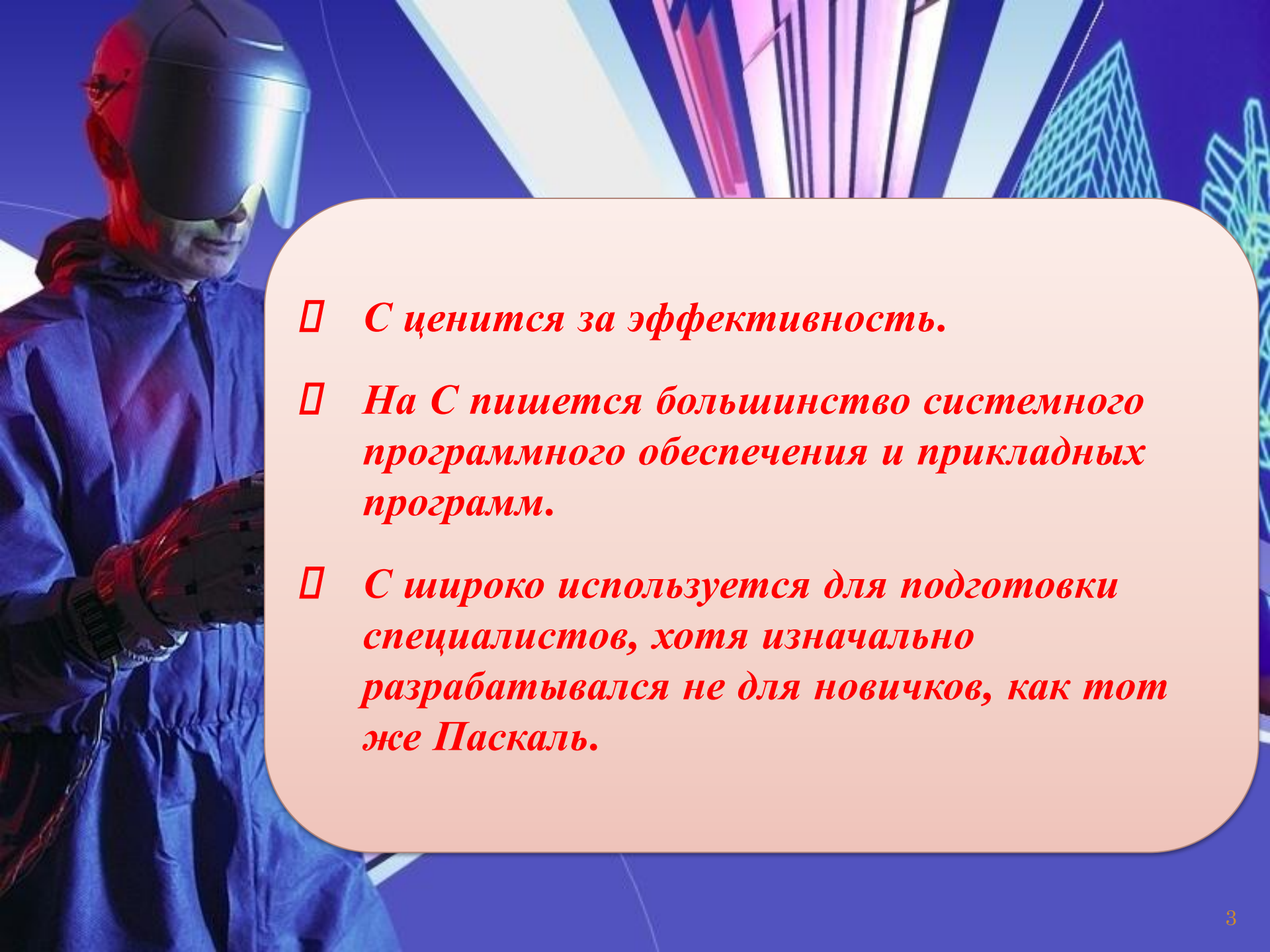
# АЛГОРИТМИЧЕСКИЙ ЯЗЫК ПРОГРАММИРОВАНИЯ С

*Написание хороших программ требует ума, вкуса и терпения.*
Б. Страуструп

# Общая характеристика языка С

- Язык С - это стандартизованный процедурный язык программирования, был разработан ещё в начале 70-х годов Кеном Томпсоном и Денисом Ритчи, сотрудниками компании Bell Labs и является продолжением языка B (читается "би").

