Лекция 5

Способы описания синтаксиса языка. История создания C и C++. Структура и конструкция программы на Си/C++

В.Бондарев

Состав языка

В тексте на любом естественном языке можно выделить 4 основных элемента: символы, слова, словосочетания и предложения. Подобные элементы содержит и алгоритмический язык, только слова называют *лексемами*, словосочетания – выражениями, а предложения – операторами. Лексемы образуются из символов, выражения – из лексем и символом, а операторы – из символов, выражений и лексем.

Лексема – минимальная единица языка, имеющая самостоятельный смысл.

Операторы бывают исполняемые (действие) и неисполняемые (объявление).

Синтаксис языка определяет правила построения элементов языка, а семантика языка определяет смысл и правила использования.

Программа на алгоритмическом языке – это совокупность описаний данных и операторов для обработки этих данных (исходный код программы). Для выполнения программы требуется перевести исходный код в машинный код – язык, понятный процессору.

Создание исполняемого файла

Рассмотрим кратко, что происходит при преобразовании исходного кода в исполняемый файл.

Исходный файл передается компилятору, который проверяет синтаксис и, если не обнаружил синтаксических ошибок, создает объектный файл – это почти готовый машинный код, в котором еще не определены конкретные адреса. Затем объектные файлы и системные библиотеки компоновщиком преобразуются в исполняемый файл (.exe для windows).

Способы описания синтаксиса языка

1. Металингвистические формулы Бэкуса-Наура

Металингвистическая формула Бэкуса-Наура (**БНФ**) записывается в виде

<A> ::= X ,

где A – название определяемой синтаксической конструкции;

X – последовательность символов языка, представляющая конструкцию;

::= – знак “есть по определению”.

В БНФ используют также **дополнительные знаки**:

| – обозначает связку «ИЛИ»;

() – круглые скобки, обозначают «И»;

{<X>} – обозначает неограниченное повторение конструкции X;

[<X>] – необязательность конструкции X.

Металингвистические формулы Бэкуса-Наура

Примеры БНФ:

<цифра>::=0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<двоичная цифра> ::= 0 | 1

<двоичный код> ::= <двоичная цифра> | <двоичный код> <дв. цифра>

Формула, в которой определяемое понятие находится в правой части, т.е. определяется через это же понятие, называется **рекурсивной.**

<двоичный код> ::= <двоичная цифра> | {<двоичная цифра>}

Способы описания синтаксиса языка

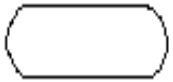
2. Синтаксические диаграммы

В этом случае синтаксические конструкции языка изображают в виде **графических схем.**

На синтаксических диаграммах применяют следующие виды графических элементов:

1) линии

2) прямоугольники со скругленными сторонами – обозначают основные символы языка

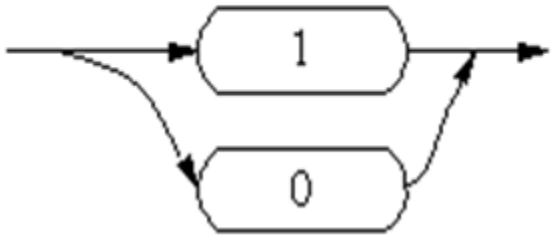


3) прямоугольники – обозначают определяемые понятия языка

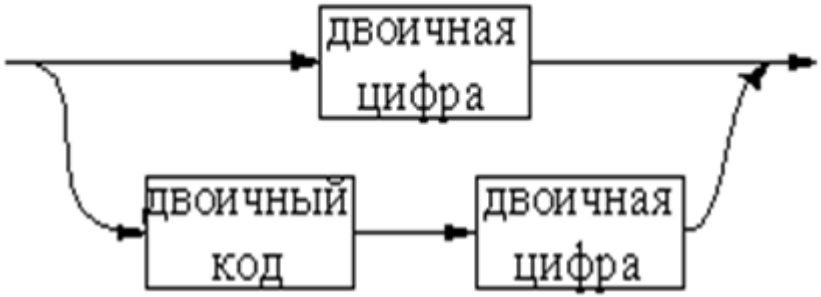
Синтаксические диаграммы

Примеры синтаксических диаграмм:

