrah
2007	Qt Creator	Qt Project
2007	Komodo	ActiveState
2008	Sublime Text	Jon Skinner
2009	PhpStorm	JetBrains
2009	Spyder	Pierre Raybaut
2013	Xamarin	Microsoft
2014	Atom	GitHub Inc.
2014	Project Jupyter	Project Jupyter
2015	Visual Studio Code	Microsoft

2 Обзор современных сред разработки

Интегрированная среда разработки (IDE) - это своего рода расширенный редактор для одного или нескольких языков программирования со встроенным компилятором, окном редактирования и утилитами отслеживания или отладки ошибок, которые упрощают разработку программы. IDE облатает кроме того ещё многими другими функциями, например:

■ Пользователь может просматривать файлы через IDE, чтобы найти тот, который ему нужен;

■ Доступна «справка», с помощью которой пользователь может быстро узнать, какие аргументы требуются конкретному методу или что может означать ошибка;

■ Пользователь может видеть измененные значения переменных, объектов и функций в режиме реального времени;

IDE - это программы для ввода исходного кода. Обычно это среды редактирования с инструментами, помогающими программистам быстро и эффективно писать исходный код. Например пользователь может создавать веб-приложения, управляемые PHP, используя комбинацию Eclipse и PHP Eclipse.

Основные функции IDE обычно включают:

■ Анализ кода: это возможность среды IDE знать ключевые слова и функции языка. IDE может использовать эту информацию, чтобы исполнять действия, как:

- □ выделение типографические ошибок;
- □ вывод списка всех доступных функций на основе соответствующей ситуации;
- □ предположение определения функции;
- □ подсветка различных элементов кода разными цветами для различных ключевых слов и функций.

■ Управление ресурсами: при создании приложений языки часто полагаются на то, что определенные ресурсы, такие как библиотека или файлы заголовков, находятся в определенном каталоге. IDE управляют этими ресурсами. IDE знает необходимые пути к библиотекам и

заголовочным файлам, поэтому ошибки можно обнаружить на этапе разработки, на этапе компиляции или сборки.

■ Инструменты отладки. В среде IDE пользователю предоставляется возможность тщательно протестировать свое приложение перед выпуском. IDE может предоставлять значения переменных в определенные моменты, подключаться к разным репозиториям данных или принимать разные параметры времени выполнения.

■ Компиляция и сборка: для языка, которому требуется этап компиляции или сборки, IDE переводит код с языка высокого уровня в объектный код целевой платформы. Требования к этой функции существенно различаются от языка к языку.

Таким образом, традиционно IDE специализируется на одном языке или наборе подобных языков. Некоторые известные IDE и их языки включают: JBuilder для Java; Пакет Metrowerks CodeWarrior для Java, C и C ++; и Microsoft Visual Studio для семейства языков Visual Basic и C. Далее будет проведено сравнение некоторых существующих IDE по их характеристикам.

2.1 IntelliJ Idea

IntelliJ IDEA - это интегрированная среда разработки (IDE), написанная на Java для разработки компьютерного программного обеспечения. Он разработана компанией JetBrains (ранее известной как IntelliJ).

Эта IDE предоставляет определенные функции, такие как завершение кода путем анализа контекста, навигация по коду, которая позволяет напрямую переходить к классу или объявлению в коде, рефакторинг кода, отладку кода, линтинг и варианты исправления несоответствий с помощью предложений. IDE обеспечивает интеграцию с инструментами сборки / упаковки, такими как grunt, bower, gradle и SBT. Он поддерживает системы контроля версий, такие как Git, Mercurial, Perforce и SVN. К базам данных, таким как Microsoft SQL Server, Oracle, PostgreSQL, SQLite и MySQL, можно получить доступ непосредственно из среды IDE в версии Ultimate через встроенную версию DataGrip.

IntelliJ поддерживает плагины, с помощью которых можно добавлять в IDE дополнительные функции. Плагины можно загружать и устанавливать либо с веб-сайта репозитория плагинов IntelliJ, либо с помощью встроенной в IDE функции поиска и установки плагинов. Каждая редакция имеет отдельные репозитории плагинов, причем в редакциях Community и

Ultimate насчитывается более 3000 плагинов каждая по состоянию на 2019 год.