- Язык С изначально был создан для программирования под операционную систему UNIX и в последствии был перенесён на множество других операционных систем и стал одним из самых популярных языков программирования.



- *С ценится за эффективность.*
- *На С пишется большинство системного программного обеспечения и прикладных программ.*
- *С широко используется для подготовки специалистов, хотя изначально разрабатывался не для новичков, как тот же Паскаль.*

# *ОБЩАЯ ХАРАКТЕРИСТИКА ЯЗЫКА С.*

*Изначально язык* **С** *был придуман, чтобы заменить Ассемблер в написании операционных систем.*

- *код* **С** *легко переносится с одной платформы на другую.*

*В настоящее время большинство ОС написано на* **С**.

- *но применение языка* **С** *не ограничивается только написанием операционных систем.*

**С** *имеет массу достоинств, он прост в изучении и лаконичен.*

- *элементы языка максимально приближены к архитектуре компьютеров.*

# ОБЩАЯ ХАРАКТЕРИСТИКА ЯЗЫКА С.

*Язык* **С** *позволяет программисту полностью контролировать компьютер, средствами самого же языка.*

- *Так, например, указатель - это адрес памяти, массив - непрерывная область памяти и т.д.*

*Многие языки взяли за основу синтаксис* **С**.

- *Для языка* **С** *характерны лаконичность, современный набор конструкций управления потоком выполнения, структур данных и большой набор операций.*

*На сегодняшний день* **С** *и языки основанные на синтаксисе* **С** *наиболее популярны в практическом программировании.*

