IT ШКОЛА SAMSUNG

ЧТО ЛУЧШЕ ЗНАТЬ ИЗ СИ НОВИЧКАМ В JAVA

Юн Светлана
к.т.н., доцент, Менеджер образовательных проектов,
Исследовательский центр Samsung, г. Москва

2 июля 2014г.

SAMSUNG

Что такое Указатель

Указатель - это переменная, значением которой является адрес другой переменной

Переменная - указатель

Может хранить только **АДРЕС ПАМЯТИ**

Другие переменные

Значения любых типов

Объявление указателя

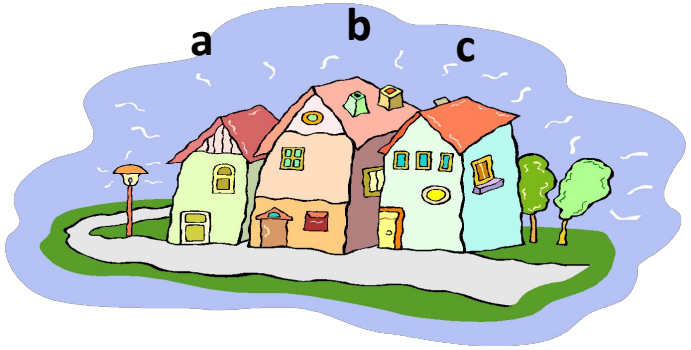
Указатели

pa

pb

pc

Переменные любых типов

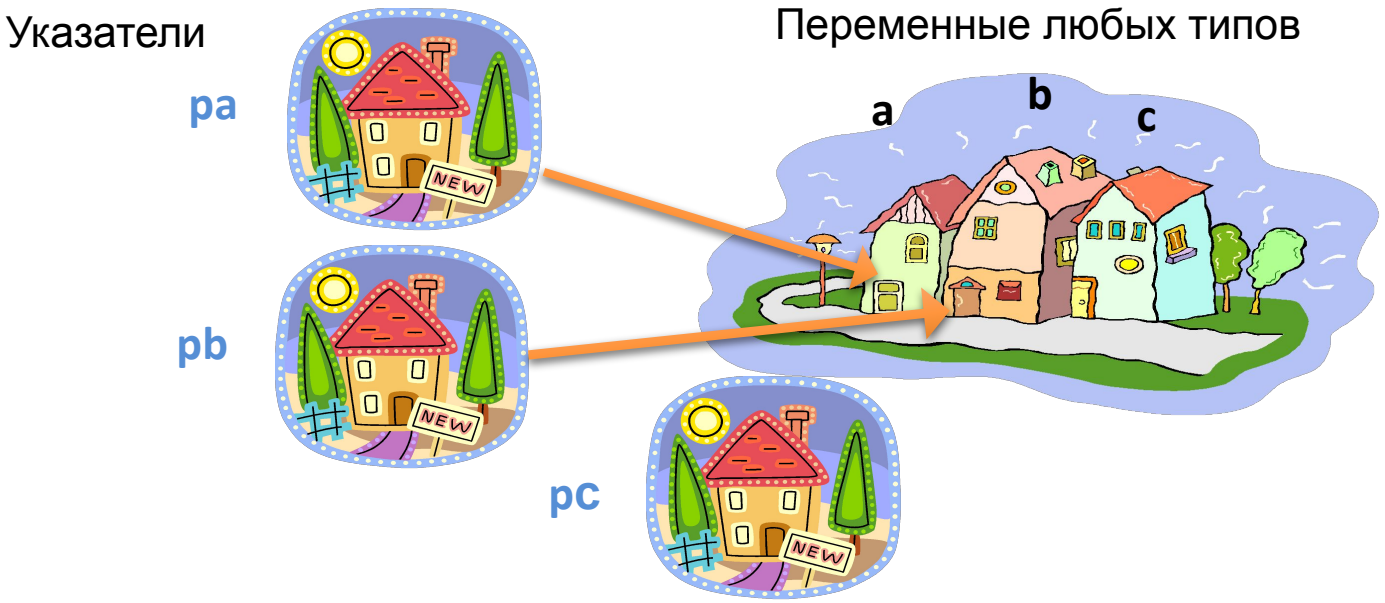



```
int   a;
int *pa;

double   b, *pb;

char c, *pc;
```

