Тема 4 ТИПЫ ДАННЫХ И ИХ ОБЪЯВЛЕНИЕ

4.1 Арифметические типы

Целые типы

1) char
(1 байт) от -128 до 127

2) short или short int
(2 байт) от -32768 до 327677

3) int

4) long или long int
(4 байт) от -2^{31} до $(2^{31}-1)$

5) unsigned char
(1 байт) от 0 до 255

6) unsigned short int
(2 байт) от 0 до 65 535

7) unsigned long int
(4 байт) от 0 до $(2^{32}-1)$

плавающие

1) **float**

2) **double**

3) long double

Важное отличие языка СИ от других языков (PL1, FORTRAN, и др.) является отсутствие принципа умолчания, что приводит **к необходимости объявления всех переменных** используемых в программе явно вместе с указанием соответствующих им типов.

Объявления переменной имеет следующий формат:

[спецафикатор-класса-памяти] спецификатор-типа имя [=инициатор] ;

Можно опускать

Можно опускать ,
Инициализировать отдельно

ПРИМЕРЫ

```
unsigned int n=1;    float x=1.0e-3, y=3.1416  ;
char b;
int c;        //(подразумевается  signed   int c  );
unsigned d=0;  // (подразумевается  unsigned int d  );
signed f;   //  (подразумевается  signed   int f  ).
```

Переменная любого типа может быть объявлена как немодифицируемая.

Это достигается добавлением ключевого слова
const к спецификатору-типа.

Объекты с типом const представляют собой данные используемые только для чтения, т.е. этой переменной не может быть присвоено новое значение.

Примеры:

```
const double A=2.128E-2;
const B=286; //(подразумевается const int B=286)
```

4. 2. Указатели

Указатель - это адрес памяти, распределяемой для размещения идентификатора (в качестве идентификатора может выступать имя переменной, массива, структуры, строкового литерала).

В том случае, если переменная объявлена как указатель, то она содержит адрес памяти, по которому может находится скалярная величина любого типа.

При объявлении переменной типа указатель, необходимо определить тип объекта данных, адрес которых будет содержать переменная, и имя указателя с предшествующей звездочкой (или группой звездочек).

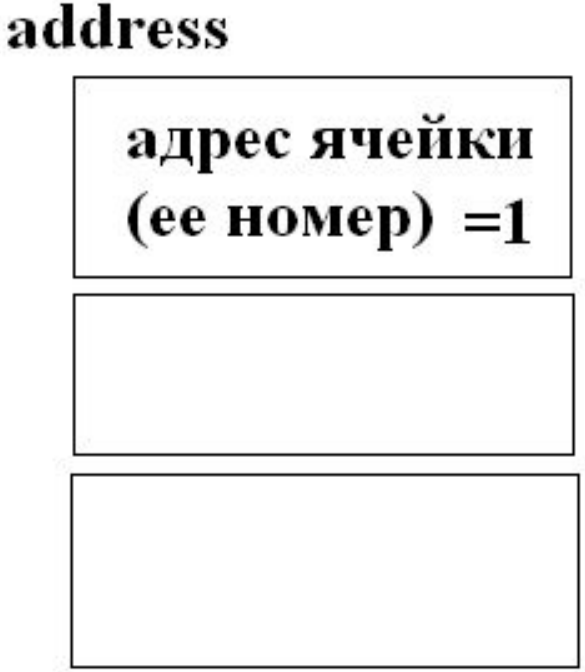
Формат объявления указателя:

