C语言

第8讲：指针基础知识

主讲：赵岩　指导老师：王宇颖

哈尔滨工业大学软件学院

May 21, 2011

目录

目录

目录

About pointer

- 指针很难吗？
- 指针很繁琐。
- 指针很灵活。
- 指针很强大。
  - 指针可以指向任意一个地方，并且可以通过指针修改那个地方的值。

About pointer

- 指针很难吗？
- 指针很繁琐。
- 指针很灵活。
- 指针很强大。
  - 指针可以指向任意一个地方，并且可以通过指针修改那个地方的值。

About pointer

- 指针很难吗？
- 指针很繁琐。
- 指针很灵活。
- 指针很强大。
  - 指针可以指向任意一个地方，并且可以通过指针修改那个地方的值。

About pointer

- 指针很难吗？
- 指针很繁琐。
- 指针很灵活。
- 指针很强大。
  - 指针可以指向**任意**一个地方，并且可以通过指针**修改**那个地方的值。

内存

- 计算机内的存储部件，运行时的指令和数据都保存在这里。
  - 为什么要放到内存中？
  - 如果内存不够怎么办？
- 速度快，但是掉电即失(挥发性)。
- 内存的基本构成单位为字节。
- C语言中的变量，按照不同的数据类型在内存中占用不同的字节空间。

内存

- 计算机内的存储部件，运行时的指令和数据都保存在这里。
  - 为什么要放到内存中？
  - 如果内存不够怎么办？
- 速度快，但是掉电即失(挥发性)。
- 内存的基本构成单位为字节。
- C语言中的变量，按照不同的数据类型在内存中占用不同的字节空间。

内存

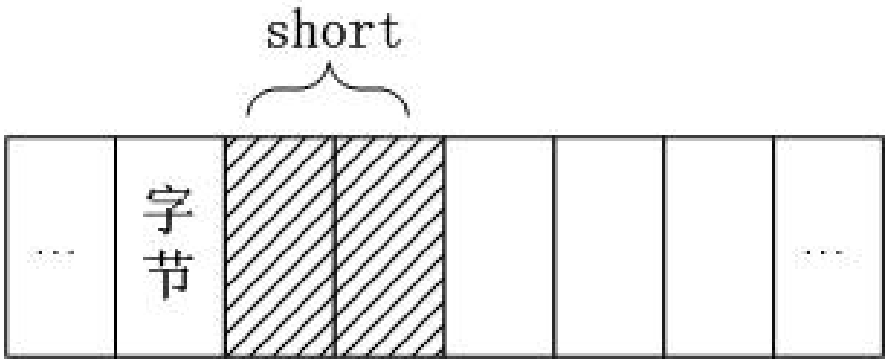
- 计算机内的存储部件，运行时的指令和数据都保存在这里。
  - 为什么要放到内存中？
  - 如果内存不够怎么办？
- 速度快，但是掉电即失(挥发性)。
- 内存的基本构成单位为字节。
- C语言中的变量，按照不同的数据类型在内存中占用不同的字节空间。

内存

- 计算机内的存储部件，运行时的指令和数据都保存在这里。
  - 为什么要放到内存中？
  - 如果内存不够怎么办？
- 速度快，但是掉电即失(挥发性)。
- 内存的基本构成单位为字节。
- C语言中的变量，按照不同的数据类型在内存中占用不同的字节空间。