# ОБЩАЯ ХАРАКТЕРИСТИКА ЯЗЫКА С.

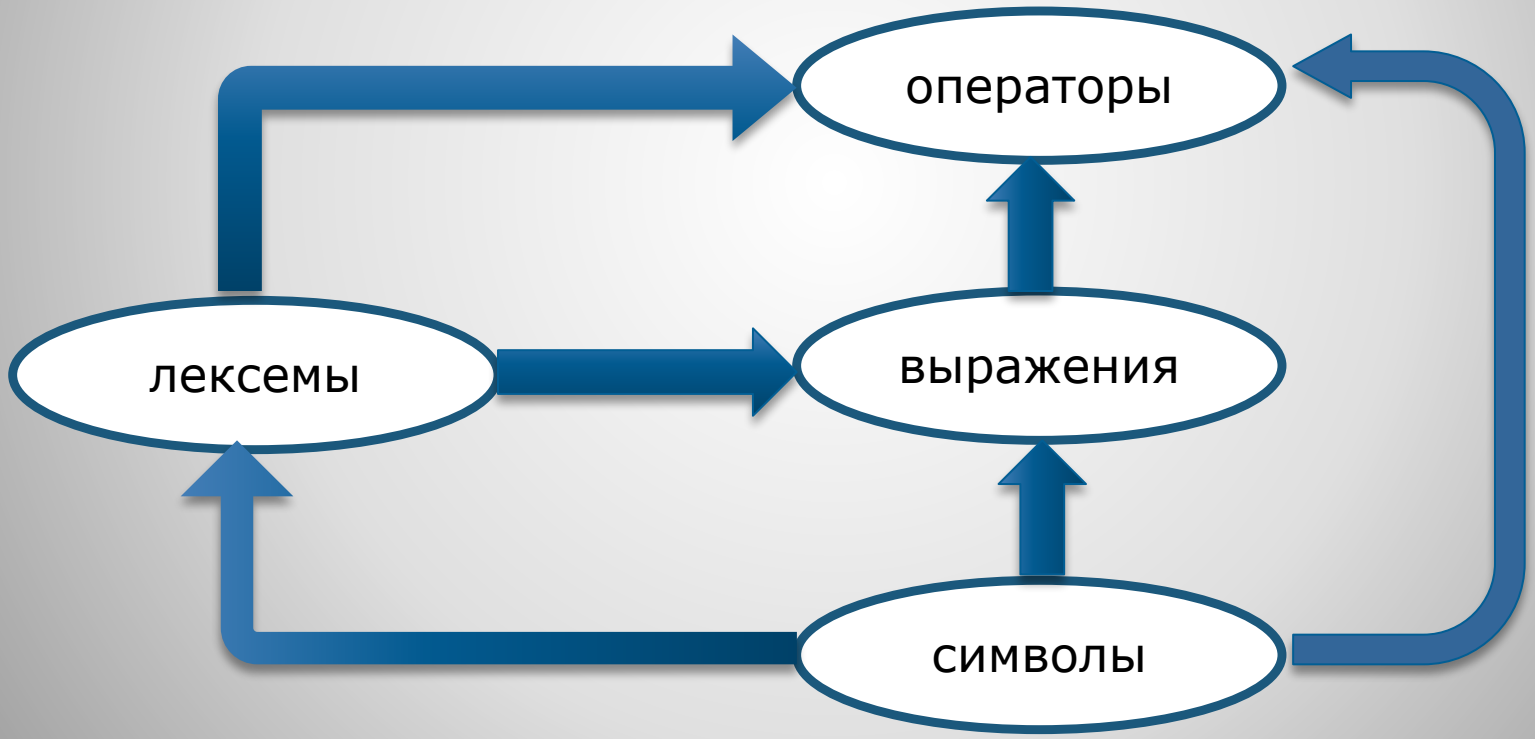
**С** *удобен для написания программ обработки текста и графики, в различных инженерных расчётах.*

*Объектно-ориентированные языки, основанные на* **С**, *хорошо подходят для программирования в оконных средах.*

# Состав языка

- В тексте на любом естественном языке можно выделить четыре основных элемента: символы, слова, словосочетания и предложения.
- Подобные элементы содержит и алгоритмический язык, только слова называют лексемами (элементарными конструкциями),

- лексемы образуются из символов,
- выражения — из лексем и символов,
- операторы — из символов, выражений и лексем




# Состав языка.

| | |
|---|---|
| Алфавит | •это основные неделимые знаки, с помощью которых пишутся все тексты на языке. |
| Лексема | •минимальная единица языка, имеющая самостоятельный смысл. |
| Выражение | •задает правило вычисления некоторого значения. |
| Оператор | •задает законченное описание некоторого действия. |

# Состав языка.

Операторы могут быть объединены в составной оператор, или блок.

- Блоком в языке С считается последовательность операторов, заключенная в фигурные скобки {}

Операторы бывают исполняемые и неисполняемые.

- Исполняемые операторы задают действия над данными.
- Неисполняемые операторы служат для описания данных, поэтому их часто называют операторами описания или просто описаниями.