<двоичная цифра>::=

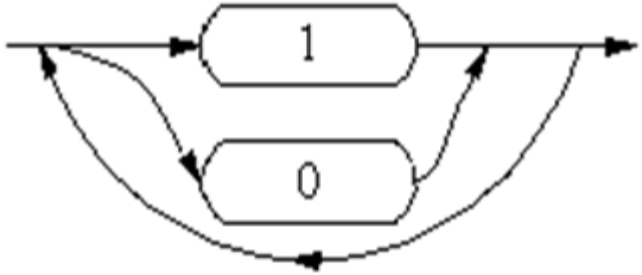


<двоичный код>::=



<двоичный код>::=

<двоичный код>::=

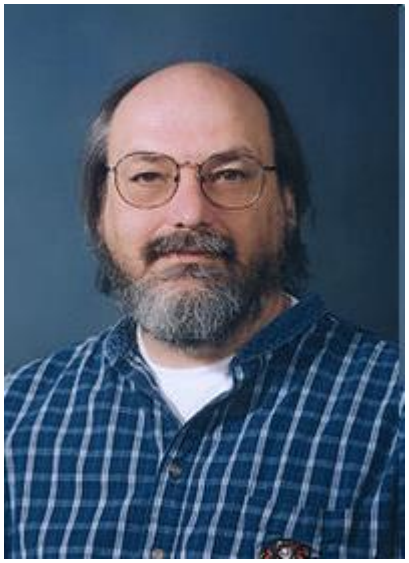


ЯЗЫК ПРОГРАММИРОВАНИЯ СИ

В.Бондарев

История создания языка

Язык программирования **Си** (англ. **C**) был создан в 1972 году сотрудниками Bell Labs **Кеном Топсоном** и **Денисом Ритчи** как развитие языка **Би** (**B**).



Был создан для реализации **ОС UNIX**. Является самым популярным языком ***системного программирования***. В настоящее время также широко применяется как язык ***прикладного программирования***.

История создания языка

В 1978 Ритчи и Керниган опубликовали первую редакцию книги «Язык программирования Си». Эта книга, известная среди программистов как «**K&R**», служила многие годы неформальной спецификацией языка. Версию языка Си, описанную в ней, часто называют «**K&R C**».

В 1983г. Американский национальный Институт Стандартизации (**ANSI**) сформировал комитет для разработки стандартной спецификации Си. По окончании этого долгого и сложного процесса в 1989г. он был наконец утверждён как «Язык программирования Си» *ANSI X3.159-1989*. Эту версию языка принято называть ***ANSI C*** или **C89**.

История создания языка

Связь с C++

1983 г. создан «Си с классами» (**Бьёрн Страуструп**), получивший название **C++.** Язык программирования **C++** произошёл от **C**. Однако в дальнейшем **C** и **C++** развивались независимо, что привело к росту несовместимостей между ними. Предпоследняя редакция Си — **C99** — добавила в язык несколько конфликтующих с **C++** особенностей.

Бьёрн Страуструп неоднократно выступал за максимальное сокращение различий между **C** и **C++** для создания максимальной совместимости между этими языками.

В курсе лекций рассматривается процедурная часть **Си++,** удовлетворяющего стандартам **ANSI C** (American National Standard Institute) и **ISO/IEC 14882** (**1998**г.)

История создания языка

В 1990 году стандарт ANSI C был принят (без изменений) совместным комитетом Международной организации по стандартизации (International Organization for Standardization, ISO) и Международной электротехнической комиссией (International Electrotechnical Commission, IEC) и опубликован в качестве первой редакции стандарта C, ***C90*** (ISO/IEC 9899:1990). Вторая редакция, ***C99***, была опубликована в 1999 году (ISO/IEC 9899:1999), а третья, ***C11***, — в 2011-м (ISO/IEC 9899:2011).

Последняя, четвертая версия стандарта C вышла в 2018 году под названием ***C17*** (ISO/IEC 9899:2018). В настоящий момент ISO/IEC работают над новейшей редакцией языка C, известной как C2x. Согласно опросу, проведенному компанией JetBrains в 2018 году, 52 % программистов на C используют C99, 36 % — C11, а 23 % работают с языком C для встраиваемых систем.