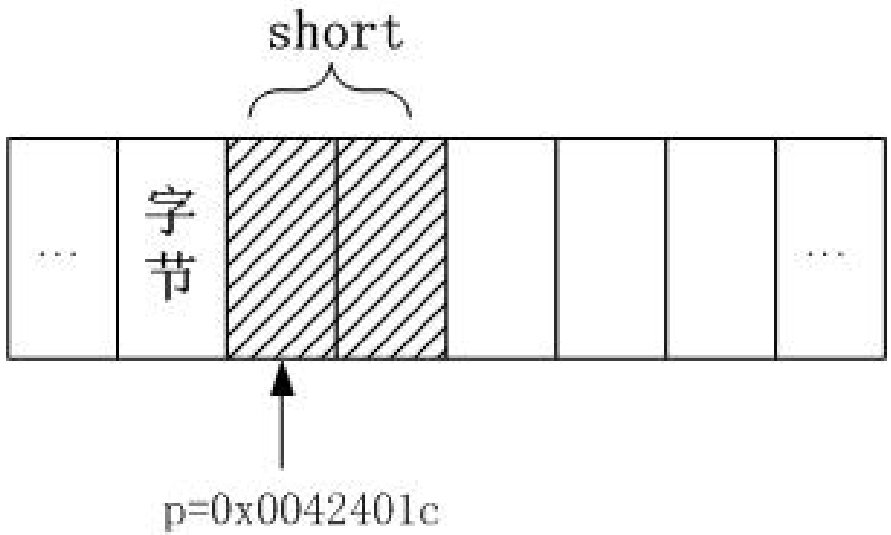
内存

- 计算机内的存储部件，运行时的指令和数据都保存在这里。
  - 为什么要放到内存中？
  - 如果内存不够怎么办？
- 速度快，但是掉电即失(挥发性)。
- 内存的基本构成单位为字节。
- C语言中的变量，按照不同的数据类型在内存中占用不同的字节空间。



