УДК 004.9 **Информационные технологии**

***Осипова Татьяна Александровна**, бакалавр МГТУ им. Н. Э. Баумана,*
Россия, г. Москва

АКТУАЛЬНЫЙ АНАЛИЗ ОСОБЕННОСТЕЙ ИСПОЛЬЗОВАНИЯ ЯЗЫКА C++

Аннотация: Используемый в настоящее время язык объектно-ориентированного программирования C++ содержит как достоинства, так и недостатки. Последние описываются в данной статье.

Ключевые слова: программирование, C++, языки программирования.

Abstract: The currently used C ++ language contains both advantages and disadvantages described in this article.

Key words: programming, C++, programming languages.

C++ - компилирующий язык программирования общего назначения. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков программирования. Уделяет значительное внимание поддержке объектно-ориентированного, а также обобщенного программирования.

Обратимся к истории. Так, C++ возник в начале 1980-х годов. Тогда Бьерн Страуструп придумал ряд усовершенствований к языку C, используя его под собственные нужды [4]. Вспомнив свою диссертацию, Страуструп дополнил язык программирования C, использующийся в качестве преемника BCPL, возможностями из языка Симула. Бьерн дополнил язык C возможности работы с объектами и классами. Далее, к 1983 году, в C++ появились константы, ссылки, виртуальные функции, другой стиль комментариев, перегрузка функций и операторов… Первый коммерческий выпуск C++ датируется октябрем 1985 года. Стоит отметить, что стандартная библиотека C++ тоже

развивалась вместе с данным языком. В 1989 году вышла вторая версия. В 2000-х годах проводились работы по модернизации стандартов языка. В настоящее время C++ продолжает свое развития с целью соответствия требованиям современного общества.

Тем не менее, данная статья акцентирует внимание на отрицательных сторонах C++. Обратимся к негативным особенностям рассматриваемого языка.

1) избыточность синтаксиса.

Изучение и последующее применение необходимых постулатов C++ не всегда позволяет сконцентрироваться на задачах и целях программы. Например, часто возникают сложности с использованием списков – для верного применения которых важны шаблоны, область видимости и т.д.

2) множественное наследование.

В некоторых ситуациях множественное наследование может привести к большому количеству проблем, которые могут усложнить дальнейшую поддержку кода, в том числе заметно увеличить сложность программ.

3) перегруженность исходного кода.

Существует версия о неизбежности совмещения перегрузки методов (overloading) и переопределения (overriding) метода.

4) длительное время компиляции.

C++ обладает большим временем компиляции по сравнению с его предшественником, языком C.

5) прямое использование указателей (менеджмент памяти).

Указатель в C++ рассматривается в качестве переменной, значением которой является адрес ячейки памяти. То есть указатель ссылается на блок данных из области памяти, причем на самое его начало. Прямое использование указателей может повлечь за собой множество проблем.

6) неудобство отладки.

В определенных случаях программистам C++, как и другим программистам, советуют внимательно читать, что пишет система/программа. Однако неудобств длительной отладки это не отменяет.

7) поддержка стандарта C++ разными компиляторами.

Несмотря на это, не существует выводов о превосходстве одного компилятора над другим. У каждого из существующих компилятора есть собственные области применения. Так, например, Microsoft Visual Studio Community содержит множество интересных инструментов, доступных, однако, в коммерческих версиях проекта [2]. Пользователь сможет использовать отладчик, редактор, IDE, отладчик, оптимизирующий компилятор, средства профилирования и т.д.

Следующий пример – Clang - компилятор C, C++, Objective C и Objective C++, разработанный под Apple, являющийся частью проекта LLVM. Данный компилятор реализует различные стандарты ISO C и C++, такие как C11, C++ 14, C++ 1z, и выпущен под лицензией BSD. Основной недостаток в предоставлении его только в исходной форме. Следствие – возникновение необходимости самостоятельной его сборки.

Tiny C Compiler рассматривается в качестве максимально компактного Linux- компилятора. Данный небольшой компилятор для Windows и Linux генерирует оптимизированные двоичные файлы x86. Также собирает, компонует и связывает код в несколько раз быстрее, чем GCC [1].

8) сложность имплементации, компилятора (действительно, имплементация подразумевает реализации различных методов).

Например, в большинстве случаев оптимизатор лучше пользователя разберется, что следует инлайнить, а что не стоит (в том числе если функция имеет модификатор inline, то это не значит, что она на самом деле она встраиваемая, а обычная скомпилированная функция может быть "развернута" и без явного указания).

9) большие накладные расходы на поддержание (отладку и изменение) существующего кода.

10) неоптимальный код (по сравнению с C).

В качестве типов оптимизации готового кода в C++ стоит выделить написание подходящего кода с начала, исключительно использование оптимальных конструкций и оптимизацию уже готового и проверенного кода.

11) высокий порог вхождения при изучении.

