

## Введение в разработку программного обеспечения (ИС и ЦД)

### Основные этапы разработки программ. Этапы разработки программы.

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

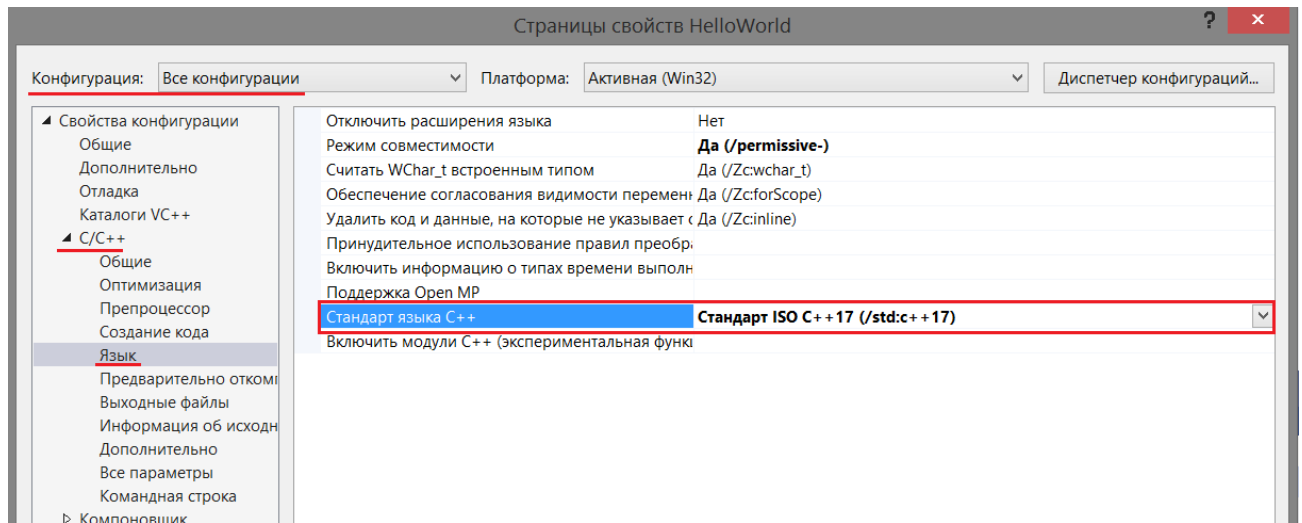
## Структура языков программирования:



## Язык программирования существует в нескольких видах:

<h3>Стандарт языка</h3> <p>набор спецификаций, определяющих его синтаксис, семантику, может исторически развиваться.</p>	<h3>Стандарты языка C:</h3> <p>1978 Kernighan, Ritchie (K&amp;R) 1989 ANSI C (C89) 1999 C99 2011 C11</p>								
<h3>Реализация языка</h3> <p>программные средства, которые обеспечивают определенный вариант стандарта языка (<i>производитель, марка, версия; могут иметь ошибки</i>)</p>	 <p>■ описано в стандарте ■ реализован в системе программирования</p>								
<h3>Способы реализации языков</h3> <p>компилируемые интерпретируемые</p>	<table><tr><td>Компилируются в машинный код</td><td>Компилируются в байт-код</td><td>Компилируются в байт-код неявно</td><td>Интерпретируются</td></tr><tr><td>C, C++, Fortran</td><td>Java, C#</td><td>Python</td><td>Perl, Bash</td></tr></table> <p>← Компилируемые Интерпретируемые →</p>	Компилируются в машинный код	Компилируются в байт-код	Компилируются в байт-код неявно	Интерпретируются	C, C++, Fortran	Java, C#	Python	Perl, Bash
Компилируются в машинный код	Компилируются в байт-код	Компилируются в байт-код неявно	Интерпретируются						
C, C++, Fortran	Java, C#	Python	Perl, Bash						