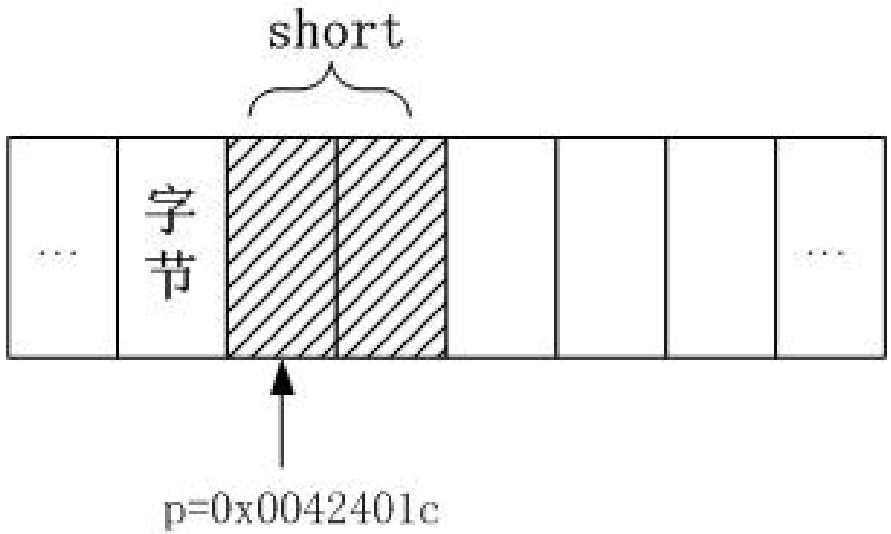
内存地址

- 地址位数为32位或64位。
- 地址通常表示成16进制数。
- 保存地址的变量–指针变量



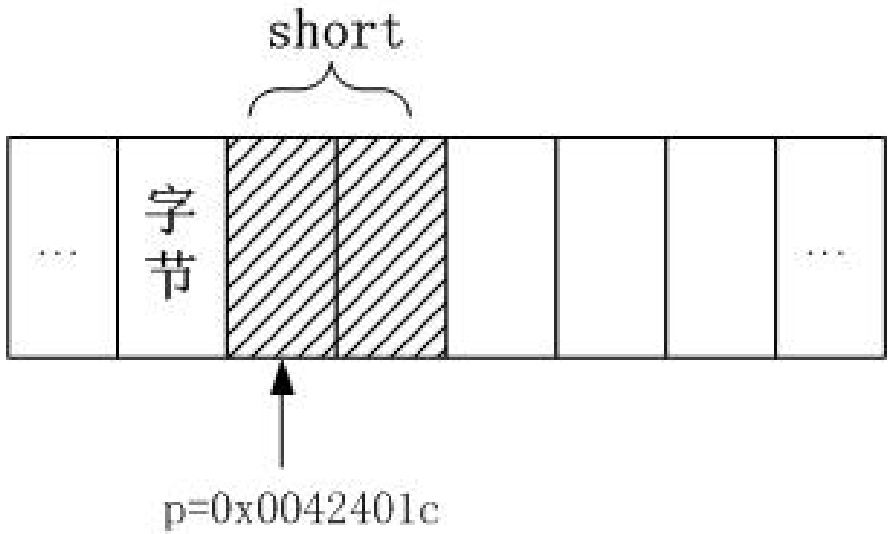
内存地址

- 地址位数为32位或64位。
- 地址通常表示成16进制数。
- 保存地址的变量–指针变量



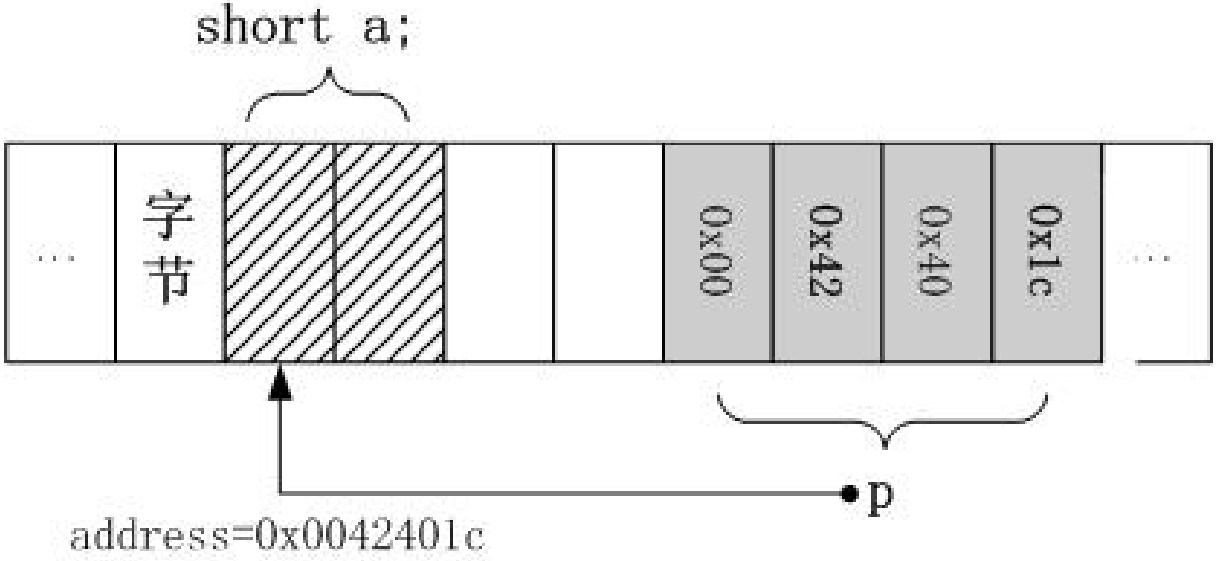
内存地址

- 地址位数为32位或64位。
- 地址通常表示成16进制数。
- 保存地址的变量–**指针变量**



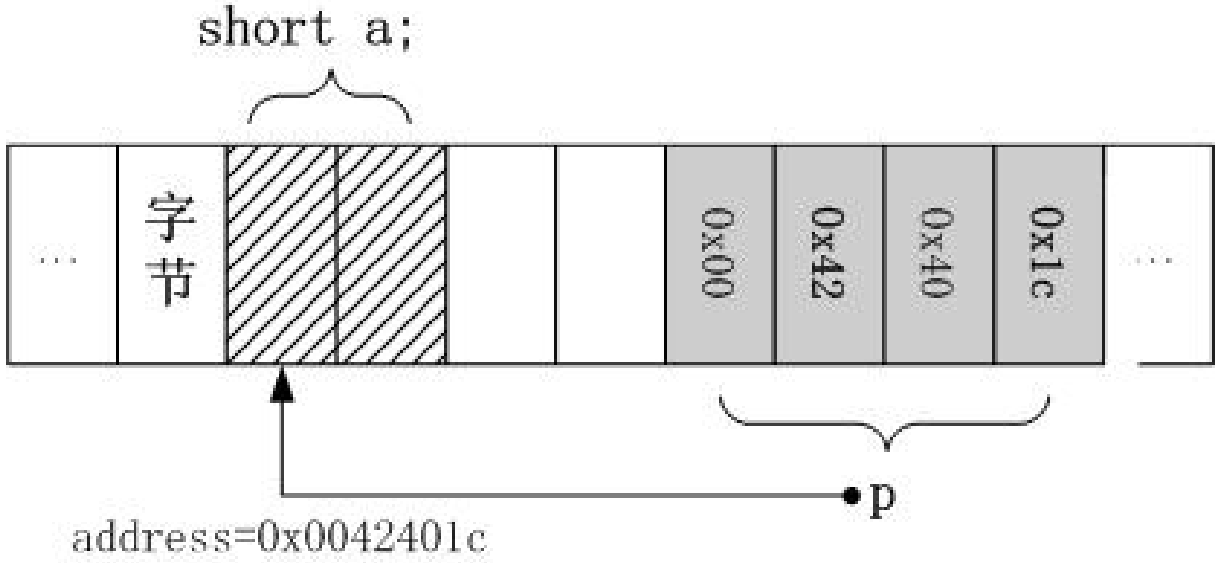
直接寻址和间接寻址

- 通过变量名直接读写内容。
- 通过指针变量间接读写内容。
  - 指针变量保存的是地址。
  - 指针变量占用四个字节（32位）
  - 指针变量有数据类型信息。



直接寻址和间接寻址

- 通过变量名直接读写内容。
- 通过指针变量间接读写内容。
  - 指针变量保存的是地址。
  - 指针变量占用四个字节（32位）
  - **指针变量有数据类型信息。**



指针的基本用法

- 声明一个指针
- 地址运算符（取地址运算符）&。
- 指针运算符（取内容运算符）*。

Demo 1: A simple example about a pointer

```
1       short a = 10;
2       short *p; /*declare a pointer variable p*/
3       p = &a; /*get address of variable a */
4       *p=15; /*Indirect Addressing */
5       printf ("value of p is %p\n", p);
6       printf ("size of p is %d\n", sizeof(p));
7       printf ("value pointed by p is %d\n", a);
```

指针的基本用法

- 声明一个指针
- 地址运算符（取地址运算符）&。
- 指针运算符（取内容运算符）*。

Demo 1: A simple example about a pointer

```
1       short a = 10;
2       short *p; /*declare a pointer variable p*/
3       p = &a; /*get address of variable a */
4       *p=15; /*Indirect Addressing */
5       printf ("value of p is %p\n", p);
6       printf ("size of p is %d\n", sizeof(p));
7       printf ("value pointed by p is %d\n", a);
```

指针的基本用法

- 声明一个指针
- 地址运算符（取地址运算符）&。
- 指针运算符（取内容运算符）*。

Demo 1: A simple example about a pointer

```
1       short a = 10;
2       short *p; /*declare a pointer variable p*/
3       p = &a; /*get address of variable a */
4       *p=15; /*Indirect Addressing */
5       printf ("value of p is %p\n", p);
6       printf ("size of p is %d\n", sizeof(p));
7       printf ("value pointed by p is %d\n", a);
```