2.2 Eclipse

Eclipse написана на языке Java. Eclipse - это многоязычная среда разработки программного обеспечения, состоящая из интегрированной среды разработки (IDE) и расширяемой системы подключаемых модулей. Eclipse SDK (Software Development Kit) включает Eclipse Java Development Tools (JDT), предлагающий IDE со встроенным инкрементным компилятором Java и полную модель исходных файлов Java. Это позволяет применять передовые методы рефакторинга и анализа кода. IDE также использует рабочее пространство. Eclipse реализует виджеты с помощью набора инструментов виджетов для Java, называемого SWT, в отличие от большинства приложений Java, которые используют стандартный набор инструментов абстрактного окна Java (AWT) или Swing. Пользовательский интерфейс Eclipse также использует промежуточный уровень графического интерфейса, называемый JFace, который упрощает создание приложений на основе SWT.

Eclipse - это расширяемая платформа для создания IDE. Она предоставляет ядро служб для управления набором инструментов, работающих вместе для поддержки задач программирования. Создатели инструментов вносят свой вклад в платформу Eclipse, упаковывая свои инструменты в подключаемые компоненты, называемые подключаемыми модулями Eclipse, которые соответствуют контракту подключаемых модулей Eclipse.

2.3 Qt Creator

Qt Creator была разработана Qt Development Frameworks на C++. Это кроссплатформенная IDE, работающая только для языка C++. Она включает в себя визуальный отладчик, интегрированный графический интерфейс и конструктор форм. Возможности IDE включают выделение синтаксиса, автодополнение, редактор для написания правильно отформатированного кода, конструктор пользовательского интерфейса и так далее. Он предоставляет плагин отладчика, который действует как интерфейс между ядром Qt Creator и внешним собственным отладчиком для отладки языка C++. Qt Creator обеспечивает поддержку для создания и запуска приложений Qt для сред персональных компьютеров (Windows, Linux, FreeBSD и Mac OS) и мобильных устройств (Symbian, Maemo и MeeGo).

2.4 PyCharm

PyCharm - это интегрированная среда разработки, используемая в компьютерном программировании, особенно для языка Python. Она также разработана чешской компанией

JetBrains. Она обеспечивает анализ кода, графический отладчик, интегрированный тестер модулей, интеграцию с системами контроля версий (VCSes) и поддерживает веб-разработку с помощью Django, а также анализ данных с помощью Anaconda.

PyCharm является кросс-платформенной IDE, с версиями для Windows, macOS и Linux. Community Edition выпускается под лицензией Apache License, а также существует Professional Edition с дополнительными функциями, выпущенная под частной лицензией.

2.5 Jupyter Notebook

Jupyter Notebook (ранее IPython Notebooks) - это интерактивная вычислительная среда в виде веб-приложения для создания документов Jupyter notebook. Термин «notebook» может в разговорной речи относиться к множеству различных сущностей, в основном к веб-приложению Jupyter Notebook, веб-серверу Jupyter Python или формату документа Jupyter в зависимости от контекста. Документ Jupyter Notebook - это документ JSON, соответствующий схеме с контролем версий и содержащий упорядоченный список ячеек ввода / вывода, который может содержать код, текст (с использованием языка разметки Markdown), математику, графики и мультимедийные данные, обычно заканчивающиеся расширением ".ipynb".

Jupyter Notebook можно преобразовать в несколько открытых стандартных форматов вывода (HTML, слайды презентации, LaTeX, PDF, ReStructuredText, Markdown, Python) с помощью функции «Загрузить как» в веб-интерфейсе, с помощью библиотеки nbconvert или команды «jupyter nbconvert». Чтобы упростить визуализацию документов Jupyter Notebook в Интернете, библиотека nbconvert предоставляется как услуга через NbViewer, которая может принимать URL-адрес любого общедоступного документа Jupyter Notebook, на лету конвертировать его в HTML и отображать его пользователю.

Jupyter Notebook может подключаться ко многим ядрам, что позволяет программировать на многих языках. По умолчанию Jupyter Notebook поставляется с ядром IPython. На момент выпуска версии 2.3 (октябрь 2014 г.) существует 49 Jupyter-совместимых ядер для многих языков программирования, включая Python, R, Julia и Haskell.

Интерфейс Notebook был добавлен в IPython в версии 0.12 (декабрь 2011 г.), а в 2015 г. переименован в Jupyter notebook (IPython 4.0 - Jupyter 1.0). Jupyter Notebook похож на интерфейс других программ, таких как Maple, Mathematica и SageMath, стиль вычислительного интерфейса, зародившийся в Mathematica в 1980-х годах. По данным The Atlantic, в начале 2018 года интерес к Jupyter обогнал популярность интерфейса ноутбука Mathematica.