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++	<b>буквы латинского алфавита:</b> [a...z], [A...Z]; цифры [0...9]; <b>символы:</b> _{}[]()#<>:;%.*+~!=",'" @ \$ <b>пробельные символы:</b> 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	<b>идентификаторы;</b> <b>ключевые (зарезервированные) слова;</b> <b>знаки операций;</b> <b>константы;</b> <b>разделители (скобки, знаки операций, точка, запятая, пробельные символы и т.д.).</b>
---------	--

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:

<b>C/C++</b>	<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>
<b>Ruby</b>	<p>начинаются с буквы или специального модификатора: имена локальных переменных начинаются со строчной буквы или знака подчеркивания (<b>alpha</b>, <b>_ident</b>); имена глобальных переменных начинаются со знака доллара (<b>\$beta</b>); имена переменных экземпляра (принадлежащих объекту) начинаются со знака «@» (<b>@foobar</b>); имена переменных класса (принадлежащих классу) предваряются двумя знаками «@» (<b>@@not_const</b>); имена констант начинаются с прописной буквы (<b>K6chip</b>);</p> <p>в именах идентификаторов знак подчеркивания «_» можно использовать наравне со строчными буквами (<b>\$not_const</b>); имена специальных переменных, начинающиеся со знака «\$» (<b>\$beta</b>).</p>
<b>MS Transact-SQL</b>	имена переменных должны начинаться с символа @

## 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.

- определение функциональных возможностей программы;
- подготовка технического задания

### 2. Выбор метода решения.

- определение исходных и выходных данных, ограничений на них;
- выполнение формализованного описания задачи;
- построение математической модели, для решения на компьютере.

### 3. Разработка алгоритма решения задачи.

- 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)

- запись алгоритма на языке программирования.

### 5. Ввод программы в компьютер

- подготовка исходного кода программы в виде текстового, который поступает на вход транслятора.

### 6. Трансляция

- преобразование исходного кода с одного языка программирования в семантически эквивалентный код на другом языке;

• получение объектного модуля.
<b>7. Компоновка</b>
<ul style="list-style-type: none"> <li>• объединение одного или нескольких объектных модулей программы и объектных модулей статических библиотек в исполняемую программу;</li> <li>• связывание вызовов функций и их внутреннего представления (кодов), расположенных в различных модулях;</li> <li>• получение исполняемого (загрузочного) файла.</li> </ul>
<b>8. Выполнение</b>
• выполнение исполняемого файла программы на целевой машине.
<b>9. Тестирование</b>
• устранение ошибок в программе.
<b>10. Отладка</b>
• обнаружение, локализация и устранение ошибок.
<b>11. Документирование</b>
• создание пользовательской документации.
<b>12. Эксплуатация</b>
• выполнение в предназначенной для этого среде в соответствии с пользовательской документацией
<b>13. Модификация (Реинжиниринг)</b>
• внесение изменений в ПО в целях исправления ошибок, повышения производительности или адаптации к изменившимся условиям работы или требованиям.
<b>14. Снятие с эксплуатации</b>
• 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			

## Структура языка программирования

- ✓ **алфавит языка:** кодировка символов; символы времени трансляции, символы времени выполнения;
- ✓ **идентификаторы:** правила образования идентификаторов; зарезервированные идентификаторы; литералы; ключевые слова;
- ✓ **фундаментальные (встроенные) и пользовательские типы данных:**
  - предопределенные типы данных, массивы фундаментальных типов;
  - типы, которые может создавать пользователь на основе фундаментальных типов (возможно описание их свойств и поведение);
- ✓ **преобразование типов:** явное и неявное (автоматическое).
- ✓ **инициализация памяти:** присвоение значения в момент объявления переменной;
- ✓ **константное выражение:** выражение, которое должно быть вычислено на этапе компиляции;
- ✓ **область видимости переменных:** доступность переменных по их идентификатору в разных частях программы; пространства имен;
- ✓ **выражения**
- ✓ **инструкции языка:** инструкция — это некое элементарное действие, несколько идущих подряд инструкций образуют блок вычислений (последовательность инструкций);
  - присваивания;
  - инструкции объявления;
  - блок вычислений;
  - ветвление;
  - циклы;
  - инструкции перехода;
  - обработка исключений;
- ✓ **программные конструкции** (декомпозиция программного кода): процедуры, функции, методы, ...