指针声明的注意事项

- “类型 *”是一个整体。
- 后面接指针变量名。

指针声明的注意事项

- “类型 *”是一个整体。
- 后面接指针变量名。

指针声明的注意事项

- “类型 *”是一个整体。
- 后面接指针变量名。

Ambigious *

```
1       short *p; /*declare a pointer variable*/
2       short a, *p;
3       *p ;    /*get value pointed by p*/
4       i * j; /*i times j*/
5       i**p; /*- what is this?*/
6       *p**p; /*- and then? */
7       *p/*p; /*-My God!*/
```

取值符和取址符

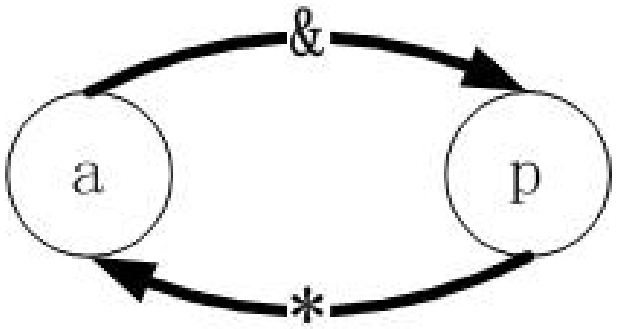
- 指针运算符作用于指针。
- 地址运算符作用于变量。

取值符和取址符

- 指针运算符作用于指针。
- 地址运算符作用于变量。

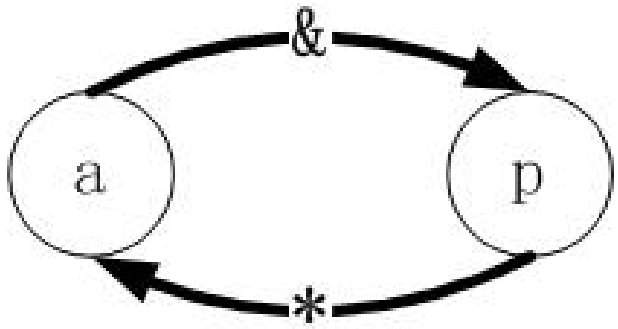
取值符和取址符

- 指针运算符作用于指针。
- 地址运算符作用于变量。



取值符和取址符

- 指针运算符作用于指针。
- 地址运算符作用于变量。




```
1       int a = 5
2       short *p = &a;
3       printf("%d", *(&(*(&a))));
```

指针运算

- 关系运算
  - <，>，==
- 算术运算
  - 加减运算+,-;
  - 自加减运算++，−;
- 只支持这两大类运算

```
1       int a = 30;
2       int *p = &a;
3       p*2 /*-How to understand this ?-*/
4       p/2 /*- ??? -*/
```

指针运算

- 关系运算
  - <，>，==
- 算术运算
  - 加减运算+,-;
  - 自加减运算++，−;
- 只支持这两大类运算

```
1       int a = 30;
2       int *p = &a;
3       p*2 /*-How to understand this ?-*/
4       p/2 /*- ??? -*/
```

指针运算

- 关系运算
  - <，>，==
- 算术运算
  - 加减运算+,-;
  - 自加减运算++，−;
- 只支持这两大类运算

```
1       int a = 30;
2       int *p = &a;
3       p*2 /*-How to understand this ?-*/
4       p/2 /*- ??? -*/
```

指针运算实例

Demo 2: Operators of a pointer

```
1       int *p1 = malloc( sizeof(int)*5 );
2       if( p1 != NULL){
3           *p1++ = 10;
4           /*++作用于p1上还是*p1上？（结合性）*/
5           /* *p1=10还是*(p1+1) = 10?（后缀运算符）  */
6           /*- bad style */
7
8           *(p1+3)=20;
9       }
10      free(p1); /*free memeory*/
11      p1 = NULL; /* Good habit */
```