Структура и конструкция программы на Си/C++

<**основные символы**> ::= <буквы> | <цифры> | <специальные знаки> | <пробельные символы> | <разделители>

<**буквы**> ::= A | B | C | … | X | Y | Z | a | b | c | … | x | y | z | _

<**цифры**> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<**специальные знаки**> ::= " | , | { | } | [|] | (|) | + | - | * | / | % | . | \ | ' | : | ? | < | = | > | ! | & | # | ~ | ; | ^ | |

<**пробельные символы**> ::= <пробел> | <символы табуляции> | <символы перехода на новую строку>

<**разделители**>::= <скобки> | <точка> | <запятая> | <пробельные символы>

Структура и конструкция программы на Си/C++

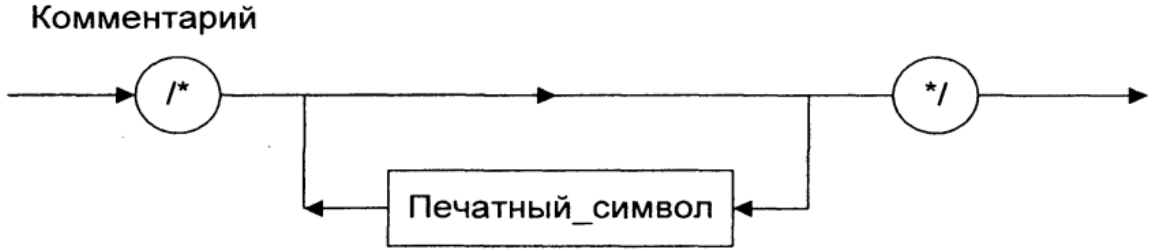
Базовыми элементами языков Си/C++ являются:

- комментарии;
- идентификаторы;
- служебные (зарезервированные) слова:
- константы;
- операторы;
- разделители.

Из базовых элементов строится программа. Рассмотрим сначала базовые элементы, а затем структуру программы.

Структура и конструкция программы на Си/С++

Комментарий – любая последовательность символов заключенная в /* */:



// это однострочный комментарий

/* это многострочный
комментарий*/

Вложенные комментарии наподобие показанного ниже не допускаются стандартом ANSI и большинством компиляторов:

```
/*
   Эта часть комментария правильная
   /* Начало этого комментария игнорируется. */
   Эта строка теперь находится вне комментария! Ошибка!
*/
/* Коммен- // Такое вложение возможно! -тарий */
// Коммен- /* И так тоже можно! */ -тарий
```

Не верно: `X = Y//* Это деление */Z;`

Верно: `X = Y/  /* Это деление */  Z;`

Лексема – минимальная единица языка, имеющая самостоятельный смысл.

Лексемы языка:

- идентификатор;
- ключевые (зарезервированные) слова;
- константы;
- знаки операций;
- разделители.

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

Идентификатор

Идентификатор – это имя сущности в программе.

В идентификаторе могут использоваться латинские буквы, цифры и знак подчеркивания. Прописные и строчные буквы различаются.

Например: **nvalue, nValue, NVALUE** – это три разных имени.

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 накладывают на нее ограничения. Идентификатор – это имя переменной, функции, типа данных и т.д.

При выборе идентификатора следует иметь в виду:

- не должен совпадать с ключевыми словами;
- не рекомендуется начинать с символа подчеркивания (имена системных функций или переменных).

Идентификатор

Для улучшения читаемости программы следует использовать осмысленные имена.

Существует соглашение о правилах создания имен (***венгерская нотация***): каждое слово, составляющее идентификатор, начинается с прописной буквы, а вначале ставиться префикс, соответствующий типу величины, например:

iMaxLenght, lpfnSetFirstDialog;

Еще одна традиция: разделять имена знаками подчеркивания:

max_lenght, number_product,
void my_function_name(); int my_variable_name;

CamelCase – принцип, когда несколько слов пишутся слитно, без пробелов, и каждое новое слово пишется с заглавной буквы. ***CamelCase*** («*ВерблюжийСтиль*») получил свое название из-за заглавных букв, которые напоминают верблюжьи горбы:

void MyFunctionName(); int MyVariableName;