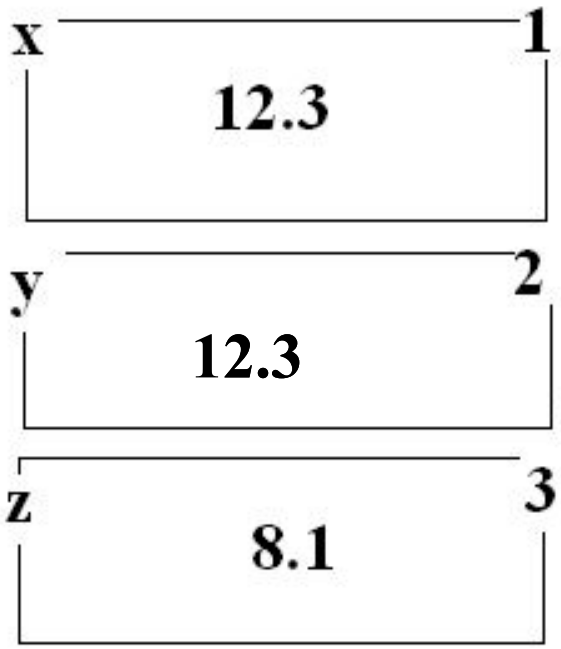
спецификатор-типа [модификатор] * ИМЯ;

спецификатор-типа *
ИМЯ;

Например

```
…
float x,y;
float *address;
x=12.3;
y=2.3;
address=&x;
y=*address;
```





4.3 Адресные операции – это унарные операции

1) Операция взятия адреса

& ИМЯ

1) операция разадресации

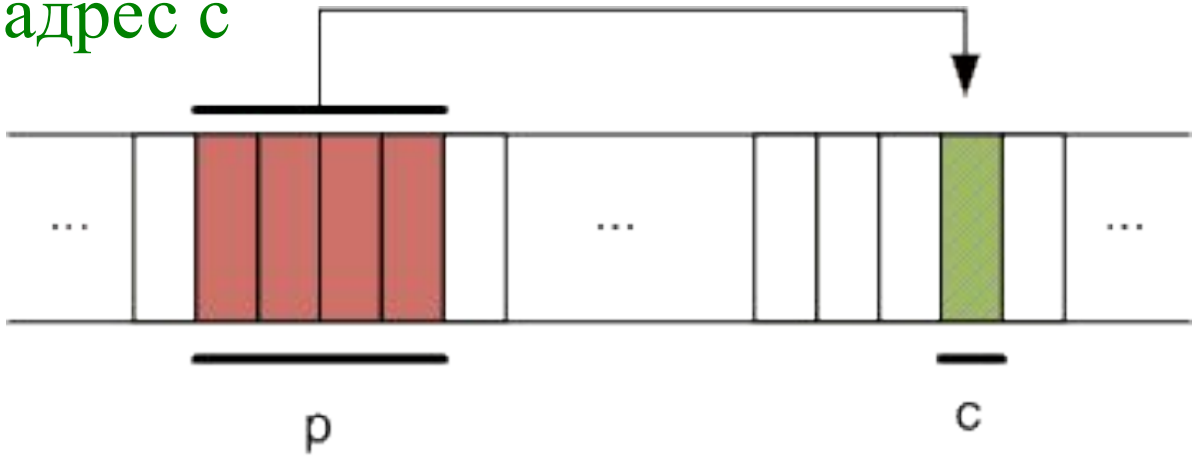
*** ИМЯ УКАЗАТЕЛЯ**

```
void main()
{
int x, y=0;
int * p;  // только объявили
p=&y;   // записали адрес, т.е. инициализировали
printf (“  адрес ячейки с именем y  = %p ”, p );
printf (“  адрес ячейки с именем y  =%p ”, &y );

x=*y; // переписываем число из ячейки y в ячейку x

printf (“  x= % d ”,  x );
}
```

```
char c;   // переменная
char *p; // указатель
scanf(“%c”,&c);
p = &c;  // p = адрес c
```