## 8. Основные этапы разработки программ

### 1) Постановка задачи (ответственность исполнителя).

#### Постановка задачи

*точная формулировка условий задачи с описанием входной и выходной информации, описание поведения программы в особых случаях*



<p><b>Описание входной информации</b></p>	<p><b>точное описание всех исходных данных, которые вводятся пользователем</b></p> <ul style="list-style-type: none"> <li>○ <b>синтаксис (формат данных);</b></li> <li>○ <b>семантика (назначение, тип, допустимые значения, область изменения, ...)</b></li> </ul>
<p><b>Описание выходной информации</b></p>	<p><b>точное описание результатов, формируемых программой</b></p> <ul style="list-style-type: none"> <li>○ <b>синтаксис и семантика выходных данных;</b></li> <li>○ <b>сообщений об ошибках;</b></li> <li>○ <b>протокол вычислительного процесса;</b></li> <li>○ <b>реакция программы на некорректность исходных данных;</b></li> <li>○ <b>и т.п.</b></li> </ul>
<p><b>Дополнительные сведения о программе</b></p>	<p><b>ограничения:</b></p> <ul style="list-style-type: none"> <li>○ <b>на используемую память;</b></li> <li>○ <b>длину программы;</b></li> <li>○ <b>время ее работы;</b></li> </ul> <p><b>идеи относительно внутреннего проектирования функций (если это необходимо);</b></p> <p><b>описание функций преобразования информации, выполняемых программой.</b></p>

### Пример

**Цель:** ознакомиться с основами кодирования информации;  
освоить кодировки **ASCII, Windows-1251**.

**Среда разработки:** создать приложение на языке программирования C++ в интегрированной среде разработки Visual Studio.

**Задача:** по коду символа, введенного с клавиатуры, определить, является этот символ цифрой, буквой латинского либо русского алфавита или другим символом.

Вывести в консоль символ, информацию о принадлежности символа к одной из категорий, его код в соответствующей кодировке ASCII или Windows- 1251.

**Входная информация:** программа принимает один символ из стандартного входного потока.

**Выходная информация:** выводит в стандартный поток вывода введенный символ, категорию, к которой он принадлежит, и код этого символа с указанием соответствующей кодировки.

## 2) Формализация задачи.

На этом этапе создается описательная информационная модель, созданная на этапе постановки задачи, выраженная каким-либо формальным языком, например, математическими формулами, адаптированными для решения данной задачи.

## 3) Разработка алгоритма решения задачи

**Алгоритм** (лат. *algorithmi*) – от имени Аль-Хорезми, узбекского математика, астронома, IX в.) – совокупность точно заданных правил, с помощью которой можно получить решение задачи за конечное число шагов.

<b>Алгоритм</b>	<i>точное предписание, определяющее вычислительный процесс, ведущий от начальных данных к искомому результату</i>
-----------------	---

Кнут Д.Э. Искусство программирования. Том 1. Основные алгоритмы, 2006г.

<b>Алгоритм</b>	<i>конечный набор правил, который определяет последовательность операций для решения конкретного множества задач и обладает пятью важными чертами: конечность, определённость, ввод, вывод, эффективность.</i>
-----------------	--

Колмогоров А.Н. Теория информации и теория алгоритмов. Изд. 1987г.

<b>Алгоритм</b>	<i>всякая система вычислений, выполняемых по строго определённым правилам, которая после какого-либо числа шагов заведомо приводит к решению поставленной задачи.</i>
-----------------	---

Марков А.А. Теория алгоритмов. (1954г.) Изд. 1984г.

