

Основы программной инженерии (ПОИТ)
Технологии разработки программного обеспечения (ИСИТ)

Основные этапы разработки программ

План лекции:

- система программирования, язык программирования;
- алфавит, основные элементы языка программирования;
- символы времени трансляции, символы времени выполнения;
- этапы и цели разработки программы;
- трудоемкость этапов разработки программ.

1. Система программирования

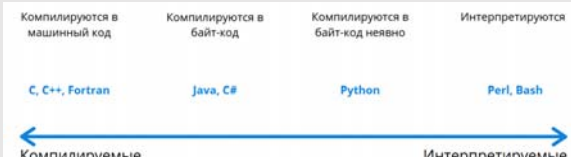
Система программирования — это комплекс инструментальных программных средств, предназначенный для автоматизации процесса разработки, отладки программного обеспечения и подготовки программного кода к выполнению.

Система программирования — это система, образуемая языком программирования, компиляторами или интерпретаторами программ, представленных на этом языке, соответствующей документацией, а также вспомогательными средствами для подготовки программ к форме, пригодной для выполнения

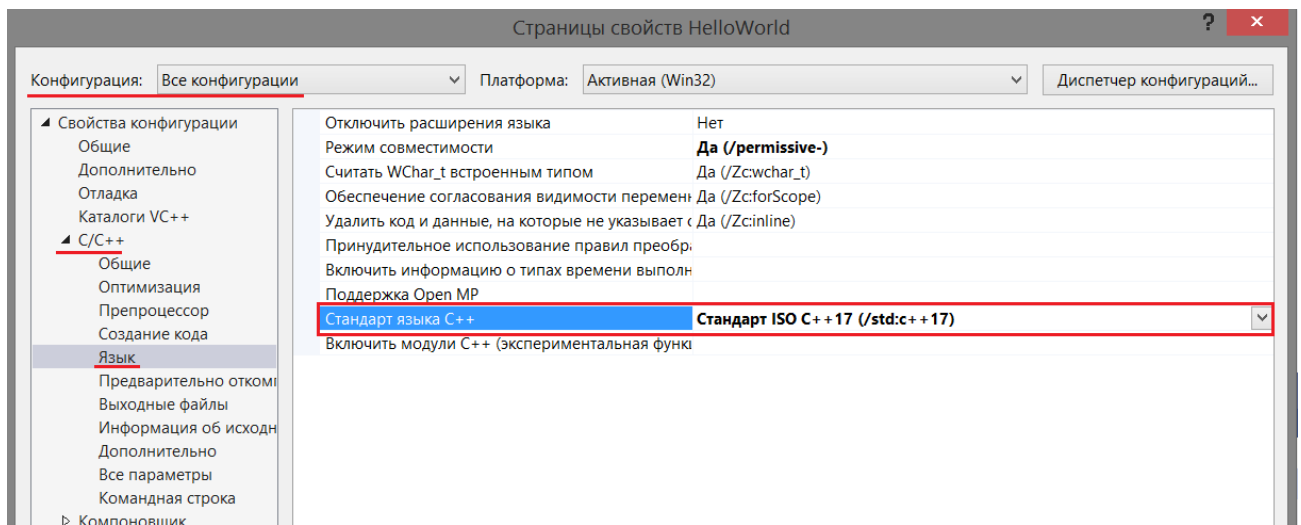
Структура языков программирования:



Язык программирования существует в нескольких видах:

<p>Стандарт языка</p> <p>набор спецификаций, определяющих его синтаксис, семантику, может исторически развиваться.</p>	<p>Стандарты языка C:</p> <p>1978 Kernighan, Ritchie (K&R)</p> <p>1989 ANSI C (C89)</p> <p>1999 C99</p> <p>2011 C11</p>
<p>Реализация языка</p> <p>программные средства, которые обеспечивают определенный вариант стандарта языка (производитель, марка, версия; могут иметь ошибки)</p>	 <p> ■ описано в стандарте ■ реализован в системе программирования </p>
<p>Способы реализации языков</p> <p>компилируемые</p> <p>интерпретируемые</p>	 <p> Компилируются в машинный код Компилируются в байт-код Компилируются в байт-код неважно Интерпретируются </p> <p> C, C++, Fortran Java, C# Python Perl, Bash </p> <p> ← Компилируемые Интерпретируемые → </p>

Изменить/подключить языковой стандарт C++ в Visual Studio можно на странице свойств проекта:



2. Алфавит языка программирования:

Алфавит языка программирования – набор символов, разрешенных к использованию языком программирования. Основывается на одной из кодировок.

Совокупность символов, используемых в языке – алфавит языка.

Базовый набор символов исходного кода:

- 1) строчные и прописные буквы латинского и национального алфавитов
- 2) цифры
- 3) знаки операций
- 4) символы подчеркивания _ и пробельные символы
- 5) ограничители и разделители
- 6) специальные символы

С помощью символов алфавита записываются **служебные слова**, которые составляют словарь языка.

3. Компилятор:

Компилятор – программа, преобразующая исходный код на одном языке программирования в исходный код на другом языке;
результат – объектный модуль.

Символы времени трансляции, символы времени выполнения:

Набор символов времени трансляции:	текст программы на языке программирования хранится в исходных файлах и основан на определенной кодировке символов
Набор символов времени выполнения:	символы, отображаемыми в среде выполнения. Любые дополнительные символы зависят от локализации

4. Компилятор CL:

исходный код C++ на ASCII, Windows-1251.

Стандарт C++: исходной код основывается на множестве символов ASCII:

Алфавит языка C++	буквы латинского алфавита: [a...z], [A...Z]; цифры [0...9]; спецсимволы: _ {} [] () # < > . : ; % . ? * + - / ^ & ~ ! = , ' " @ \$ пробельные символы: пробел, символы табуляции, символы перехода на новую строку.
--------------------------	---

Дополнительные символы **времени выполнения** определяются **setlocale**.

