Замечания по поводу констант и инициализации переменных:

1. **Константы** можно условно разделить на три группы:

а) без имени (имеют только значение) – **Литералы**:

a = b + 25 ; или strcpy(s, “123”);

текстовое представление значения константы

литерал (участвует своим значением в выражении)

б) с именем (имеют и **имя**, и **значение**) – **Именованные константы**:

Общий вид: #define <Имя> <Значение>

#define N 2; эквивалентно Const N=2; в Паскале

нельзя

Константы такого вида (#define <имя> <значение>) более точно подходят к определению простых (нетипизированных) констант в Паскале, при использовании которого <имя> присутствует только в исходном тексте программы, а при построении кода программы **компилятор прямо подставляет** <значение> вместо <Имя>там, где будет найдено константа <имя>;

в) типизированные (имеют и тип, и имя, и значение)

Общий вид на Си: const <Тип> <Имя> = <Значение>;

const int N=25; эквивалентно Const N: integer =25; в Паскале

(почти)

модификатор признак секции констант

Константы такого вида (с модификатором const) в Си похожи на типизированные константы в Паскале, однако в Си исключается возможность изменять значение константы (в Паскале при настройках по умолчанию значения типизированных констант можно менять). Любая попытка модификации или назначения новой величины дает в итоге ошибку.

1

Замечание: типизированной константе из Паскаля (const c: integer=0;) в Си более соответствует определение вида

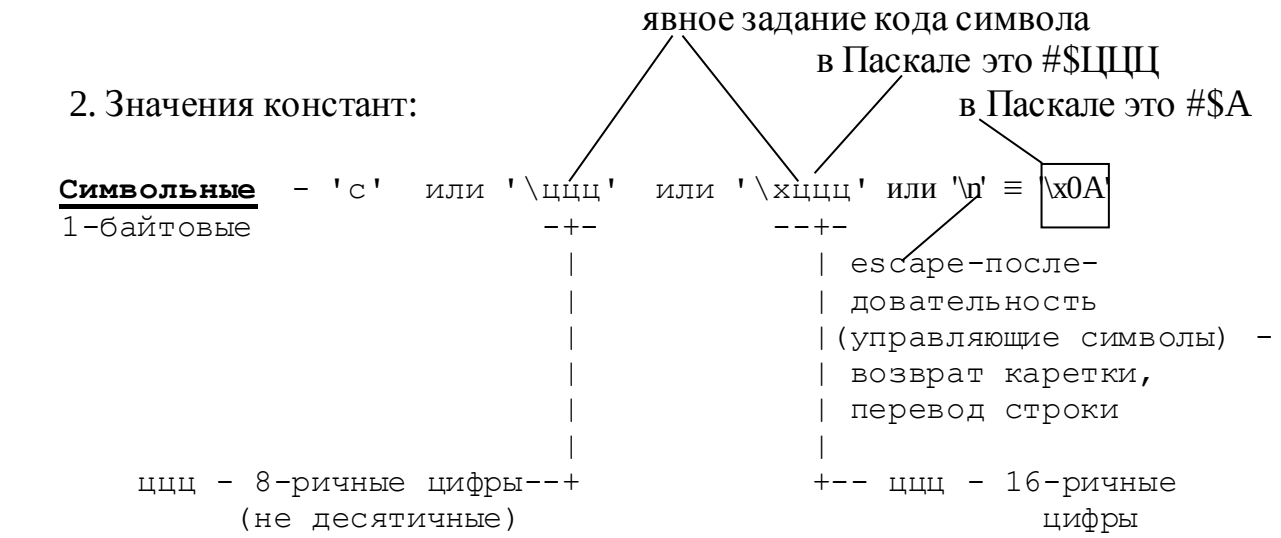
int c = 0; (без const).