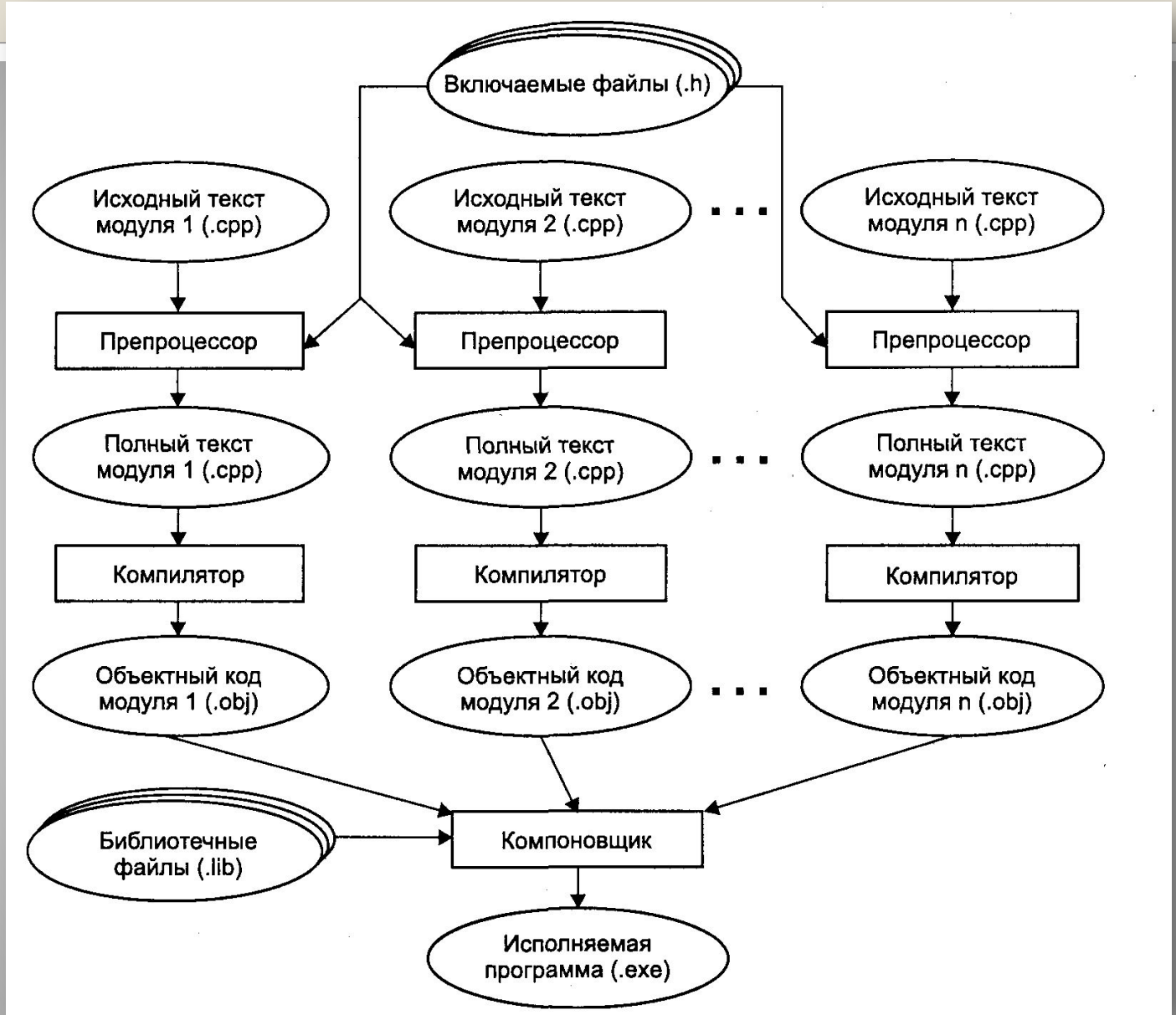
# Состав языка.

Для того чтобы выполнить программу, требуется перевести ее на язык, понятный процессору — в машинные коды.

- Этот процесс в языке **C** состоит из нескольких этапов

**Этапы создания исполняемой программы.**

# Этапы создания исполняемой программы.




# Этапы создания исполняемой программы.

# Алфавит языка

- прописные и строчные латинские буквы и знак подчеркивания;
- арабские цифры от 0 до 9;
- специальные знаки: " { } , . ! # % & [ ] ( ) + - / * : ? < = > ;

- Из символов алфавита формируются лексемы языка:
  - идентификаторы
  - ключевые слова
  - знаки операций
  - константы
  - разделители

Границы лексем определяются другими лексемами, такими, как разделители или знаки операций.