Требуется квалификация программиста. Более того, сегодня C++ продолжает идти по пути бесконечного усложнения. Современные условия с некоторой вероятностью делают путь его развития трудным. Кроме этого, для работы с данным языком программирования необходимо всегда применять ручное выделение и освобождение памяти, откуда вытекает необходимость внимательности [5; 6].

12) непригодность для больших проектов вследствие экспоненциального роста наследуемых конструкций.

13) проблемы поддержки конструкции C.

Так, языки имеют большое количество различий, и могут возникнуть трудности. Пробуя скомпилировать любую программу под любым компилятором на C++, есть риск столкнуться с основными сложностями. Есть мнения о некоторых неточностях и несоответствиях документации и языка разработки.

14) узость применения.

C++ не широко распространен. В то же время существует версия: ведущая область применения рассматриваемого C++ - крайне популярное и широко применимое системное программирование. Так, например, увидеть библиотеку, основанную на STD-STL, достаточно редкое явление. Другой пример: при использовании **C++** окажется невозможным создание портативной библиотеки для работы с регулярными выражениями, так как в этом случае не получится выполнить компиляцию и генерацию кода в выполнимом файле.

Однако, справедливости ради стоит обратить внимание: C++ не в полной мере, но использовался для создания множества полезных приложений. В качестве примеров стоит упомянуть следующие.

- основные приложения Adobe Systems;

- браузер Mozilla Firefox;

- почтовый клиент Thunderbird;

- продукты Sun: The HotSpot Java Virtual Machine, компиляторы Sun, OpenOffice;

- ОС Apple OS X написана преимущественно на C++;

- поисковой движок Гугл;

- сервер MySQL;

- ПО для самолетов F-16 и F-35;

- MySQL Cluster;

- операционная система Symbian;

- браузер Chromium;

- visual C++ частично применялся Microsoft:

- Windows XP;

- Vista, 7;

- Windows NT;

- Windows 9x;

- Microsoft Office;

- Internet Explorer;

- Visual Studio;

- SQL.

15) парадигма ООП не эффективна при решении ряда задач.

Более того, open-source сообществе не практикуется использование C++ (например, компиляторы gcc, ядро Linux, системные библиотеки, основные open-source дистрибутивы). К тому же расширения, предоставляемые C++, могут быть реализованы стандартными средствами языка C. Это дает возможность программистам “C/C++” использовать C-style.

16) плохая поддержка модульности.

В классическом языке C отсутствует модульность на уровне язык, точнее, обеспечение ее переложено на компоновщик [3]. При этом подключение интерфейса внешнего модуля через препроцессорную вставку заголовочного

файла (#include) серьезно замедляет компиляцию при подключении большого количества модулей (из-за того, что результирующий файл, обрабатываемого компилятором, оказывается слишком велик по размеру).

17) унаследованный от C препроцессор несколько отличается примитивностью.

Однако сам Б. Страуструп говорил о развитии языка и его области применения следующее: "Я не думаю, что с выходом новой версии что-либо изменится с точки зрения применения C++, и не имеет смысла искусственно продвигать этот язык в другие области. C++0x создается не для того, чтобы прорваться в новые домены, а для того, чтобы усовершенствовать язык для его более эффективного использования в сфере инфраструктурных применений, где очень много работы и постоянно появляются новые задачи".

Получены следующие выводы. Существует версия, что Б. Страуструп предложил не полностью верную концепцию дополнения существующего языка C избыточными конструкциями вместо создания отдельного инструментария. Данные дополнения не могут эффективно применяться в областях применения языка C (системное программирование) и слишком громоздкие и избыточные для создания приложений. Предполагаемая ниша C++ отчасти перекрывается языком C снизу, сверху – на уровне приложений – Java, C sharp, Python, JavaScript. Следствием этого является незавершенность стандарта C++, последних актуальных редакций языка, и, как следствие, невозможность применения новых конструкций из-за опасности кроссплатформенной несовместимости.

Библиографический список:

1. Байджелоу С. "Железо ПК. Хитрости: как перестать ковыряться в компьютере" М.: ИНФРА М, 2009, 416 с.

2. Колисниченко Д.А. " Компьютер. Большой Самоучитель По Ремонту, Сборке и Модернизации" АСТ, 2008, 260 с.

3. Романов В. П. " Техническое обслуживание средств вычислительной техники " М.: Издательско-торговая корпорация «Дашков и К», 2010, 185 с.

4. Романов В.П. " Техническое обслуживание средств вычислительной техники: Методические указания к лабораторному практикуму" Новокузнецк: ФГОУ СПО "Кузнецкий индустриальный техникум", 2008, 49 с.

5. Степаненко О.С " Сборка Компьютера" Диалектика, 2009, 544 с.

6. Цилькер Б.Я.,Орлов С.А. " Организация ЭВМ И Систем" Питер, 2010, 672 с.