<b>Алгоритм</b>	<i>точное предписание, определяющее вычислительный процесс, идущий от варьируемых исходных данных к искомому результату.</i>
-----------------	--

ГОСТ 19.004–80

<b>Программа</b>	<i>алгоритм, записанный в форме, воспринимаемой вычислительной машиной.</i>
------------------	---

<b>Программирование</b>	<i>это также раздел прикладной математики, разрабатывающий методы использования вычислительных машин для реализации алгоритмов</i>
<b>Свойства алгоритмов</b>	<p><i>дискретность</i> (возможность разбиения на шаги);</p> <p><i>понятность</i> (ориентирован на исполнителя);</p> <p><i>определенность</i> (однозначность толкования инструкций);</p> <p><i>конечность</i> (возможность получения результата за конечное число шагов);</p> <p><i>массовость</i> (применимость к некоторому классу объектов);</p> <p><i>эффективность</i> (оптимальность времени и ресурсов, необходимых для реализации алгоритма).</p>
<b>Процесс алгоритмизации</b>	<p><i>разложение всего вычислительного процесса на отдельные шаги</i></p> <p><i>установление взаимосвязей между отдельными шагами алгоритма и порядка их следования</i></p> <p><i>полное и точное описание содержания каждого шага</i></p> <p><i>проверка правильности составленного алгоритма</i></p>
<b>Способы описания алгоритмов</b>	<p><i>словесно-формульный</i> (на естественном языке);</p> <p><i>графический</i> (структурный или блок-схемой);</p> <p><i>использование псевдокода</i> (специальных алгоритмических языков);</p> <p><i>с помощью сетей Петри;</i></p> <p><i>программный.</i></p>

## Словесно-формульный способ

### Пример:

**Задача.** По коду символа, введенного с клавиатуры, определить, является этот символ цифрой, буквой латинского либо русского алфавита или другим символом. Вывести в консоль информацию, к какой категории символов он

принадлежит, и его код в соответствующей кодировке ASCII или Windows- 1251.

Словесно-формульным способом алгоритм решения этой задачи может быть записан в следующем виде:

1. Ввести символ
2. Если код символа попадает в диапазон от 30 в шестнадцатеричной системе счисления (0x30) до 39 в шестнадцатеричной системе счисления (0x39) включительно, то п.3, в противном случае п.5.
3. Вывести «Это цифра», символ цифры, ASCII, код символа в таблице ASCII.
4. Перейти к п.12 (конец).
5. Иначе: если код символа попадает в диапазон от 41 в шестнадцатеричной системе счисления (0x41) до 7A в шестнадцатеричной системе счисления (0x7A) включительно, то п.6, в противном случае п.8.
6. Вывести «Это латинская буква», символ буквы, ASCII, код символа в таблице ASCII.
7. Перейти к п.12 (конец).
8. Иначе: если код символа попадает в диапазон от 0xC0 до 0xFF включительно, то п.9 в противном случае п.11.
9. Вывести «Это русская буква», символ буквы, Windows- 1251, код символа в таблице Windows- 1251.
10. Перейти к п.12 (конец).
11. Вывести «Это не цифра и не буква», символ, код символа в таблице Windows- 1251
12. КОНЕЦ.

## Блок-схемы




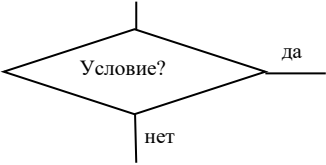
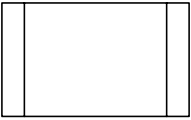
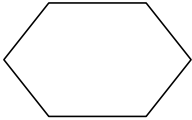


Выработаны соглашения для изображения схем-алгоритмов и закреплены ГОСТ и международными стандартами.