Тип указателя определяется типом переменной, на которую он указывает (ссылается). Тип необходим, чтобы компилятор смог интерпретировать область памяти на которую указывает указатель

* в объявлении переменной показывает, что объявляется указатель

Инициализация указателя




```
int   a=10;
int *pa;

double   b, *pb;

char *pc;
```

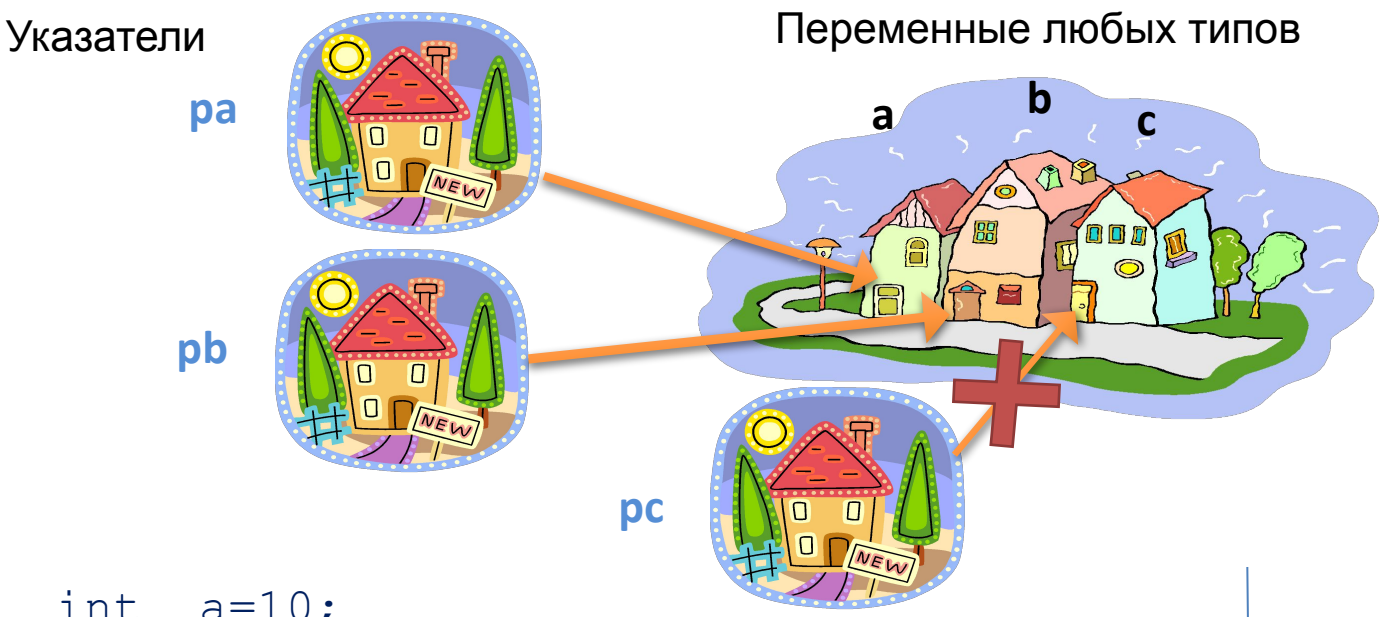
```
int   a=10;
int *pa;
pa=&a;

double   b, *pb;
pb=&b;

pc=NULL;
```

```
pb=&a; //???
```

& - операция взятия адреса переменной

NULL - нулевой указатель . Используется для того, чтобы показать, что данная переменная-указатель не указывает ни на какой объект (определен в stdio.h , stddef.h и др.)

Разыменование указателя



& - операция взятия адреса переменной

* - операция разыменования, косвенного обращения к переменной (не путать с * при объявлении указателя!)

```
int  a=10;
int *pa;
pa=&a;
printf ("%d %d", a,*pa);//???
```

10 10

```
double  b, *pb;
pb=&b;
*pb=3.5;
printf ("%lf %lf", b,*pb);//???
```

3.5 3.5

```
char c, *pc;
*pc='R'; //???
```

Указатель не инициализирован – непредсказуемый результат!!!

Указатели и массивы. Адресная арифметика

pA

A- адрес начала массива

```
int  A[10];

int *pA=A; //эквивалентно pA=&A[0];

pA++;//эквивалентно pA=&A[1];

*(pA+1)=3; //эквивалентно A[2]=3;

*(pA+10)=7; //???
```

Имя массива - адрес начала массива

++ сдвиг указателя на одну переменную вперед

-- сдвиг указателя на одну переменную назад

Размер сдвига в байтах соответствует типу указателя

Адреса можно сравнивать. Верно неравенство: &A[0]<&A[1]

Зачем нужны указатели?