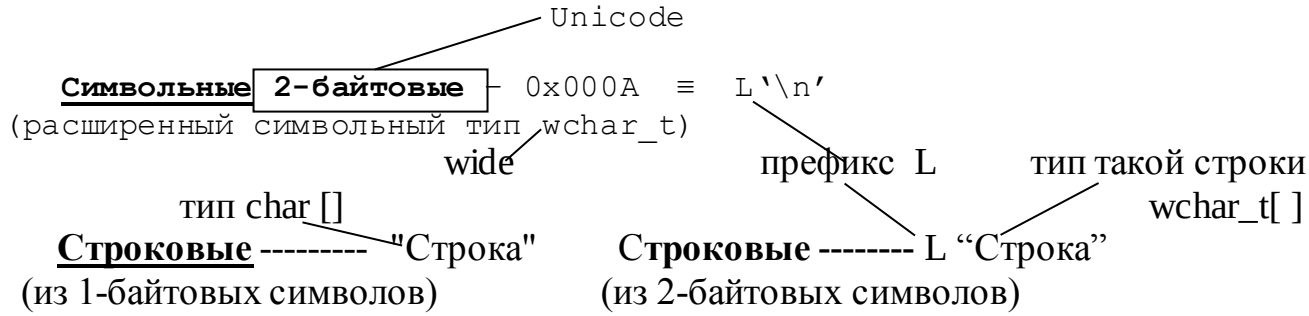
Модификатор const в Си часто используется в заголовках (прототипах) функций при описании формальных параметров:

void f(*const* char b,…).

Модификатор const указывает (требует), чтобы компилятор следил (вставлял специальный код) за тем, чтобы при выполнении процедуры **параметр *b* не изменялся** (в теле данной функции). Но в отличие от Паскаля слово Const при описании формального параметра в заголовке **не означает в Си передачу по адресу**.

2. Значения констант:





2

Целые - 0x20 (шестнадцатеричные) = 32_{10}
040 (восьмеричные) = 32_{10} (octal)
32 (десятичные) = 32_{10} (decimal)

могут быть только беззнаковыми

если без суффикса, то будет int

Кроме **префикса** (0X или 0) целые константы могут иметь **суффикс**:

а) *u* или *U* - беззнаковая константа (внутреннее представление значения вида 3500U в памяти будет рассматриваться как беззнаковое);

б) *l* или *L* - типа long (занимает 4 байта).

Замечание: надо различать

67000 --- по умолчанию это значение типа int (2 или 4 байта в зависимости от разрядности архитектуры), поэтому если разрядность архитектуры 16 бит, то при выделении компилятором памяти под константу будет потеряна старшая часть числа (старшие 2 байта),

67000L - это значение типа long (занимает 4 байта или 8);

35000+35000 -- тип int 2 байта
35000+35000L -- тип long int 4 байта
32767+1 -- тип int 2 байта
32767L+1 -- тип long int 4 байта

в) *h* или *H* – типа short (занимает 2 байта).

Эти суффиксы можно комбинировать (uh или hu – unsigned short, ul или lu– unsigned long);

г) *ll* или *LL* – типа long long (8 байт). Но это нестандартный тип.

Вещественные – задаются как в Паскале: а) .25; б) 125. ; в) 123.45; г) 1.5e-5. Можно еще указать суффиксы:

- без суффикса – тип double (8 байт) -- по умолчанию;
- с суффиксом *f* или *F* - тип float (4 байта);
- с суффиксом *l* или *L* - тип long double (10 байт).

вещественные числа хранятся приближенными

3. Си позволяет **явно инициализировать** при объявлении любую переменную по образу типизированных констант в Паскале. Формат следующий:

<тип> <имя> = <значение> ; ≠ **static** <тип> <имя> = <значение> ;

при каждом входе в блок — при первом входе в блок

Элементы данных, нуждающиеся более чем в одной величине (массивы, структуры), должны иметь значения, заключенные в фигурные скобки и отделенные запятыми:

```
int x = 1, y = 2;
char name[]="Франк";
char answer ='Y';
char list[2][10]  = {"Первая строка", "Вторая строка"};.
```

длина больше 10

4. В программе на Си если глобальным и статическим переменным явно не назначено (присвоено) значение (в месте описания), то по умолчанию перед началом выполнения программы им **присваивается начальное значение 0** (автоматически). Автоматические переменные (в стеке находятся) и динамические не инициализируются автоматически.

```
void main(void)
 {
   int i;
   char *s = new char[3];
  //то же самое, что
   char *s = (char *)malloc(3); ≡ char s [3];
```

```
int i;
void main(void)
  {
    .....
  }
```

```
void main(void)
 {
    static int i;
    .......
 }
```