Единая система программной документации (ЕСПД), частью которой является Государственный стандарт — **ГОСТ 19.701-90 «Схемы алгоритмов программ, данных и систем».**

Рассматриваемый ГОСТ 19.701-90 практически полностью соответствует международному стандарту ISO 5807:1985.

Условные графические изображения, используемые для составления блок-схем, называют символами.

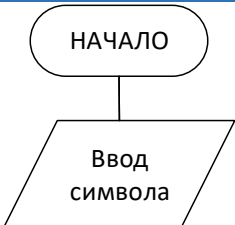
## Основные элементы схем алгоритма

 <p>Блок начала-конца алгоритма</p>	 <p>Блок ввода-вывода данных</p>	 <p>Блок вычислений</p>	 <p>Условный блок</p>
 <p>Предопределенный процесс</p>	 <p>Блок подготовки ()</p>	 <p>Комментарий</p>	 <p>Соединитель (ссылка на текущую страницу при разрыве схемы)</p>

## Типы процессов

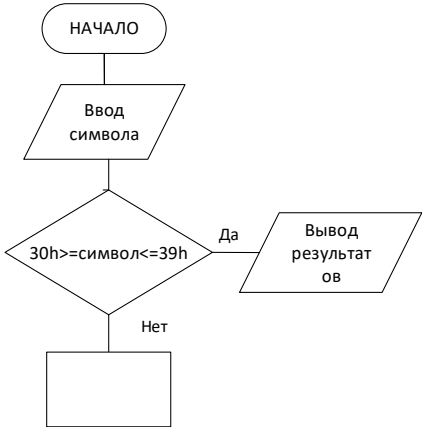
### Линейные процессы.

направление вычислений не зависит от значения исходных данных и получаемых в результате решения задачи промежуточных результатов.

1. Ввести символ		<pre>cout &lt;&lt; "Введите символ "; cin &gt;&gt; code;</pre>
------------------	---	--

### Разветвляющиеся процессы.

вычислительные процессы, в которых в зависимости от значения некоторого признака проводятся вычисления по одному из нескольких возможных направлений, называются ветвящимися (разветвляющимися).

2. Если код символа попадает в диапазон от 30 <sub>16</sub> до 39 <sub>16</sub> включительно, то п.3, в противном случае п.5.		<pre>if (code &gt;= '0' &amp;&amp; code &lt;= '9') {     printf("Это цифра %c, код ASCII = %X", code, code);     cout &lt;&lt; endl; } else</pre>
---	---	---

## 4) Кодирование

### **Кодирование:**

запись разработанного алгоритма в виде программы на выбранном языке программирования.

Результатом этапа является исходный код программы на ЯП.

**Система программирования:**  
комплекс программных средств, предназначенных для автоматизации процесса разработки, отладки ПО и подготовки программного кода к выполнению



```
#include <windows.h>
#include <iostream>

int main()
{
    SetConsoleOutputCP(1251);
    SetConsoleCP(1251);
    using namespace std;
    unsigned char code;
    cout << "Введите символ ";
    cin >> code;
    if (code >= '0' && code <= '9')
    {
        printf("Это цифра %c, код ASCII = %X", code, code);
        cout << endl;
    }
    else if (code >= 'A' && code <= 'z')
    {
        printf("Это латинская буква %c, код ASCII = %X", code, code);
        cout << endl;
    }
    else if (code >= 0xC0 && code <= 0xFF)
    {
        printf("Это русская буква %c, код Windows-1251 = %X", code, code);
        cout << endl;
    }
    else
    {
        printf("Это не цифра и не буква <%c>, код = %X", code, code);
        cout << endl;
    }
    system("pause");
    return 0;
}
```

## 5) Ввод программы в компьютер

### **Инструментальные средства программирования:**

#### **Текстовый редактор**

*компонента системы программирования (или IDE) – программа, позволяющая подготовить исходный код программы*