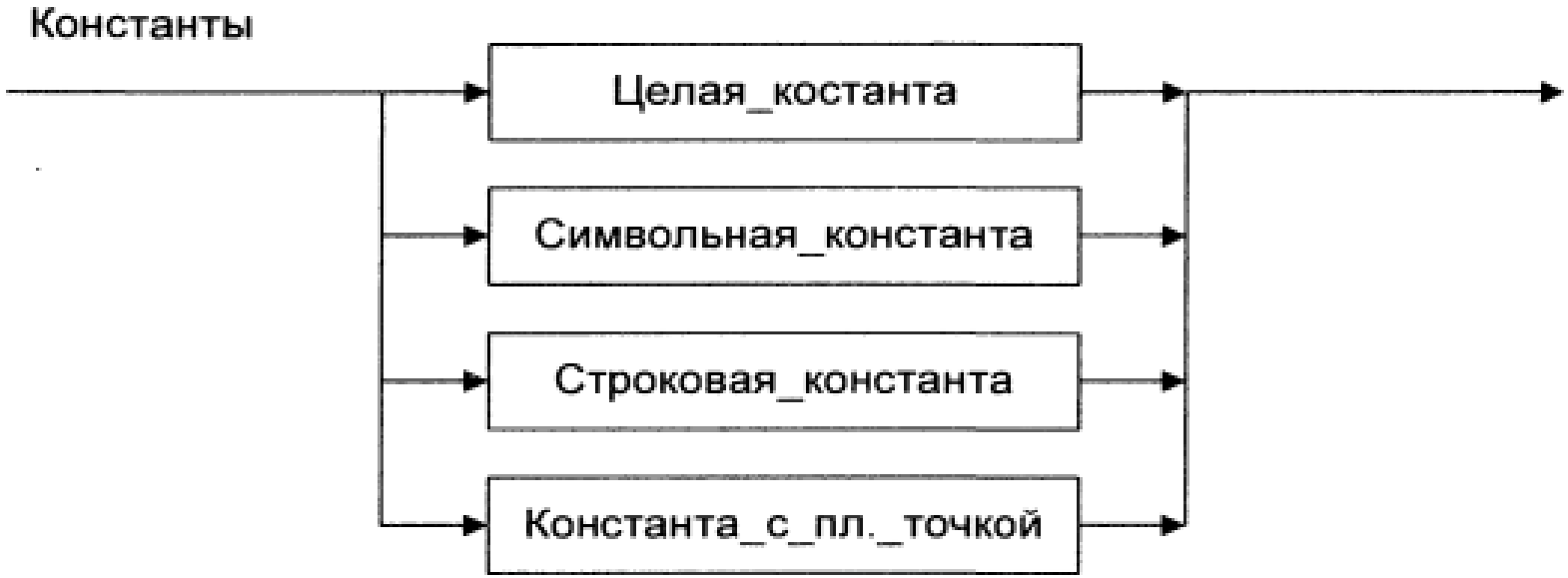
Ключевые слова C/C++

Ключевые слова – это предварительно определенные зарезервированные идентификаторы, имеющие специальные значения.

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

Константы

Константа – это неизменяемая величина.