ПЕРЕДАЧА ДАННЫХ В ФУНКЦИЮ

Если необходимо вернуть из функции в качестве результата больше, чем одну переменную, либо передать в функцию большой связанный объем данных (структуру данных), то достаточно передать адрес начала области данных, а исходя из понимания типа данных и их структуры компилятор сможет понять, что лежит в этой области данных

ДИНАМИЧЕСКОЕ ВЫДЕЛЕНИЕ ПАМЯТИ

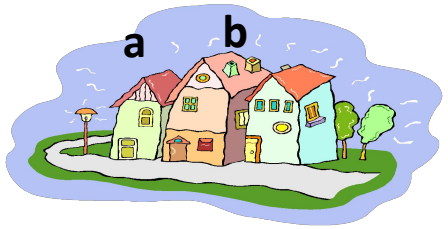
Позволяет выделять память под динамические переменные по ходу выполнения программы. Применяется тогда, когда не известно заранее, какие переменные нам будут необходимы для хранения данных

…..

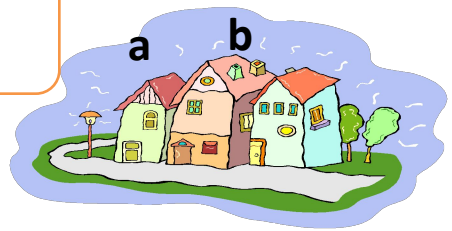
Способы передачи параметров в функцию

```
#include <stdio.h>

void sort(int a, int b)
{
     int c;
     if(a<b)
     { c=a;
        a=b;
        b=c;}
}
int main(void)
{
     int a, b;
     a=10;
     b=20;
     sort(a, b);
     printf("a=%d
b=%d\n",a,b);
}
```

Формальные параметры



Фактические параметры



Problems @ Javadoc Declaration Console

<terminated> 111.exe [C/C++ Application] C:\Work\Android_

a=10 b=20

Передача по значению

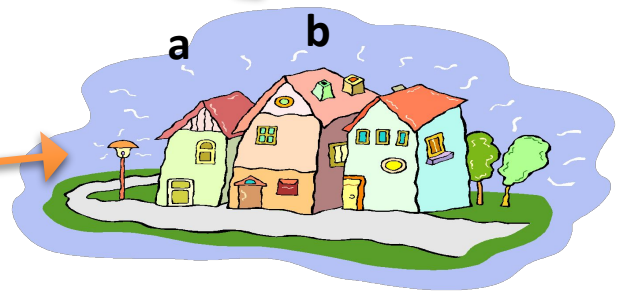
Значения фактических параметров копируются в формальные параметры функции.

Параметры функции могут рассматриваться как локальные переменные, для которых выделяется память при вызове функции и производится инициализация значениями фактических параметров.

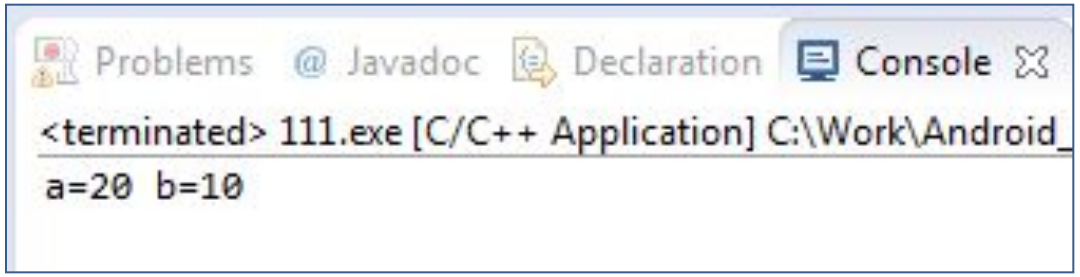
При выходе из функции значения этих переменных теряются

Способы передачи параметров в функцию

```
#include <stdio.h>

void sort(int* pa, int* pb)
{
    int c;
    if(*pa<*pb)
    {
        c=*pa;
       *pa=*pb;
       *pb=c;
    }
}

int main(void)
{
    int a, b;
    a=10;
    b=20;
    sort(&a, &b);
    printf("a=%d
b=%d\n",a,b);
    return 1;
}
```