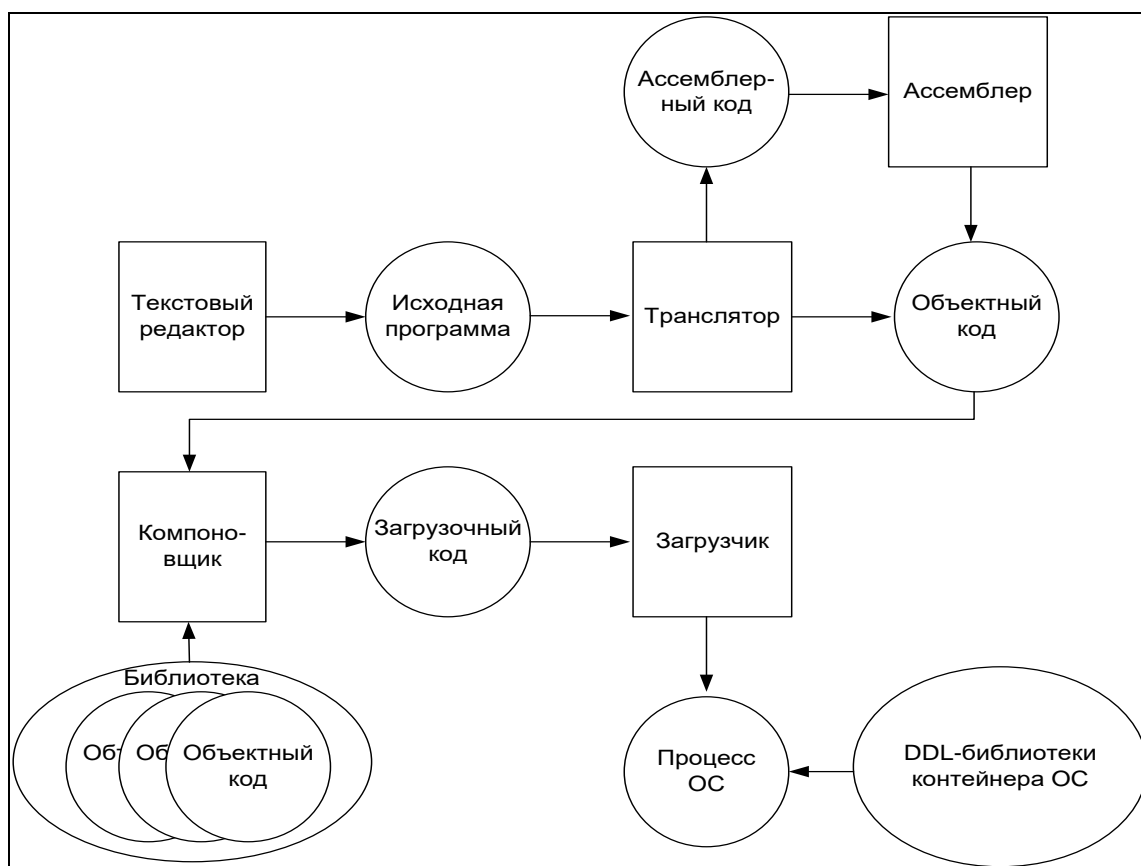
#### **Интегрированная среда разработки** (integrated development environment - IDE)

*набор инструментов для разработки и отладки программ, имеющий общую интерактивную графическую оболочку, поддерживающую выполнение всех основных функций жизненного цикла разработки программы*

### **Примеры IDE (визуальные среды):**

Eclipse, Microsoft Visual Studio, NetBeans, Qt Creator, ...