2.6 Microsoft Visual Studio

Microsoft Visual Studio - это интегрированная среда разработки (IDE) от Microsoft. Она используется для разработки компьютерных программ, а также веб-сайтов, веб-приложений, веб-сервисов и мобильных приложений. Visual Studio использует платформы разработки программного обеспечения Microsoft, такие как Windows API, Windows Forms, Windows Presentation Foundation, Windows Store и Microsoft Silverlight.

Visual Studio включает редактор кода, поддерживающий IntelliSense (компонент автодополнения кода), а также инструмент рефакторинга кода. Интегрированный отладчик работает как отладчик на уровне исходного кода, так и как отладчик на уровне компьютера. Другие встроенные инструменты включают профилировщик кода, конструктор для создания приложений с графическим интерфейсом, веб-дизайнер, конструктор классов и конструктор схемы базы данных. Он принимает плагины, которые расширяют функциональность почти на всех уровнях, включая добавление поддержки систем управления версиями (таких как Subversion и Git) и добавление новых наборов инструментов, таких как редакторы и визуальные дизайнеры, для языков, специфичных для предметной области, или наборов инструментов для других аспектов разработки программного обеспечения.

Visual Studio поддерживает 36 различных языков программирования и позволяет редактору кода и отладчику поддерживать (в разной степени) практически любой язык программирования при условии, что существует служба для конкретного языка. Встроенные языки включают C, C ++, C ++ / CLI, Visual Basic .NET, C , F , JavaScript, TypeScript, XML, XSLT, HTML и CSS. Поддержка других языков, таких как Python, Ruby, Node.js и M среди других, доступна через плагины. Java (и J) поддерживались в прошлом.

Самая базовая версия Visual Studio, версия Community, доступна бесплатно. Слоган Visual Studio Community edition: «Бесплатная полнофункциональная IDE для студентов, разработчиков с открытым исходным кодом и индивидуальных разработчиков».

2.7 Code::Blocks

Code::Blocks разработана командой разработчиков Code::Blocks team. Написана на C++. Она использует кроссплатформенную операционную систему. Это бесплатная IDE с открытым исходным кодом. Её можно использовать как для языка C, так и для C++. Code::Blocks поддерживает несколько компиляторов, включая MinGW / gcc, Digital Mars, Microsoft Visual C ++, Watcom, LCC, Borland C ++ и компилятор Intel C ++ и так далее.

Функции IDE включают:

- выделение синтаксиса;
- сворачивание кода с помощью компонента редактора Scintilla, C++;
- автозавершение кода;
- обозреватель классов;
- интегрированный список задач;
- интегрированный интерфейс отладчика, который поддерживает GDB (GNU Debugger) и, в некоторой степени, консольный отладчик Microsoft-CDB (Console Debugger);

Для набора инструментов wxWidgets существует интегрированный плагин быстрой разработки приложений под названием wxSmith.

2.8 KDevelop

KDevelop разработан KDE (K Desktop Environment) и полезен в системе Linux, чтобы предоставить пользователю систему, похожую внешне на Windows. Написана на C++. Она использует кроссплатформенную операционную систему и работает на платформе KDE. Это бесплатное программное обеспечение. KDE не включает в состав KDevelop компилятор. Вместо этого используется внешний компилятор, такой как gcc/g++, для создания исполняемого кода. KDevelop использует встроенный текстовый редактор, а редактором по умолчанию является KDE Advanced Text Editor. Этот редактор по умолчанию можно заменить редактором на основе Qt. Он включает в себя такие функции, как редактор исходного кода с подсветкой синтаксиса и автоматическим отступом, браузер классов, конструктор графического интерфейса пользователя, интерфейс для коллекции компиляторов GNU и отладчик GNU, мастера для создания и обновления определений классов и инфраструктуры приложения, автоматическое завершение кода, встроенное в поддержке Doxygen. Он использует архитектуру на основе плагинов. Это не зависит от языка программирования и системы. Он поддерживает другие технологии, такие как Qt, GTK + и wxWidgets.