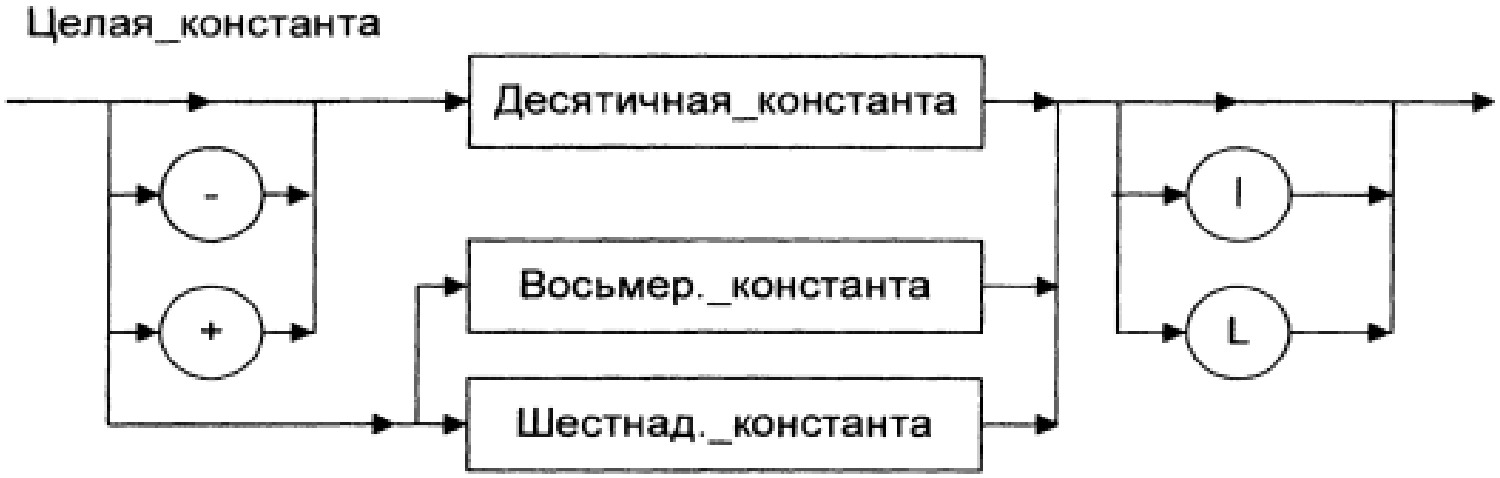
Целые константы:

8, -32, 1 234 – в десятичной записи;

01, 020, 07155 – в восьмеричной записи;

0xA, 0x1B8, 0X00FF – в шестнадцатеричной записи.

Константы



Константа типа **long** завершается буквой **1** или **L**, например **123456789L**

Беззнаковые константы заканчиваются буквой **u** или **U**, а окончание **ul** или **UL** говорит о том, что тип константы - **unsigned long**. Например: **1234u**, 1**234UL**

Диапазон значений констант для 16-разрядного процессора от -32 768 до +32 767, для 32-хразрядного процессора от -2^{31} до $+(2^{31}-1)$

0xf5UL -?