# Алфавит языка.

- **Идентификаторы**
  - Идентификатор — это имя программного объекта.
  - В идентификаторе могут использоваться латинские буквы, цифры и знак подчеркивания.
  - Прописные и строчные буквы **различаются**.
  - Первым символом идентификатора может быть буква или знак подчеркивания, но не цифра.
  - Пробелы внутри имен не допускаются.

*Длина **идентификатора** по стандарту не ограничена, но некоторые компиляторы и компоновщики налагают на нее ограничения.*

## Алфавит языка.

- **Ключевые слова**
  - это зарезервированные идентификаторы, которые имеют специальное значение для компилятора.
  - Их можно использовать только в том смысле, в котором они определены.
  - Интерпретация или компиляция ключевого слова всегда вызывает вполне определенное действие компьютера.

# Алфавит языка.

# Список ключевых слов C

| | | | |
|---|---|---|---|
| asm | else | new | this |
| auto | enum | operator | throw |
| bool | explicit | private | true |
| break | export | protected | try |
| case | extern | public | typedef |
| catch | false | register | typeid |
| char | float | reinterpret_cast | typename |
| class | for | return | union |
| const | friend | short | unsigned |
| const_cast | goto | signed | using |
| continue | if | sizeof | virtual |
| default | inline | static | void |
| delete | int | static_cast | volatile |
| do | long | struct | wchar_t |
| double | mutable | switch | while |
| dynamic_cast | namespace | template | |

**Ключевые слова.**

- **Знаки операций**
  - Знак операции — это один или более символов, определяющих действие над операндами
  - Внутри знака операции пробелы не допускаются.

# Алфавит языка.

Операции делятся на унарные, бинарные и тернарную

- по количеству участвующих в них операндов

Один и тот же знак может интерпретироваться по-разному

- в зависимости от контекста

Все знаки операций представляют собой отдельные лексемы

- за исключением [ ], ( ) и ? :

Большинство стандартных операций может быть переопределено (перегружено)

# Знаки операций.

- **Константы**
  - В языке **C** существуют целые, вещественные, символьные и строковые константы.
    - Компилятор, выделив константу в качестве лексемы, относит ее к одному из типов по ее внешнему виду.
    - Программист может задать тип константы самостоятельно.

## Алфавит языка.

# Форматы целых констант

| Формат | Описание | Примеры |
|---|---|---|
| Десятичный | последовательность десятичных цифр, начинающаяся не с нуля, если это не число нуль | 8, 0, 199226 |
| Восьмеричный | нуль, за которым следуют восьмеричные цифры (0,1,2,3,4,5,6,7) | 01, 020, 07155 |
| Шестнадцатеричный | 0x или 0X, за которым следуют шестнадцатеричные цифры (0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F) | 0xA, 0x1B8, 0X00FF |

**Целые константы.**

# Форматы вещественных констант

| Формат | Описание | Примеры |
|---|---|---|
| Десятичный | **[цифры].[цифры]**<br>может быть опущена<br>либо целая часть, либо дробная,<br>но не обе сразу. | 5.7, .001, 35. |
| Экспоненциальный | **[цифры][.][цифры]{E\|e}[+\|-][цифры]**<br>может быть опущена<br>либо целая часть, либо дробная,<br>но не обе сразу<br><br>если указаны обе части,<br>символ точки обязателен. | 0.2E6, .11e-3,<br>5E10 |

**Вещественные константы.**

# Форматы символьных и строковых констант

| Константа | Описание | Примеры |
|---|---|---|
| Символьная | Один или два символа, заключенных в апострофы | 'A', 'ю', '*', 'db', '\0', '\n', '\012', '\x07\x07' |
| Строковая | Последовательность символов, заключенная в кавычки | "Здесь был Vasia", "\tЗначение r=\0xF5\n" |

**Символьные и строковые константы.**

Символьные константы, состоящие из одного символа, занимают в памяти один байт и имеют стандартный тип **char**

Двухсимвольные константы занимают два байта и имеют тип **int**

- первый символ размещается в байте с меньшим адресом

Пустая символьная константа недопустима.