2.9 CodeLite

CodeLite разработана Eran Ifrah на C ++. Использует кроссплатформенную операционную систему. Это бесплатная IDE с открытым исходным кодом для языков C и C ++. Для программирования используется набор инструментов wxWidget. Программа компилируется и отлаживается с помощью бесплатных инструментов (MinGW и GDB) для Mac OS X, Windows, Linux и FreeBSD. Она запускает любой сторонний компилятор или инструмент с интерфейсом командной строки. В его функции входят управление проектами (рабочая область / проекты), завершение кода, рефакторинг кода, просмотр исходного кода, выделение синтаксиса, интеграция с Subversion, интеграция cscope, интеграция Unittest ++, интерактивный отладчик, построенный на GDB, и редактор исходного кода.

2.10 Dev-C++

Dev-C ++ разработана Bloodshed Software на Delphi. Она была создана для операционной системы Microsoft Windows. Dev-C++ - это бесплатная IDE, распространяемая под Стандартной общественной лицензией GNU для программирования на C и C++. Она поставляется в комплекте с MinGW, бесплатным компилятором. Хостинг проекта - SourceForge. Dev-C++ была первоначально разработана программистом Колином Лапласом. Dev-C++ также можно использовать в сочетании с Cygwin или любым другим компилятором на основе gcc. Еще один аспект Dev-C++ - это использование DevPaks, упакованных расширений в среде программирования с дополнительными библиотеками, шаблонами и утилитами. Пакеты DevPak часто содержат, (но не ограничиваются ими), утилиты с графическим интерфейсом, в том числе популярные наборы инструментов, такие как GTK+, wxWidgets и FLTK. Пользователи Dev-C++ могут загружать дополнительные библиотеки или пакеты кода, которые увеличивают объем и функциональность Dev-C++, например графику, сжатие, анимацию, поддержку звука и многое другое.

2.11 BlueJ

BlueJ разработана BlueJ team. Первоначальный автор - Майкл Коллинг. Эта IDE написан на языке Java. Она использует кроссплатформенную операционную систему и платформу Java. BlueJ разработана в основном для образовательных целей, но подходит для разработки маломасштабного программного обеспечения. BlueJ была создана для поддержки преподавания и изучения объектно-ориентированного программирования. Её функции включают представление объектной ориентации, простоту интерфейса, взаимодействие с объектами, панель кода, регрессионное тестирование, поддержку групповой работы, гибкую систему расширений, файлы Jar и апплеты, переводы и т. д. Работа в среде BlueJ позволяет получить конкретный опыт использования таких абстрактных концепции, как отношения

класса / объекта, создание экземпляра объекта, вызов метода, передача параметров и т. д. Эти абстрактные концепции традиционно трудны для понимания новичком, и предоставление конкретных представлений для них предназначено для облегчения процесса обучения. Хотя это эффективное программное обеспечение, в нем отсутствуют некоторые функции, такие как проверка кода в реальном времени и обнаружение ошибок, предлагаемые исправления для предупреждений / ошибок, сворачивание кода и т. д.

2.12 NetBeans

NetBeans - продукт корпорации Oracle. Эта IDE написана на Java для кроссплатформенной операционной системы. Она использует платформу Java SE для исполнения программ. IDE NetBeans работает везде, где установлена JVM, включая Windows, Mac OS, Linux и Solaris. Платформа NetBeans позволяет разрабатывать приложения из набора составных программных компонентов, называемых модулями. Приложения, основанные на платформе NetBeans (включая IDE NetBeans), могут быть расширены сторонними разработчиками. IDE NetBeans - это интегрированная среда разработки с открытым исходным кодом. IDE NetBeans поддерживает разработку всех типов приложений Java, а именно Java SE, включая JavaFX, Java ME, EJB и т. д. Поддерживает функцию модульности. Каждый модуль предоставляет четко определенные функции, такие как поддержка языка Java, редактирование или поддержка CVS (система одновременного управления версиями) и SVN (дополнительный номер версии). NetBeans также включает профилировщик, который помогает разработчикам находить утечки памяти и оптимизировать скорость. Инструменты проектирования графического интерфейса пользователя позволяют создавать прототипы и проектировать графические интерфейсы пользователя Swing путем перетаскивания и размещения компонентов графического интерфейса. Редактор JavaScript обеспечивает расширенную поддержку JavaScript, Ajax и CSS. Функции редактора JavaScript включают выделение синтаксиса, рефакторинг, завершение кода для собственных объектов и функций, создание скелетов классов JavaScript, создание обратных вызовов Ajax из шаблона и автоматические проверки совместимости браузера.