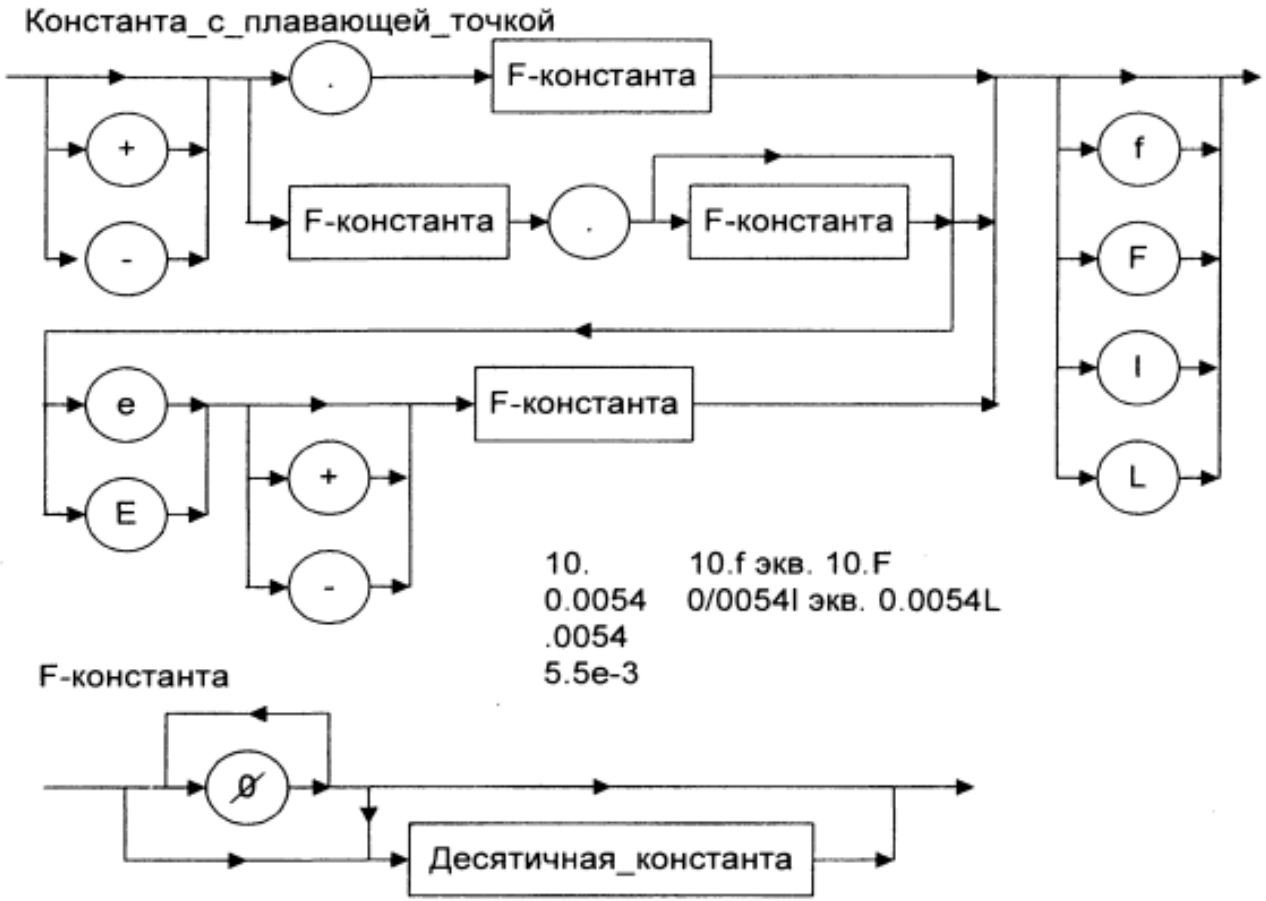
Константы

Константы с плавающей точкой:

5.2, 0.35, 0.2E+6, -1.1e-3

Константы с плавающей точкой по умолчанию имеют тип **double**. Окончание **f** или **F** указывает, что константа имеет тип **float**.

Например, **123.4f**



Константы

Символьная константа есть целое, записанное в виде символа, обрамленного одиночными кавычками, например **'x'**. Значением символьной константы является числовой код символа из набора символов на данной машине.

Пример: **'F', '*', '1', ' ', '\n'**

Для кодирования одного символа используется 1 байт (256 символов).

Символы делятся на две группы: печатные и непечатные.

Пример непечатных (управляющих) символов:

'\n' – переход на новую страницу;

'\t' – табуляция;

'\0' – нулевой символ, признак конца строки.

Константы

Строковая константа - это нуль или более символов, заключенных в двойные кавычки, как, например,

"Я строковая константа", "Hello world"

В строки можно включать эскейп-последовательности. Например, \" (представляет собой двойную кавычку):

"Издательский дом \"Питер\""

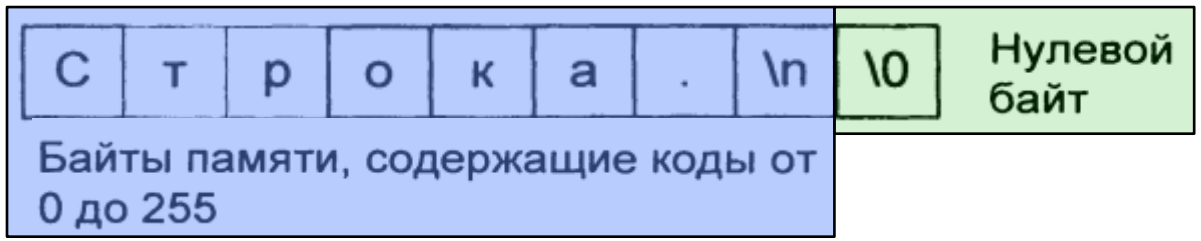
Строковые константы можно **конкатенировать** ("склеивать") во время компиляции; например, запись двух строк

"Здравствуй," " мир!"

эквивалентна записи одной следующей строки: **"Здравствуй, мир!".**

Для запоминания строковых констант используется по 1 байту на каждый символ и ***автоматически*** добавляется признак конца строки.

В памяти компьютера строка **"Строка\n"** будет представлена в 9 байтах:



Сколько памяти занимает пустая строка ""?