**Роль и значение автоматизированных информационных систем на практике**

Информационная система – организационно-техническая система, которая предназначена для выполнения информационно-вычислительных работ или предоставления информационно-вычислительных услуг, удовлетворяющих потребности системы управления и ее пользователей – управленческого персонала, внешних пользователей (инвесторов, поставщиков, покупателей) путем использования и/или создания информационных продуктов.

Информационно-вычислительная работа – деятельность, связанная с использованием информационных продуктов. Типичным примером информационной работы является поддержка информационных технологий управления.

Информационно-вычислительная услуга – это разовая информационно-вычислительная работа.

Информационные системы существуют в рамках системы управления и полностью подчинены целям функционирования этих систем.

Система управления представляет собой совокупность объекта управления, например, предприятие, и субъекта управления – управленческого аппарата. Аппарат объединяет в себе сотрудников, формирующих цели, перерабатывающих информацию, вырабатывающих и принимающих решения, а также контролирующих их выполнение.

Теперь ближе к конкретике. В моем случае я разрабатываю автоматизированную информационную систему «Абитуриент колледжа» при помощи языка C++/CLI в среде Visual Studio 2017 c использованием Windows Forms и компактной встраиваемой реляционной базой данных – SQLite, которая поддерживает достаточно полный набор команд SQL и доступна в исходных кодах (на языке C), следовательно, ее можно совершенно спокойно использовать и с компилятором g++, и с VS. Для g++ необходимо скомпилировать ее из исходных кодов через gcc: gcc -c sqlite3.c, далее присоединить файл на выходе к рабочему проекту: g++ main.cpp sqlite3.o.

В Microsift Visual Studio можно создавать базу данных SQLite как программным способом с использованием провайдера ADO.NET, так и используя Server Explorer, который доступен в Visual Studio. В целом провайдер для SQLite реализует весь функционал, который необходим для работы с базами как на связанном, так и несвязанном уровнях ADO.NET.

**Обзор среды программирования Visual Studio**

Microsoft Visual Studio —интегрированная среда разработки программного обеспечения и ряд других инструментальных средств. Она была разработана в 1997 году компанией Microsoft. Данная среда позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода , добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения.

* 1. **Системные требования**

Программа была написана на персональном компьютере MacBook Air (13-inch, Early 2015) со следующими характеристиками:

* Двухъядерный процессор Intel Core i5-5250U, 1600 МГц (12 x 133).
* Графика Intel HD Graphics 6000 1536 МБ
* Оперативная память 8 ГБ 1600 МГц DDR3
* ОС Microsoft Windows 7 Ultimate Service Pack 1 в виртуальной машине Parallels Desktop версии 13.0.1 (42947)

## Анализ средств разработки

Сколько существуют компьютеры, столько же и существуют средства разработки программного обеспечения. С течением времени они совершенствуются, постоянно прогрессируют и с каждым разом становятся все удобнее в использовании и проще в изучении.

Для начала подвергнем анализу наиболее популярные языки программирования: Java, C#, C++ и Visual Basic [Data Structures…, 1999].

### Анализ языка программирования

Таблица 1. Парадигмы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Возможность | Язык | | | |
| Java | C# | C++ | Visual Basic |
| Императивная | + | + | + | + |
| Объектно-ориентированная | + | + | + | + |
| Рефлексивная | -/+ | +/- | +/- | +/- |
| Обобщенное программирование | + | + | + | + |
| Логическая | - | - | - | - |
| Распределительная | + | -/+ | +/- | - |

* Императивная. Императивный язык описывает не задачу, а способ ее решения.
* Объектно-ориентированная. Она основана на представлении всего в языке в виде объектов. Объекты, в свою очередь, могут содержать в себе как переменные, так и методы для работы с ними. Выполняется поддержка основополагающих принципов ООП: полиморфизм, наследование и инкапсуляция.
* Рефлексивная. Это возможность, предполагающая, что язык программирования может оперировать собственным кодов как данными.
* Обобщенное программирование. При написании алгоритмов есть возможность передавать им данные любых типов.
* Логическая. Программа является набором некоторых правил и фактов вывода в определенном логическом исчислении.
* Распределительная. Поддержка языком программирования возможности распараллеливания вычислений.

Таблица 2. Типизация

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Возможность | Язык | | | |
| Java | C# | C++ | Visual Basic |
| Статическая типизация | + | + | + | + |
| Явная типизация | + | + | + | + |
| Неявное приведение типов без потери данных | - | + | + | + |
| Неявное приведение типов с потерей данных | - | - | + | + |
| Неявное приведение типов в неоднозначных ситуациях | - | + | + | + |
| Алиасы типов | - | + | + | - |
| Информация о типах в runtime | + | + | -/+ | + |
| Информация о типах-параметрах в runtime | - | + | -/+ | + |

* Статическая типизация. Невозможность изменения типов в процессе выполнения после их объявления.
* Явная типизация. Параметры и типы переменных указываются явно.
* Неявное приведение типов без потери данных. Например, приведение числа с плавающей точкой к целому числу.
* Неявное приведение типов с потерей данных. Например, приведение целого числа к числу с плавающей точкой.
* Неявное приведение типов в неоднозначных ситуациях. Допустим, при попытке сложения числа 7 и строки «3» мы можем получить, как число 10, так и строку «73».
* Алиасы типов. Замена одного алиаса типа на другой, полностью эквивалентный ему, например, #typedef TYPE1 TYPE2.
* Информация о типах в runtime. Наличие возможности выяснить точный тип объекта в runtime.