2.13 Сравнение IDE для языков C/C++

Таблица 2.1 — Сравнение IDE для языков C/C++

IDE	Лицензия	Windows	Linux	Отладчик	Автодо-полнение	Браузер классов
Code::Blocks	GPL	Да	Да	Да	Да	Да
CodeLite	GPL	Да	Да	Да	Да	Да
Dev-C++	GPL	Да	Нет	Да	Нет	Нет
Eclipse	EPL	Да	Да	Да	Да	Нет
KDevelop	GPL	Да	Да	Да	Да	Да
NetBeans	CDDL, GPL, LGPL	Да	Да	Да	Да	Да
Qt Creator	GPL	Да	Да	Да	Да	Да
Visual Studio	Проприетарная	Да	Нет	Да	Да	Да

2.14 Сравнение IDE для Python

Таблица 2.2 — Сравнение IDE для Python

IDE	Лицензия	Windows	Linux
NetBeans	CDDL, GPL, LGPL	Да	Да
PyCharm	ASL	Да	Да
Spyder	MIT	Да	Да
Eclipse	EPL	Да	Да
Visual Studio	Проприетарная	Да	Нет

2.15 Сравнение IDE для Java

Таблица 2.3 — Сравнение IDE для Java

IDE	Лицензия	Windows	Linux	Разработка GUI
Eclipse	EPL	Да	Да	Да
IntelliJ Idea	ASL	Да	Да	Да
KDevelop	GPL	Нет	Да	Нет
NetBeans	CDDL, GPL, LGPL	Да	Да	Да
BlueJ	GPL	Да	Да	Нет

2.16 Vim как IDE

Vim - это клон с текстового редактора vi Билла Джоя для Unix с некоторыми дополнениями. Автор Vim, Брэм Мооленаар, основал его на исходном коде для переноса редактора Stevie на Amiga и выпустил общедоступную версию в 1991 году. Vim разработан для использования как из интерфейса командной строки, так и как отдельное приложение с графическим пользовательским интерфейсом. Vim - это бесплатное программное обеспечение с открытым исходным кодом. Лицензия совместима с Стандартной общественной лицензией GNU через специальный пункт, разрешающий распространение измененных копий «под GNU GPL версии 2 или любой более поздней версии».

С момента выпуска для Amiga кроссплатформенная разработка сделала Vim доступным для многих других систем. В 2006 году он был признан самым популярным редактором среди читателей Linux Journal; В 2015 году опрос разработчиков Stack Overflow показал, что он стал третьим по популярности текстовым редактором и пятой по популярности средой разработки в 2019 году.

Пусть Vim и является в первую очередь текстовым редактором, количество разработчиков, использующих его в качестве IDE, очень велико. Vim не загромождён ненужными функциями, в отличие от многих тяжеловесных IDE, большая часть функционала которых оказывается бесполезна для большинства пользователей. VIM позволяет настроить как общую конфигурацию для всех проектов, так и отдельную для каждого языка программирования. Есть возможность загрузить одни плагины для C, другие для Python и другие для Java.

Обычная IDE может имеет тесно интегрированную языковую поддержку и более практичные функции для рефакторинга кода. Однако, например, может оказаться, что программисту, пишущему код в основном на C++, возникла необходимость взглянуть на код Java. При наличии Vim не нужно устанавливать отдельную IDE и получить минимальную языковую поддержку без дополнительных действий.

В Vim все операции выполняются с клавиатуры, что помогает не отвлекаться от написания кода.

Vim - гибкий и практичный инструмент, но его нужно настроить заранее. IDE - в большинстве случаев позволяет сразу приступить к работе. Vim является мощным и многофункциональным инструментом, в то время как IDE - мощный, но узкоспециализированный инструмент. Vim поддерживает сколько угодно языков, IDE - поддерживает 1-3 языка, но лучше, чем VIM.

Тем не менее Vim нужны плагины для выполнения некоторых функций, доступных в обычных IDE, которые не встроены в Vim изначально. Вот несколько плагинов Vim, которые делают его более похожим на IDE:

2.16.1 Просмотр проекта / файлового дерева

■ NERDTree - это плагин древовидного обозревателя для навигации по файловой системе;

■ vtreeexplorer - это файловый менеджер на основе дерева;

■ project дает "проектное"представление файлов, а не прямое представление файловой системы;

■ ide отслеживает статус файлов (открытые / отредактированные / закрытые / только для чтения) в проекте с помощью значков; автоматически строит и обновляет правила подсветки синтаксиса на основе файлов проекта (C / C ++ / Java); избегает дублирования буфера.