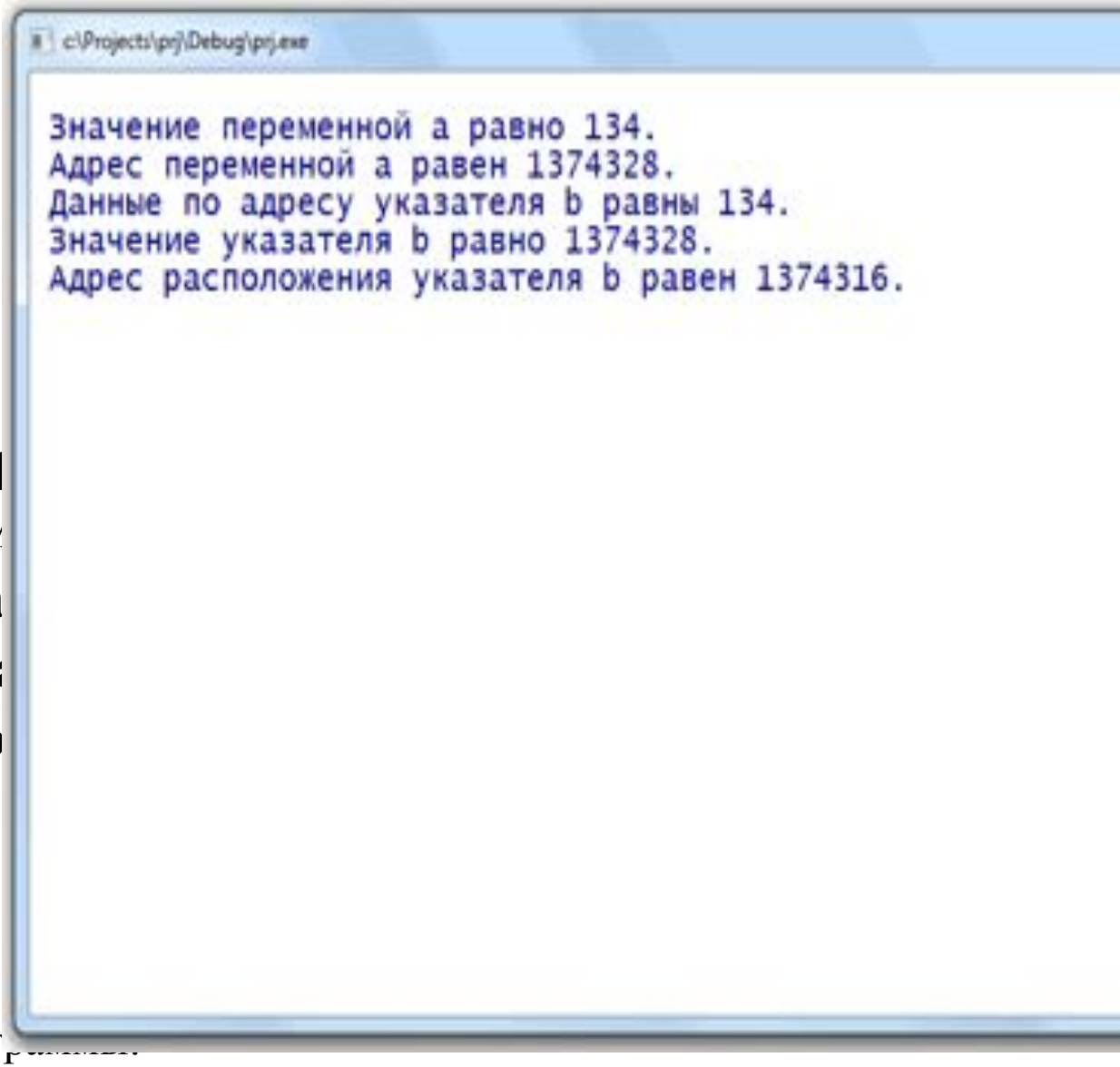


Для указанного примера обращение к одним и тем же значениям переменной и адреса представлено в таблице