## Общая схема системы программирования:

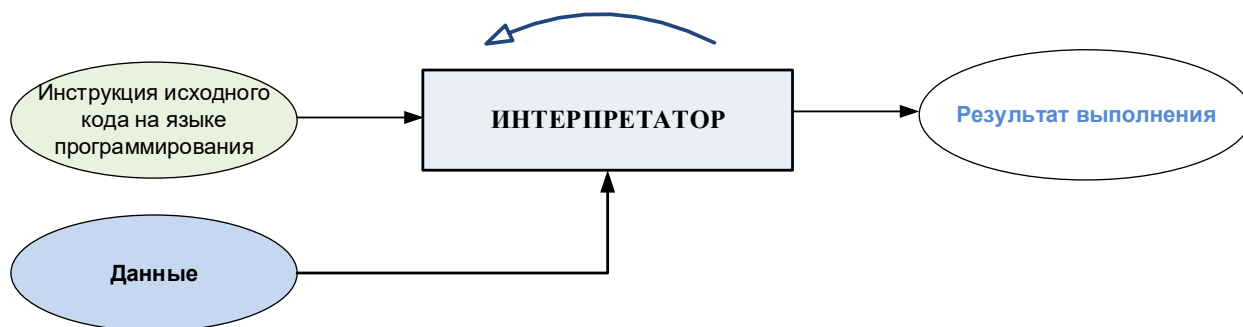


## 6) Компиляция

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

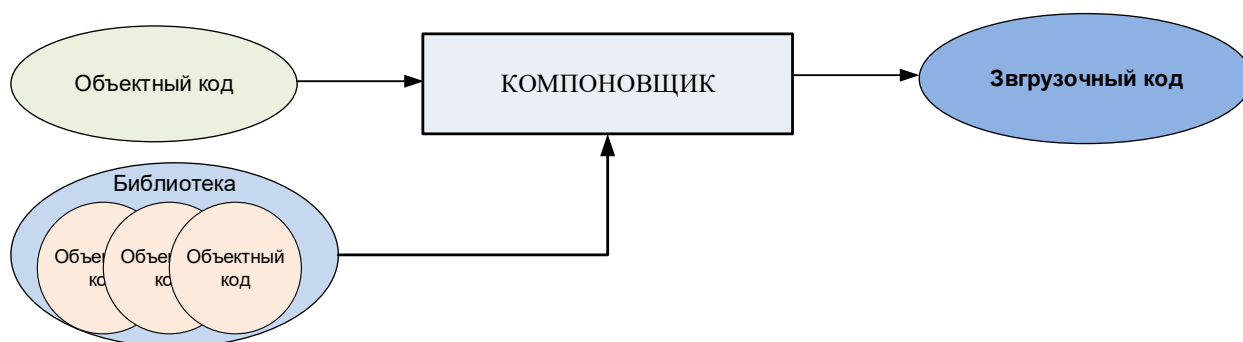


**Интерпретатор** – разновидность транслятора. Переводит и выполняет программу с языка высокого уровня в машинный код строка за строкой.



## 7) Компоновка

**Компоновщик** (linker, редактор связей) – программа, принимающая один или несколько объектных модулей и формирующая на их основе загрузочный модуль.



Если программа состоит из нескольких объектных файлов, **компоновщик** собирает эти файлы в единый исполнимый модуль, вычисляя и подставляя адреса вместо неопределенных внешних имен, в течение **времени компоновки** (статическая компоновка) или во **время исполнения** (динамическая компоновка).