■ :help netrw нужен для получения информации о проводнике, поставляемом с Vim. По умолчанию он не отображает файлы в виде дерева, но может использовать параметр g: netrw_liststyle. Он также предлагает полезные параметры сортировки файлов (по дате, размеру, имени).

■ Local_vimrc управляет проектами как файлами в одном дереве каталогов;

■ Projectionist обеспечивает детальную конфигурацию проекта с использованием «проекций»;

2.16.2 Буфер / просмотр файлов

■ bufexplorer позволяет перемещаться по открытым буферам;

■ minibufexpl - элегантный проводник буферов; занимает очень мало места на экране;

■ lookupfile - файлы поиска с использованием Vim7 ins-Completion;

■ плагин Command-T, вдохновленный окном «Перейти к файлу», привязанным к Command-T в TextMate;

■ MRU - доступ к недавно открывавшимся файлам;

■ ctrlp - поиск с поддержкой регулярных выражений, предоставляет доступ ко всем функциям с помощью ctrl-p;

2.16.3 Просмотр кода

■ taglist дает представление об исходном коде, который просматривается в данный момент;

■ Tagbar похож на список тегов, но может упорядочивать теги по объему. Рекомендуется для языков программирования с классами, например C ++, Java, Python;

■ Indexer автоматически генерирует теги для всех файлов в проекте и поддерживает теги в актуальном состоянии. Использует ctags. Хорошо работает с плагином project или самостоятельно;

■ CCTree - это обозреватель дерева вызовов, браузер исходного кода на основе Cscope и анализатор потока кода;

■ exUtility - глобальный поиск, поиск символов, отслеживание тегов;

■ ShowMarks наглядно показывает расположение отметок;

■ lh-tags автоматически обновляет базу данных ctags и предоставляет альтернативу «: tselect» для навигации по коду;

2.16.4 Написание кода

■ AutoComplPop завершает код по мере ввода;

■ YouCompleteMe - еще один плагин завершения;

■ CRefVim Справочное руководство по C, специально разработанное для Vim;

2.16.5 Функциональность Vim

■ bufkill позволяет очистить буфер, не закрывая окно;

■ undotree или gundo визуализирует дерево отмены;

■ Surround упрощает удаление / изменение / добавление скобок / кавычек / XML-тегов / многое другое;

2.16.6 Компиляция

■ vim-dispatch позволяет асинхронно запускать команды оболочки. При запуске компилятора окно быстрого исправления будет заполнено любыми потенциальными ошибками;

■ Build Tools Wrapper предоставляет способы компилировать программы (возможность компилировать в фоновом режиме, на нескольких ядрах и т. д.), а также тестировать и выполнять программы. Этот плагин также может фильтровать выходные данные компиляции на лету. При компиляции проектов под CMake care плагин BTW позволяет переключать режим компиляции (на самом деле каталог). Текущий режим компиляции (и имя проекта) будет отображаться в строке состояния каждого буфера (в том числе буфера быстрого исправления) через плагин airline;

2.16.7 Отладка

■ встроенный плагин отладчика терминала в комплекте;

■ clewn реализует полную поддержку gdb в редакторе vim: точки останова, контрольные переменные, завершение команды gdb, окна сборки и т. д.;

■ pyClewn похож на clewn, но написан на Python и также поддерживает pdb.;

■ vim-debug, который создает интегрированную среду отладки в VIM;

■ плагин gdbvim позволяет просматривать в vim, то, что отлаживается в gdb;

■ vim-lldb: обеспечивает интеграцию отладки lldb;

2.17 Emacs

Emacs - это семейство текстовых редакторов, которые отличаются своей расширяемостью. В руководстве по наиболее широко используемому варианту, GNU Emacs, он описывается как «расширяемый, настраиваемый, самодокументирующийся редактор отображения в реальном времени». Разработка первого Emacs началась в середине 1970-х годов, и работа над его прямым потомком, GNU Emacs, активно продолжается до сих пор.

Emacs имеет более 10 000 встроенных команд, а его пользовательский интерфейс позволяет пользователю объединять эти команды в макросы для автоматизации работы. Реализации Emacs обычно содержат диалект языка программирования Lisp, который обеспечивает возможность глубокого расширения, позволяя пользователям и разработчикам писать новые команды и приложения для редактора. Были написаны расширения для управления электронной почтой, файлами, схемами и RSS-потоками, а также клонами ELIZA, Pong, Conway's Life, Snake и Tetris.