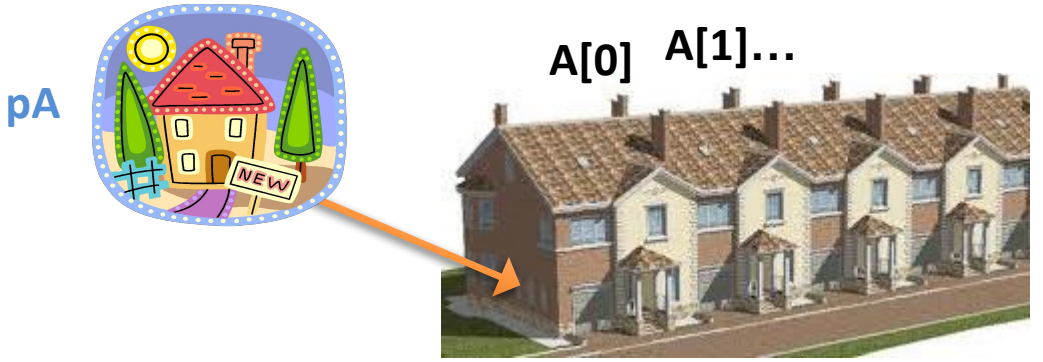


Передача по ссылке
Если в качестве параметров передать адреса переменных, то путем использования операции разыменования можно изменить исходные переменные



Передача массивов в функцию – передача по ссылке

```
 #include <stdio.h>

/* функция переворачивает массив заданной
     размерности*/
void reverse(int* pA, int n)
{
    int c;
    /*установить на посл.элемент массива*/
    int  *p=pA+n-1;

while (pA<p)
{
    c=*pA; *pA=*p;*p=c;
    pA++;p--;
}}

int main(void)
{
     const int n=5;
     int i, A[n]={1,2,3,4,5};
     reverse(A,n);
     for(i=0;i<n;i++)
          printf("A[%d]=%d\n",i,A[i]);
     return 1;
}
```



A- адрес начала массива

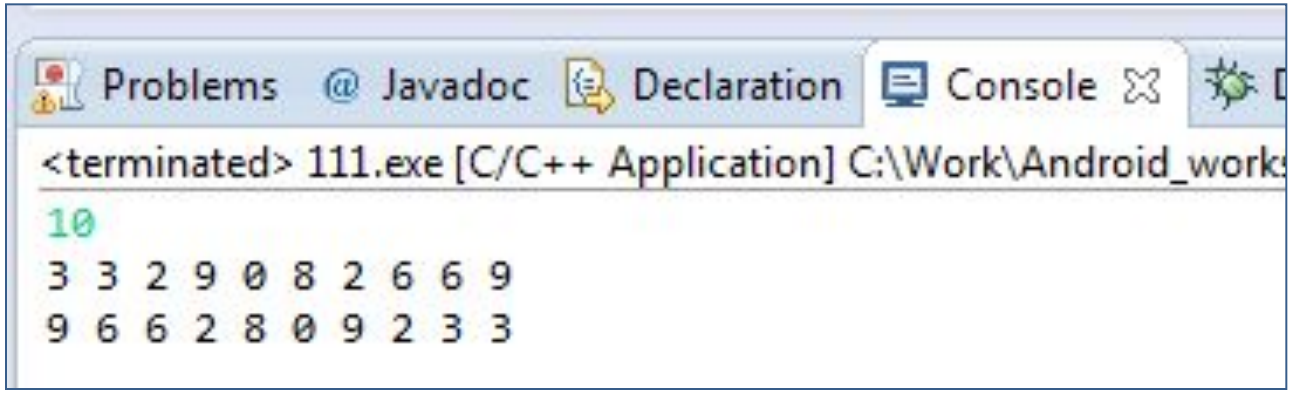
```
Problems
<terminated> 1
A[0]=5
A[1]=4
A[2]=3
A[3]=2
A[4]=1
```

Динамическое выделение памяти

```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    int n;
    int i, *p, *pA;
    //ввод размера массива
    scanf("%d",&n);
    pA= (int*)malloc(sizeof(int)*n);
    p=pA;
    //заполнение случ. числами
    for(i=0;i<n;i++)
    {
        *p=(int)rand()%10;
        p++;
    }
    for(i=0;i<n;i++)
    printf("%d ",pA[i]);
    reverse(pA,n);
    printf("\n");
    for(i=0;i<n;i++)
       printf("%d ",pA[i]);
    free(pA);
    pA=NULL;
    return 1;
}
```