Таблица 3. Компилятор-интерпретатор

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Возможность | Язык | | | |
| Java | C# | C++ | Visual Basic |
| Open-source | + | + | + | + |
| Возможность компиляции | + | + | + | + |
| Bootstrapping | + | + | + | ? |
| Многопоточная компиляция | + | - | + | + |
| Интерпретатор командной строки | - | - | +/- | + |
| Условная компиляция | -/+ | + | + | + |

* Open-source компилятор или интерпретатор. Эта возможность говорит о наличии полноценного open-source компилятора или интерпретатора.
* Возможность компиляции. Способность компилировать код в нативный или в byte-cod с возможностью JIT компиляции.
* Bootstrapping. Это значит, что компилятор написан на том же языке, что и компилируется им.
* Многопоточная компиляция. Способность компилятора распараллеливать процесс сборки на несколько потоков при условии наличии таковых.
* Интерпретатор командной строки. Возможность вводить инструкции языка в командную строку с последующим незамедлительным их выполнением.
* Условная компиляция. Наличие возможности включения или отключения компиляции участка программного кода в зависимости от выполнения условий, например, в C++ это делается с помощью #if … #endif.

Таблица 4.Управление памятью

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Возможность | Язык | | | |
| Java | C# | C++ | Visual Basic |
| Создание объектов на стеке | - | + | + | - |
| Неуправляемые указатели | - | + | + | - |
| Ручное управление памятью | - | + | + | - |
| Сборка мусора | + | + | -/+ | + |

* Объекты на стеке. Способность создавать экземпляры любого типа данных не «в куче», а на стеке.
* Неуправляемые указатели. Характерен прямой доступ к памяти и наличие адресной арифметики.
* Ручное управление памятью. Возможность оперировать с данными «в куче» посредством, например, операторов new и delete в языке C++.
* Сборка мусора. Компилятор способен сам отслеживать неиспользуемые участи памяти в куче и освобождать их.

Таблица 5. Управление потоками вычислений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Возможность | Язык | | | |
| Java | C# | C++ | Visual Basic |
| Конструкция goto | - | + | + | + |
| Инструкция break без метки | + | + | + | + |
| Инструкция break с меткой | + | - | - | + |
| Поддержка try/catch | + | + | + | + |
| Ленивые вычисления | - | -/+ | + | - |

* Конструкция goto. Подразумевает возможность безусловного перехода к метке.
* Инструкция break без метки. Безусловный выход из ближайшего цикла.
* Инструкция break с меткой. Поддержка безусловного выхода из цикла, помеченного меткой.
* Поддержка try/catch. Наличие возможности обрабатывать исключения с помощью конструкции try/catch.
* Ленивые вычисления. Предполагает экономию времени на проведении вычислений, результаты которых в дальнейшем не понадобятся.

Таблица 6. Типы и структуры данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Возможность | Язык | | | |
| Java | C# | C++ | Visual Basic |
| Кортежи | - | +/- | +/- | +/- |
| Многомерные массивы | +/- | + | + | + |
| Динамические массивы | +/- | +/- | + | + |
| Ассоциативные массивы | +/- | + | + | + |
| Контроль границ массивов | + | + | +/- | + |
| Цикл foreach | + | + | + | + |
| Целые числа произвольной длины | + | + | - | + |
| Целые числа с контролем границ | - | - | - | - |

* Кортежи. Способность компилятора вернуть из метода кортеж. Кортеж – это неименованный тип данных, содержащий безымянные поля произвольного типа.
* Многомерные массивы. Наличие в языке возможности создавать многомерные массивы, например, array[N][M].
* Динамические массивы. Наличие в языке возможности создавать массивы, которые способны изменять свой размер в процессе выполнения программного кода.
* Ассоциативные массивы. Это так называемые hash-таблицы.
* Цикл foreach. Наличие в языке конструкции, благодаря которой существует возможность перебора всех элементов коллекции.
* Наличие поддержки длинной целочисленной арифметики.
* Целые числа с контролем границ. Определение типа чисел с заданным диапазоном, например, int range [-77, 69], и при попытке присвоения переменной range значения, выходящего за границы, происходила бы ошибка.

Таблица 7. Объектно-ориентированные возможности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Возможность | Язык | | | |
| Java | C# | C++ | Visual Basic |
| Интерфейсы | + | + | + | + |
| Мультиметоды | - | -/+ | -/+ | - |
| Переименование членов при наследовании | - | - | -/+ | - |
| Множественное наследование | - | - | + | - |

* Интерфейсы. Наличие синтаксической и семантической конструкций в программном коде, обеспечивающая специфицирование услуг, предоставляемых классом.
* Множественное наследование. Способность при создании класса наследоваться от нескольких, что позволяет производному классу содержать в себе функционал базовых классов.