	Переменная	Указатель
Адрес	&c	p
Значение	c	*p

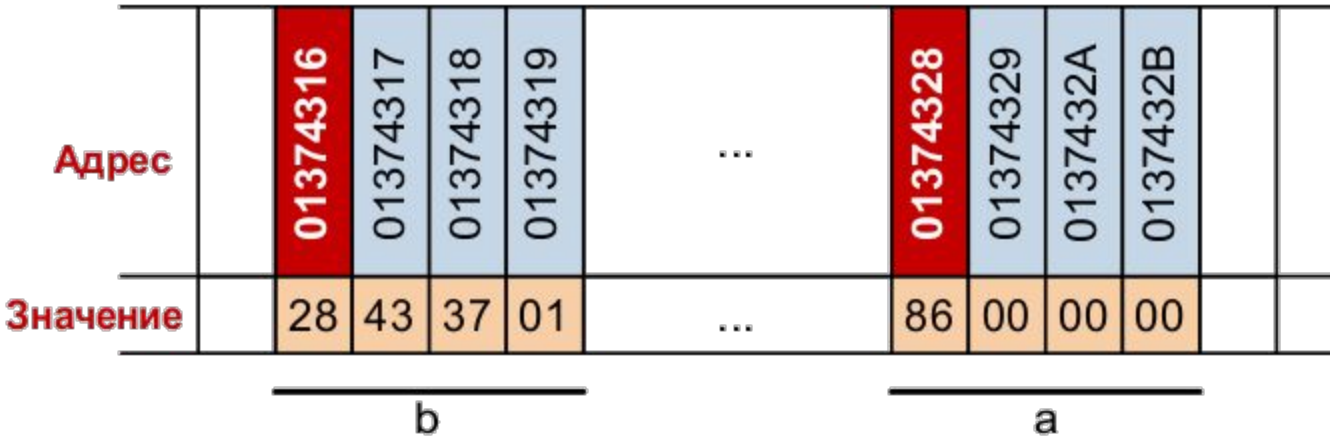
Пример

```
#include <stdio.h>
#include <stdlib.h>
int main() {
  int a, *b;
    a=134;
    b=&a;
  printf("\n Значение пе
  printf("\n Адрес перем
  printf("\n Данные по а
  printf("\n Значение ук
  printf("\n Адрес распо
  getchar ();
  return 0;
}
```