Оригинальный EMACS был написан в 1976 году Дэвидом А. Муном и Гаем Л. Стилом-младшим как набор редакторов MACroS для редактора TECO. Он был вдохновлен идеями TECO-макроредакторов TECMAC и TMACS.

Самая популярная и наиболее портируемая версия Emacs - это GNU Emacs, созданная Ричардом Столлманом для проекта GNU. XEmacs - это вариант, ответвившийся на GNU Emacs в 1991 году. GNU Emacs и XEmacs используют похожие диалекты Lisp и по большей части совместимы друг с другом. Разработка XEmacs неактивна.

Emacs, наряду с vi, является одним из двух основных соперников в традиционных войнах редакторов в культур Unix. Emacs - один из старейших бесплатных проектов с открытым исходным кодом, который все еще находится в стадии разработки.

2.18 Сравнение Emacs и Vim

Таблица 2.4 — Сравнение Emacs и Vim

Функция	Vim	Emacs
Нажатие клавиши	Vim сохраняет каждую комбинацию нажатых клавиш. Это создает путь в дереве решений, который однозначно идентифицирует любую команду.	Команды Emacs - это комбинации клавиш, для которых клавиши-модификаторы удерживаются, в то время как другие клавиши нажимаются; команда выполняется после полного ввода. Это по-прежнему формирует дерево решений из команд, но не из отдельных нажатий клавиш. Пакет Emacs на основе vim (дерево отмены) предоставляет пользовательский интерфейс для дерева.
Пользовательская среда	Vim, как и Emacs, изначально использовался исключительно внутри текстовой консоли, не предлагая графического интерфейса (GUI). Многие современные производные vi, например MacVim и gVim включают графические интерфейсы. Однако поддержка пропорциональных шрифтов по-прежнему отсутствует. Также отсутствует поддержка шрифтов разного размера в одном документе.	Emacs, изначально предназначенный для использования из консоли, имел поддержку графического интерфейса X11, добавленную в Emacs 18 и сделанную по умолчанию в версии 19. Текущие графические интерфейсы Emacs включают полную поддержку пропорционального интервала и изменения размера шрифта. Emacs также поддерживает встроенные изображения и гипертекст.
Поддержка языков и скриптов	Vim имеет элементарную поддержку других языков, кроме английского. Современный Vim поддерживает Unicode, если используется с терминалом, поддерживающим Unicode.	Emacs полностью поддерживает все Unicode-совместимые системы письма и позволяет свободно смешивать несколько скриптов.
Интерфейс навигации	Vim использует различные режимы редактирования. В «режиме вставки» клавиши вставляют символы в документ. В «нормальном режиме» (также известном как «командный режим», не путать с «режимом командной строки», который позволяет пользователю вводить команды), простые нажатия клавиш выполняют команды Vim.	Emacs использует одновременное нажатие горячих клавиш. Клавиши или связки клавиш можно определить как префиксные команды, которые переводят Emacs в режим ожидания дополнительных нажатий клавиш, составляющих окончание команды. Связки клавиш могут зависеть от режима, изменяя стиль взаимодействия.

3 Научная задача в рамках спецсеминара

Моя работа в рамках спецсеминара рассматривает возможность применения нейросетевых методов к решению проблемы сегментации и детекции объектов по многоволновым данным космических телескопов (в данном случае оптического, микроволнового и рентгеновского диапазонов). В качестве основы для нейросетевой архитектуры использовалась модель U-net.

Вся программная часть научной работы ведётся на языке Python, так как его библиотеки лучше всего подходят для обработки научных данных. Кроме того, этот язык является самым популярным инструментом в области глубокого обучения, и для этих целей также существует большое количество библиотек для разных случаев.

Обработка космических данных подразумеват наличие возможности добавления иллюстраций совместно с кодом, проведение экспериментов и сравнение их между собой. Лучше всего для такого формата работы подойдёт Jupyter Notebook. «Ноутбуками» удобно обмениваться, изменять и редактировать отдельные данные, проверяя эксперимент в одельной «клетке», при этом не нужно перезапускать всю программу заново. Для обучения нейронных сетей формат «ноутбуков» тоже очень хорошо подходит - вместе с кодом программы сохраняется информация о том, как проходило обучение модели.

Однако кроме проведения экспериментов и обучения нейросетевых моделей иногда приходится программировать модули с функциями, которые так или иначе понадобятся ещё не раз - и лучше всего здесь справляется Vim. С его помощью можно быстро открыть файл нужного модуля, найти функцию и отредактировать её параметры. Однако для этого можно использовать любую IDE, но Vim окажется быстрее большинства таких вариантов.