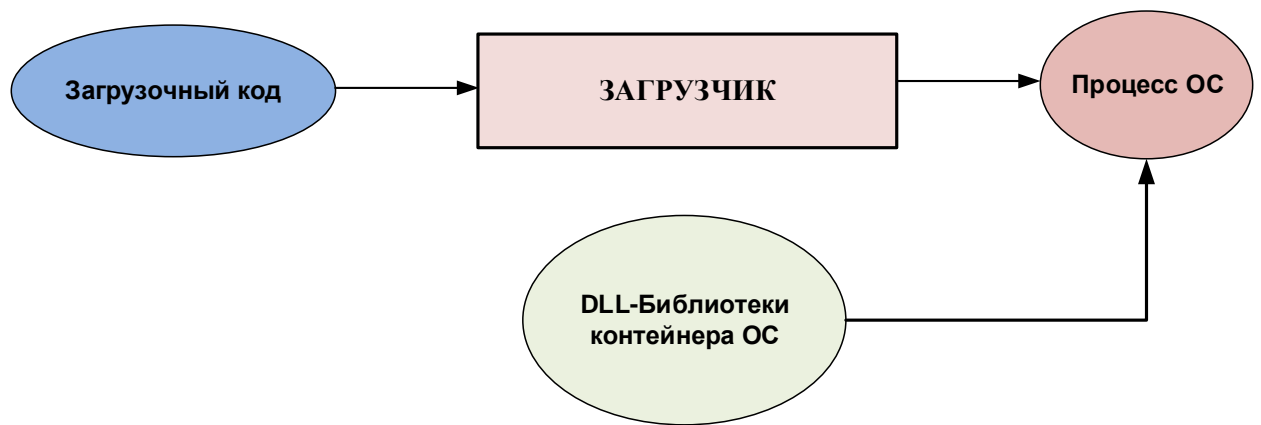
## 8) Выполнение исполняемого файла программы на целевой машине

**Загрузочный код** – результат работы компоновщика.

Один файл загрузочного кода – загрузочный модуль.

**Загрузчик** (loader) – программа, обычно входящая в состав операционной системы, предназначенная для запуска процесса операционной системы на основе загрузочного модуля.





## 9) Отладка программы

**Отладка программы** – процесс поиска, локализации и устранения ошибок в программе.

**Отладчик** (debugger) – компонента системы программирования (или IDE) – программа, позволяющая контролировать ход выполнения программы (приостанавливать, выполнять пошагово), просматривать и изменять области памяти и т.п.

Ошибки в программе можно разделить на три группы:

- синтаксические (ошибки в исходном коде программы);
- времени выполнения (выявляются на этапе выполнения);
- алгоритмические.

Этап отладки можно считать законченным, если программа правильно работает на нескольких наборах входных данных.

**Программы-отладчики:**

Microsoft Visual Studio, GNU Debugger, DBX, WinDbg, TotalView.

## 10) Тестирование программы

**Тест** – это набор конкретных значений исходных данных, при которых известен ожидаемый результат работы программы.

На этапе тестирования и отладки проверяется, работает ли программа, если работает, то правильно ли. Проверяется отсутствие ошибок в программе.

С этой целью выполняется проверка поведения программы на большом количестве входных наборов данных, в том числе и наборах заведомо

неверных данных (учет ситуаций, для которых программа в принципе не предназначена).

Если результаты работы программы соответствуют ожидаемым – значит задача решена, иначе – на одном из этапов допущена ошибка.

На этапе тестирования и отладки требуются как знания по предметной области, так и знание основ программирования. Так как без знаний в предметной области мы не можем знать результирующих данных в тестах, а без знаний в программировании мы не сможем отыскать ошибки и составить наиболее полный набор тестов, учитывающий все частные случаи и исключения.

## **11) Документирование, поддержка и обновление программы**

**Документирование** – создание текстовых и графических материалов по использованию программы в помощь пользователям и разработчикам (общее описание возможностей программы, техники использования, типовые примеры и т.д.).

## **12) Эксплуатация**

Выполнение операций, выполняемых персоналом, эксплуатирующем систему в предназначенной для этого среде в соответствии с пользовательской документацией.

## **13) Модификация (Реинжиниринг)**

**Реинжиниринг** – это модификация программного продукта при необходимости исправить ошибки, выявленные в процессе эксплуатации, модернизировать или адаптировать программу к изменившимся требованиям.

Некоторые нештатные ситуации и ошибки проявляются только в процессе длительного использования программы с множеством входных данных. В этих случаях может потребоваться обновление кода программы.

## **14) Снятие с эксплуатации**

Завершение жизненного цикла ПП и изъятие его из эксплуатации