Функция malloc() выделяет память в динамической памяти (куче) заданного размера (в байтах)

Функция free() освобождает память



Функция malloc()

```
pA= (int*)malloc(sizeof(int)*n);
```

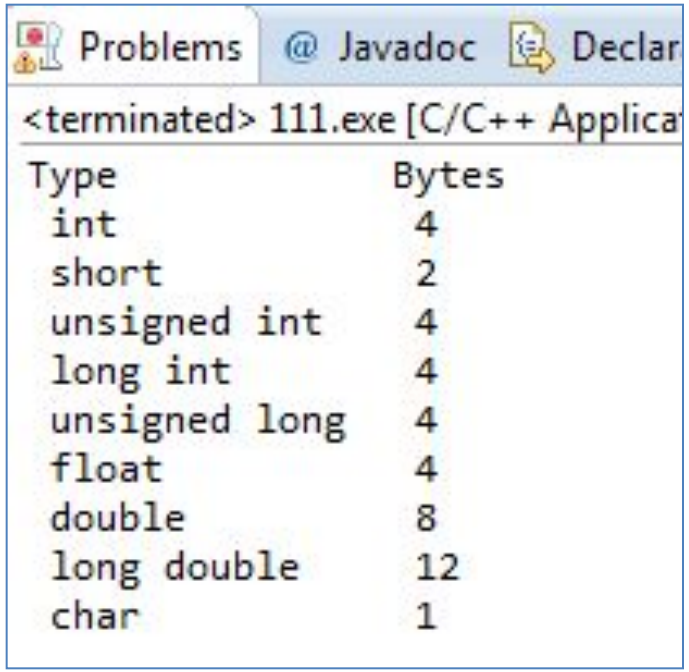
void *malloc(size_t size)
функция возвращает значение void*. Для того, чтобы присваивание было произведено корректно необходимо преобразование типа указателя. В примере: (int*)

size_t size – объем необходимой памяти в байтах.

sizeof(int) – оператор определения размера памяти, отводимого под одну переменную заданного типа данных

Оператор sizeof()

```
#include<stdio.h>

int main(void)
{
  printf("Type\t\tBytes");
  printf("\n int\t\t %d",sizeof(int));
  printf("\n short\t\t %d",sizeof(short));
  printf("\n unsigned int\t %d",sizeof(unsigned int));
  printf("\n long int\t %d",sizeof(long));
  printf("\n unsigned long\t %d",sizeof(unsigned long));
  printf("\n float\t\t %d",sizeof(float));
  printf("\n double\t\t %d",sizeof(double));
  printf("\n long double\t %d",sizeof(long double));
  printf("\n char\t\t %d\n",sizeof(char));
  return 1;
}
```



Функция free(). Утечки памяти

```
free(pA);
pA=NULL;
```

void free(void *ptr)
ptr – указатель на освобождаемую область памяти

После free() переменная pA хранит указатель на несуществующую переменную («висячая ссылка»), поэтому считается правилом хорошего тона такой указатель обнулять.

Утечки памяти: Необходимо отслеживать соответствие каждому malloc() своего free()

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
  int i;
  int *pA[5],*p;
  for (i=0;i<5;i++)
  {
     p= (int*)malloc(sizeof(int));
     pA[i]=p;
  }
  free(p); //освобождено 1 раз
  p=NULL;
  return 1;
}
```

Указатели в Си и Java

	Указатели в Си	Ссылки на объекты и массивы в Java
Адресная арифметика	Возможны операции адресной арифметики	Не возможны
Передача параметров функции по ссылке	Можно передавать указатели на любые типы данных	Нельзя передать ссылки на базовые типы, только на объекты и массивы. Функция не может изменить полученную ей ссылку. Однако она может вызвать метод объекта, изменяющий поля этого объекта
Выделение памяти	malloc()	new
Освобождение памяти	free()	Память освобождается автоматически специальным процессом, называемым *сборщик мусора* (англ. *garbage collector*)

IT ШКОЛА SAMSUNG

Юн Светлана
svetlana.yun@samsung.com

SAMSUNG