Заключение

По информации выше можно судить, что для всех популярных языков программирования существует огромное множество сред разработки. В общем смысле они имеют одни и те же базовые функции, и выбор пользователя уже зависит от ограничений отдельных IDE.

Для кого-то важнее всего будет возможность настроить абсолютно любую деталь интерфейса, для кого-то принципиальна совместимость с наибольшим количеством операционных систем (если, например, разработчик по каким-то причинам пользуется средой разработки на нескольких устройствах с разными операционными системами), для кого-то важна возможность добавить определённый компилятор или отладчик для работы, для кого-то важна скорость загрузки IDE и её легковесность (если, например нужно работать на устройстве с ограниченными характеристиками).

Идеальным вариантом является Vim, как инструмент обладающий абсолютно всеми возможными свойствами, однако для его настройки требуется большое количество времени, и ещё больше времени нужно пользователю для того, чтобы запомнить все нужные команды и в принципе освоиться.

Так или иначе выбор IDE будет зависеть от конкретной ситуации и от разрабатываемого продукта. Лучше всего, когда у разработчика есть выбор, и он имеет возможность использовать любимую IDE - это может положительно сказаться на качестве продукта. Однако при работе в команде приходиься учитывать интересы других разработчиков и общие правила компании. В любом случае, после определенного количества времени, проведённого за работой в какой-либо IDE, программист привыкает к ней, и скорость и качество работы уже не будут зависеть от этого фактора.

Список использованных источников

1. *С. Давыдов, А. Ефимов.* IntelliJ IDEA. Профессиональное программирование на Java. / А. Ефимов. С. Давыдов. — СПб.: БХВ, 2005.

2. *S. Satav S. Satpathy, K. Satao.* A Comparative Study and Critical Analysis of Various Integrated Development Environments of C, C++, and Java Languages for Optimum Development / K. Satao S. Satav, S. Satpathy. — 2011.

3. *S. Kavitha, S. Sindhu.* COMPARISON OF INTEGRATED DEVELOPMENT ENVIRONMENT (IDE) DEBUGGING TOOLS: ECLIPSE VS NETBEANS / S. Sindhu S. Kavitha. — International Research Journal of Engineering and Technology, 2015.

4. *S. Saunders D. K. Fields, E. Belayev.* IntelliJ IDEA in action / E. Belayev S. Saunders, D. K. Fields. — Greenwich, Conn. : Manning, 2006.

5. *Анфилатов, И.* Немного о VIM и IDE / И. Анфилатов. — Habr, 2017.

6. *Boudreau T.; Glick, J.; Greene S.* NetBeans: The Definitive Guide / J.; Greene S. Boudreau, T.; Glick. — 2002.

7. *A. Robbins E. Hannah, L. Lamb.* Learning the vi and Vim Editors / L. Lamb A. Robbins, E. Hannah. — O'Reilly Media, 2008.

8. *Gove, Darryl.* Solaris Application Programming / Darryl Gove. — 2008.

9. *Barnes David J.; Kölling, Michael.* Objects First with Java: A Practical Introduction Using BlueJ / Michael Barnes, David J.; Kölling. — 2011.

10. *Holzner, Steve.* Eclipse / Steve Holzner. — 2004.

11. *Burnette, Ed.* Eclipse IDE Pocket Guide / Ed Burnette. — 2005.

12. *Ciccarelli, Eugene.* An Introduction to the Emacs Editor / Eugene Ciccarelli. — 1978.

13. *Stallman, Richard M.* EMACS: The Extensible, Customizable, Self-Documenting Display Editor / Richard M. Stallman. — 1981.

14. *Glickstein, Bob.* Writing GNU Emacs Extensions / Bob Glickstein. — 1997.

15. *Neil, Drew.* Practical Vim / Drew Neil. — 2015.

16. *Oualline, Steve.* Vim (Vi Improved) / Steve Oualline. — 2001.

17. *Krochmalski, Jaroslaw.* IntelliJ IDEA Essentials / Jaroslaw Krochmalski. — 2014.

18. *Eng, Lee Zhi.* Qt5 C++ GUI Programming Cookbook / Lee Zhi Eng. — 2019.

19. *Ritchie, Peter.* Practical Microsoft Visual Studio 2015 / Peter Ritchie. — 2016.

20. *Шарп, Джон.* Microsoft Visual C. Подробное руководство / Джон Шарп. — 2015.