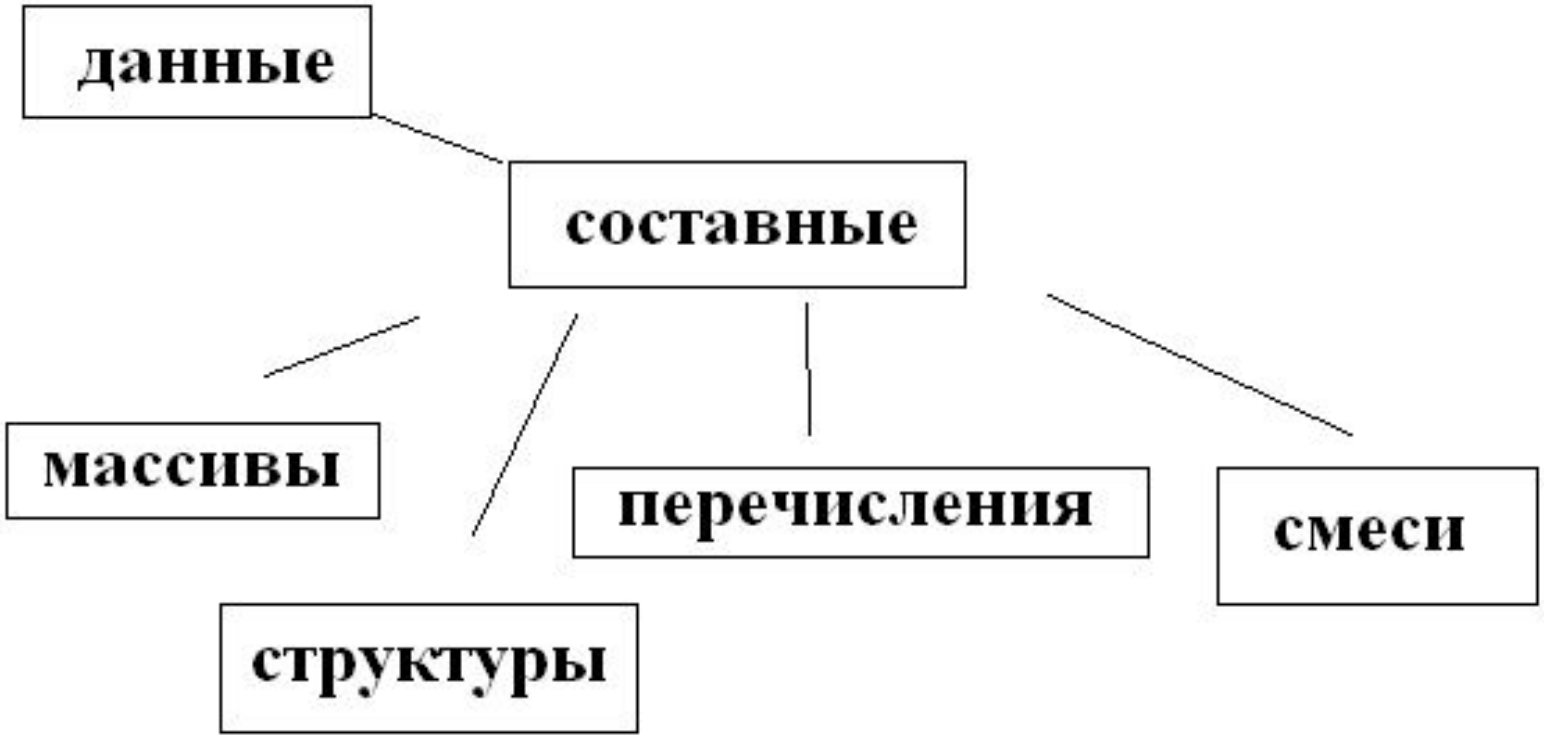


Результат выполнения прог

Расположение в памяти переменной a и указателя b:



Необходимо помнить, что компиляторы высокого уровня поддерживают **прямой способ адресации**: младший байт хранится в ячейке, имеющей младший адрес.


данные
составные
массивы
структуры
перечисления
смеси

4.3. МАССИВЫ

Массивы - это группа элементов одинакового типа (double, float, int и т.п.).

Массив - это несколько пронумерованных переменных, объединенных общим именем. Все переменные имеют ОДИН И ТОТ ЖЕ ТИП.

Из объявления массива компиллятор должен получить информацию о типе элементов массива и их количестве.

Объявление массива имеет два формата:

1) **спецификатор-типа идентификатор** [константное - выражение];

2) **спецификатор-типа** идентификатор **[ ];**

Спецификатор-типа задает тип элементов объявляемого массива

Рассмотрим **ПОЛКУ с N ящиками**,
пусть **имя полки - var.** Тогда каждый ящик-ячейка имеет имя

var[0]
var[1]
...
var[N-1]

Нумерация идет с НУЛЯ.

```
  var
 -----------------------------------------   -----------------
 |         |         |          |           |         |
 |         |         |          | ....   ...|         |
 |         |         |          |           |         |
 -----------------------------------------   -----------------
/ var[0] /  var[1] /  var[2] /          / var[N-1] /
---------   ---------  ---------           -----------
```

Массив объявляется так:

int var[N];

здесь N - его размер, число ячеек.

В операторах для обращения к **n-ому ящичку (где 0 <= n < N)** используется имя ящика

var[n]

где n - целое значение (или значение целой переменной, или целочисленного выражения), "индекс в массиве".

Эта операция [] называется "индексация массива".

Индексация - есть ВЫБОР одного из N ящиков при помощи указания целого номера.

var - массив (N ячеек)

n - выражение (формула), выдающая целое значение в интервале 0..N-1

var[n] - взять один из элементов массива. Один из всех.

n - номер ящика - называется еще и "индексом" этой переменной в массиве.

Индекс - часть ИМЕНИ ПЕРЕМЕННОЙ.