Таблица 8. Функциональные возможности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Возможность | Язык | | | |
| Java | C# | C++ | Visual Basic |
| First class function | - | + | + | ? |
| Анонимные функции | - | + | + | + |
| Лексические замыкания | + | + | + | + |
| Частичное применение | - | ? | +/- | ? |
| Каррирование | - | + | +/- | - |

* First class function или объекты первого класса. В контексте определенного языка это названия сущностей, способных передаваться в качестве параметра, быть полученными при выполнении функции или быть присвоенными переменной.
* Анонимные функции. Наличие особого вида функций, объявляемых в месте использования и не получающих уникального идентификатора для их вызова.
* Лексическое замыкание. Наличие в функции ссылки на переменную, объявленную вне тела функции и не переданную в нее в качестве параметра.

Таблица 9. Прочие свойства

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Возможность | Язык | | | |
| Java | C# | C++ | Visual Basic |
| Макросы | - | - | + | + |
| Шаблоны | + | + | + | + |
| Поддержка Unicode в идентификаторах | + | + | + | + |
| Перегрузка функций | + | + | + | + |
| Динамические переменные | ? | ? | + | + |
| Значение параметров по умолчанию | - | + | + | + |
| Локальные функции | +/- | +/- | + | +/- |
| Наличие библиотек для работы с графикой | + | + | + | + |

* Макросы. Наличие макросистемы, обрабатывающей код программы до ее компиляции или выполнения.
* Шаблоны. Наличие возможности создания обобщенных классов для расширения функционала. Например, в C++ это template классы.
* Перегрузка функций. Возможность создания перегруженных функций, позволяющих принимать на вход различные наборы параметров.
* Динамические переменные. Способность языка создавать в нем переменные «в куче».
* Значение параметров по умолчанию. Способность вызывать конструкторы или функции без явной подстановки значений входных параметров, которые проинициализированы в описании функции.
* Наличие библиотек для работы с графикой. Возможность использовать функционал таких библиотек как: OpenGL, WebGL, OpenML, DirectX.

Исходя из приведенных сравнительных характеристик можно сказать, что по своей универсальности при написании программного обеспечения языку C++ нету равных. Считается, что при реализации программного продукта нужно отталкиваться от поставленной цели, и для этого в наибольшей степени подходят два языка: C++ и Java. Они оба обладают возможностью распараллеливания вычислений и созданием хорошего графического интерфейса, хотя в Java для этого требуется значительно больше оперативной памяти, что склоняет выбор не в его пользу. Также, в отличии от языка C++, Java не поддерживает прямой доступ к памяти компьютера и ленивые вычисления, что сильно увеличило бы объем кода. Помимо прочего C++ обладает более хорошей типизацией данных.

**Алгоритм работы программы «Абитуриент колледжа»**

Для оформления документов в колледж абитуриенту необходимо сдать относительно небольшое кол-во документов: аттестат за 9 или 11 класс и портфолио, но т.к. абитуриентов достаточно много, а людей в приемной комиссии, наоборот, очень мало, то данные мероприятия могут занимать достаточно продолжительное количество времени, которое можно потратить на более рациональные вещи.

Существует два варианта решения данной задачи:

1. Прием новых сотрудников для приемной комиссии
2. Оптимизация процесса приема документов в приемной комиссии

Прием новых сотрудников не особо рациональное действие: каждому следует каждый месяц выплачивать зарплату и размещать большее количество рабочих мест (стол, стул, бумага и прочие расходные материалы), поэтому мы будет действовать по второму пункту.

Оптимизировать процесс приема документов рациональнее всего с помощью автоматизации каких-либо базовых действий: занесение информации о среднем балле аттестата, формирование рейтинга абитуриентов или выставление текущей даты приема – все это можно реализовать с помощью автоматизированной системы приема абитуриентов.

За счет интуитивно понятного интерфейса для работы с программой необходимы самые базовые знания работы с персональным компьютером.

Алгоритм программы достаточно простой, есть возможность занести данные нового абитуриента, возможность посмотреть полный рейтинг абитуриентов, а также рейтинг по специальностям, вывод данных на печать, также присутствует возможность изменения и удаления абитуриентов. Для того, чтобы добавить нового абитуриента, необходимо:

1. Ввести ФИО абитуриента
2. Ввести средний балл по аттестату
3. Ввести приоритет (да/нет)
4. Ввести форму обучения абитуриента (очная/заочная)
5. Ввести направление абитуриента
6. Ввести номер заявления, если необходимо
7. Ввести форму оплаты (бюджет/договор)
8. Оригинал аттестата или копия

После записи нового абитуриента в базу данных, есть возможность распечатать отчет о приеме документов в колледж. Отчет содержит всю информацию, необходимую для зачисления абитуриента.

Таким образом, я могу утверждать, что метод работы программы предельно прост.