# Символьные константы.

# Символ обратной косой черты используется для представления:

кодов, не имеющих графического изображения

- например, \a — звуковой сигнал, \n — перевод курсора в начало следующей строки

символов апострофа ( ' ), обратной косой черты ( \ ), знака вопроса (?) и кавычки ( " )

любого символа с помощью его шестнадцатеричного или восьмеричного кода

- например, \073,\0xF5
- значение должно находиться в диапазоне от 0 до 255

**Символьные константы.**

Последовательности символов, начинающиеся с обратной косой черты, называют *управляющими,* или *escape-последовательностями*

- Управляющая последовательность интерпретируется как одиночный символ.
- Если непосредственно за обратной косой чертой следует символ, не предусмотренный таблице, результат интерпретации не определен.
- Если в последовательности цифр встречается недопустимая, она считается концом цифрового кода.

## Символьные константы.

| Символ | Код | Наименование |
|---|---|---|
| \a | 7 | звуковой сигнал |
| \b | 8 | возврат на шаг |
| \f | C | перевод страницы (формата) |
| \n | A | перевод строки |
| \r | D | возврат каретки |
| \t | 9 | горизонтальная табуляция |
| \v | B | вертикальная табуляция |
| \\ | 5C | обратная косая черта |
| \’ | 27 | апостроф |
| \" | 22 | кавычка |
| \? | 3F | вопросительный знак |
| \0ddd | - | восьмеричный код символа |
| \0xddd | ddd | шестнадцатеричный код символа |

**Допустимые значения *escape-последовательностей*.**

Управляющие последовательности могут использоваться и в строковых константах, называемых иначе *строковыми литералами*.

- Например, если внутри строки требуется записать кавычку, ее предваряют косой чертой, по которой компилятор отличает ее от кавычки, ограничивающей строку: **"Фирма \"IVK\""**

Все строковые литералы рассматриваются компилятором как различные объекты.

# Строковые константы.

Строковые константы, отделенные в программе только пробельными символами, при компиляции объединяются в одну.

- Длинную строковую константу можно разместить на нескольких строках, используя в качестве знака переноса обратную косую черту, за которой следует перевод строки.
- Эти символы игнорируются компилятором, при этом следующая строка воспринимается как продолжение предыдущей.

# Строковые константы.

Например, строка

**"Никто не доволен своей \ внешностью, но все довольны \ своим умом"**

полностью эквивалентна строке

**"Никто не доволен своей внешностью, но все довольны своим умом"**

## Строковые константы.

В конец каждого строкового литерала компилятором добавляется нулевой символ, представляемый управляющей последовательностью \0.

- Поэтому длина строки всегда на единицу больше количества символов в ее записи.
- Таким образом, пустая строка "" имеет длину 1 байт.

Обратите внимание на разницу между строкой из одного символа и символьной константой.

- например, "A" и 'A'.

# Строковые константы.

- **Комментарии**
  - Комментарий либо начинается с двух символов «косая черта» (**//**) и заканчивается символом перехода на новую строку, либо заключается между символами-скобками **/*** и ***/**.
  - Внутри комментария можно использовать любые допустимые на данном компьютере символы, а не только символы из алфавита языка **C**, поскольку компилятор комментарии игнорирует.
  - Вложенные комментарии-скобки
- **стандартом не допускаются**,
  - хотя в некоторых компиляторах разрешены.

# Алфавит языка.

***Рекомендуется использовать для пояснений //-комментарии,***

***а скобки /* */ применять для временного исключения блоков кода при отладке.***

## Комментарии.