По умолчанию, локаль: **SetLocale (LC_ALL, "C")** устанавливает стандартный контекст C.

Во время выполнения можно изменить или запросить кодовую страницу языкового стандарта, используя вызов **setlocale**.

Директива **#pragma** позволяет указать целевой языковой стандарт во время компиляции. Это гарантирует, что строки с расширенными символами будут сохраняться в правильном формате.

Алфавит служит для построения слов в языке программирования, которые называют лексемами. Примеры лексем:

Лексемы	<i>идентификаторы; ключевые (зарезервированные) слова; знаки операций; константы; разделители (скобки, знаки операций, точка, запятая, пробельные символы и т.д.).</i>
----------------	--

Границы лексем определяются с помощью других лексем, таких, как разделители или знаки операций.

5. Идентификатор:

Идентификатор — имя компонента программы (переменной, функции, метки, типа и пр.), составленное программистом по определенным правилам.

Примеры правил составления идентификаторов в языках программирования:

C/C++	<p>начинаются с буквы или подчеркивания;</p> <p>не совпадают с ключевыми словами C++ или с именами библиотечных функций;</p> <p>могут состоять из любого количества символов, но компилятор гарантирует, что будет считать значащими только 31 первых символов идентификаторов, не имеющих внешней связи;</p> <p>не более 6 значащих символов идентификаторов с внешней связью;</p> <p>идентификаторы чувствительны к регистру.</p> <p>Длина идентификатора по стандарту не ограничена.</p>
Ruby	<p>начинаются с буквы или специального модификатора. имена локальных переменных начинаются со строчной буквы или знака подчеркивания (alpha, _ident);</p> <p>имена глобальных переменных начинаются со знака доллара (\$beta);</p> <p>имена переменных экземпляра (принадлежащих объекту) начинаются со знака «@» (@foobar);</p> <p>имена переменных класса (принадлежащих классу) предваряются двумя знаками «@@» (@@not_const);</p> <p>имена констант начинаются с прописной буквы (K6chip);</p> <p>в именах идентификаторов знак подчеркивания «_» можно использовать наравне со строчными буквами (\$not_const);</p> <p>имена специальных переменных, начинающиеся со знака «\$» (\$beta).</p>
MS Transact-SQL	<p>имена переменных должны начинаться с символа @</p>

Python

используются символы Unicode.

начинаются с латинской буквы в любом регистре или символа подчёркивания, могут содержать цифры.

не совпадают с ключевыми словами (их список можно узнать по `import keyword; print(keyword.kwlist)`), нежелательно переопределять встроенные имена.

Имена, начинающиеся с символа подчёркивания, имеют специальное значение.

Идентификатор создается при объявлении переменной, функции, типа и т. п.

6. Основные этапы разработки программ

Программа — логически упорядоченная последовательность команд, необходимых для решения определенной задачи.

Программа — алгоритм, записанный на языке программирования.

Текст программы — полное законченное и детальное описание алгоритма на языке программирования.

Этапы и цели разработки программы:

1. Постановка задачи. <ul style="list-style-type: none">• определение функциональных возможностей программы;• подготовка технического задания
2. Выбор метода решения. <ul style="list-style-type: none">• определение исходных и выходных данных, ограничений на них;• выполнение формализованного описания задачи;• построение математической модели, для решения на компьютере.
3. Разработка алгоритма решения задачи. <ul style="list-style-type: none">• выполняется на основе ее математического описания;• полное и точное описание, определяющее вычислительный процесс, ведущий от начальных данных к искомому результату.
4. Написание программы на языке программирования (кодирование) <ul style="list-style-type: none">• запись алгоритма на языке программирования.
5. Ввод программы в компьютер <ul style="list-style-type: none">• подготовка исходного кода программы в виде текстового, который поступает на вход транслятора.
6. Трансляция <ul style="list-style-type: none">• преобразование исходного кода с одного языка программирования в семантически эквивалентный код на другом языке;• получение объектного модуля.
7. Компоновка <ul style="list-style-type: none">• объединение одного или нескольких объектных модулей программы и объектных модулей статических библиотек в исполняемую программу;• связывание вызовов функций и их внутреннего представления (кодов), расположенных в различных модулях;• получение исполняемого (загрузочного) файла.
8. Выполнение <ul style="list-style-type: none">• выполнение исполняемого файла программы на целевой машине.
9. Тестирование <ul style="list-style-type: none">• устранение ошибок в программе.
10. Отладка <ul style="list-style-type: none">• обнаружение, локализация и устранение ошибок.
11. Документирование <ul style="list-style-type: none">• создание пользовательской документации.
12. Эксплуатация <ul style="list-style-type: none">• выполнение в предназначенной для этого среде в соответствии с пользовательской документацией
13. Модификация (Реинжиниринг) <ul style="list-style-type: none">• внесение изменений в ПО в целях исправления ошибок, повышения производительности или адаптации к изменившимся условиям работы или требованиям.
14. Снятие с эксплуатации <ul style="list-style-type: none">• завершение жизненного цикла ПП и изъятие его из эксплуатации.

7. Трудоемкость этапов

Этапы	Трудозатраты	Ошибки	
		Появление	Выявление
Постановка задачи	10%	40-46%	50%
Математическая формулировка			
Выбор метода решения			
Составление алгоритма	20%	35-38%	
Написание программы на языке программирования	15%		
Ввод программы в компьютер	5%	5-10%	
Выполнение программы			
Тестирование	40%		45%
Отладка			
Документирование	10%		3%
Эксплуатация			
Реинжиниринг			