Пример:

```
int var[5];                                  /* 1 */

var[0] = 2;                                  /* 2 */

var[1] = 3 + var[0];                         /* 3 */

var[2] = var[0] * var[1];                    /* 4 */

var[3] = (var[0] + 4) * var[1];              /* 5 */

printf("var-третье = %d\n", var[3]);
```

В ходе этой программы элементы массива меняются таким образом:

	var[0]	var[1]	var[2]	var[3]	var[4]
/* 1 */	мусор	мусор	мусор	мусор	мусор
/* 2 */	**2**	мусор	мусор	мусор	мусор
/* 3 */	**2**	**5**	мусор	мусор	мусор
/* 4 */	**2**	**5**	**10**	мусор	мусор
/* 5 */	**2**	**5**	**10**	**30**	мусор

Как видим, каждый оператор изменяет лишь ОДНУ ячейку массива за раз.

Пример: объявление + инициализация

```
char carr[3] = {‘d’, ‘F’, ‘y’};

double dbarr[4]= {0.0, 0.5, -4.5, 2.0};

int narr[] = {2, 67, 7, 8};
```

- **Массивы НЕЛЬЗЯ присваивать целиком, язык Си этого не умеет.**

```
int a[5];
int b[5];
a = b;  /* ошибка */
```

- **Также нельзя присвоить значение сразу всем элементам (ячейкам) массива:**

```
a = 0;  /* ошибка */
```

не делает того, что нами ожидалось, а является ошибкой.

- **Для обнуления всех ячеек следует использовать цикл:**

```
int i;
for(i=0; i < 5; i++)    /* для каждого i присвоить a[i] = 0; */
        a[i] = 0;
```

ИМЯ Массива = адрес ячейки, где хранится 0 элемент

Пример

…

```
double M[9],    *p;
…
int n;
p=M; // в ячейку p  записан адрес ячейки M[0]
        // эквивалентно                p=&M[0];
printf(“ %p \n”,   p);
p=M+3; // в ячейку p  записан адрес ячейки M[0+3]
printf(“ %p \n”,p); // печать адреса
n=p-M; // n=(адрес M[3])-(адрес M[0])=3 – сдвиг по адресам
```

Многомерные массивы

тип имя [конст. – выраж.1] [конст.- выраж.2];

тип имя [конст. – выраж.1] [конст.- выраж.2] [конст.- выраж.3];

и т.д.

Пример

…

double C [2][4];

…

Объявление+инициализация

```
double C [2][4]={   1 .0,   2.0,   -3.0,      -5.0},
                    {-3.3,   0.0,    11.,   1.e-10}
                };
```

$$\begin{matrix} C00 & C01 & C02 & C03 \\ C10 & C11 & C12 & C13 \end{matrix}$$

Пример – печать массива

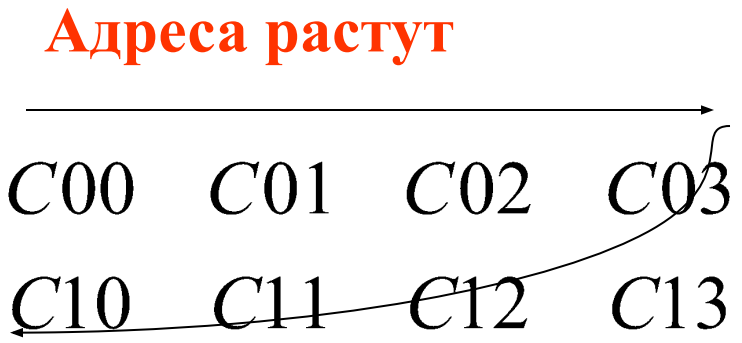
…

```
double C [2][4];
```

…

```
printf (“%f \n”, C[0][0]);
printf (“%f \n”, *C);

printf (“%f \n”, C[0][2]);
printf (“%f \n”, *(C+2) );
```



Тема 5 ОПЕРАТОРЫ И ОПЕРАЦИИ

5.1. Классификация операторов

5.2 Оператор- выражение

Большинство операторов является операторами выражение, которые имеют вид

ВЫРАЖЕНИЕ;

Обычно операторы выражение являются присваиваниями и вызовами функций. Например,

1) Вызовы функций вывода и ввода

```
printf(“введите целое число x=”);
scanf(“%d”, & x);
```

2) Присваивания **a=1;**

1) Операция присваивания – бинарная операция.

В отличие от других языков в С присваивание выполняет операция, а не оператор.