数组实现-颠倒字符串

Reverse a string

```
1       char a[200] = "abcde";
2       int len = strlen(a);
3       char temp;
4       int i ;
5       for(i = 0; i<len/2 ;i++){
6           temp = a[i];
7           a[i] = a[len-1-i];
8           a[len-1-i] = temp;
9       }
10      printf("%s\n",a);
```

指针运算实例2-颠倒字符串

Demo 3: Pointer implementation

```
1       char a[200] = "abcde";
2       int len = strlen(a);
3       char *p1 = a;
4       char *p2 = a+len-1;
5       char temp;
6       while(p1<p2){
7           temp = *p1; *p1 = *p2; *p2 = temp;
8           p1++;
9           p2--;
10      }
11      printf("%s\n",p1);
```

指针的一宗罪–wild pointer

- 不初始化。(wild pointer)

Demo 4: wild pointer

```
1       short *p; /*declare a pointer variable*/
2       printf ("%p \n",p);
3       printf("%d",*p) ;
4       *p=5;
```

指针的一宗罪–wild pointer

- 声明时强制带等号。

```
1       short *p = NULL ; /*declare a null pointer*/
```

指针的一宗罪–wild pointer

- 声明时强制带等号。

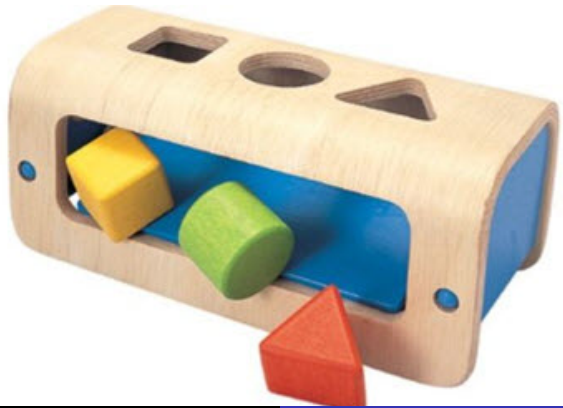
```
1       short *p = NULL ; /*declare a null pointer*/
```

指针的二宗罪–指鹿为马

- 类型不对

wrong pointer type

```
1       int a = 10;
2       int *p2 = &a;
3       short *p1 = &a; /*-error 1*/
4       p1 = p2; /*-error 2*/
5       p1 = 100; /*-error 3*/
6       printf("%d",*p1) ;
```



指针的二宗罪–指鹿为马的解决方法

- 同等类型数据
  - 地址符号
  - 其他指针
  - 数组
  - 申请的内存
- 空指针
  - NULL

指针赋值实例

Assignment of a pointer

```
1  int *p1,*p2;
2  int a = 10;
3  int array[]={1,2,3};
4  p2 = &array[0];
5  p1=&a; /*地址符号*/
6  p1=p2+1; /*其他指针*/
7  p1=array; /*数组*/
8  p1=(int *)malloc(sizeof(int)); /*申请的内存*/
9  p1=NULL; /*空指针*/
```

指针的三宗罪–指针越界

- 多发生于字符串或数组的越界。

Demo 5: Out of bound 1

```
1       int age[3] ={28,30,33};
2       int *p = &age[0];
3       int index ;
4       scanf("%d",&index);
5       *(p+index) = 40;
```

指针的三宗罪–指针越界

Demo 6: Out of bound 2

```
1       int salary[3]={1000,1500,2000};
2       int age[3] ={28,30,33};
3       int *p = &age[0];
4       int index ;
5       scanf("%d",&index);
6       *(p+index) = 40;
7       printf("%d",salary[0]);
```

指针的三宗罪–指针越界的解决方法

- **细心**。
- 确定指针所指的**位置**。
- 必要的**校验**语句。

```
1       char a[200] = "abcde";
2       int age[3] ={28,30,33};
3       int *p = &age[0];
4       int index ;
5       scanf("%d",&index);
6       if(index>=0&& index<3){
7           *(p+index) = 40;
8       }
9       printf("%d",salary[0]);
```

思考与实践

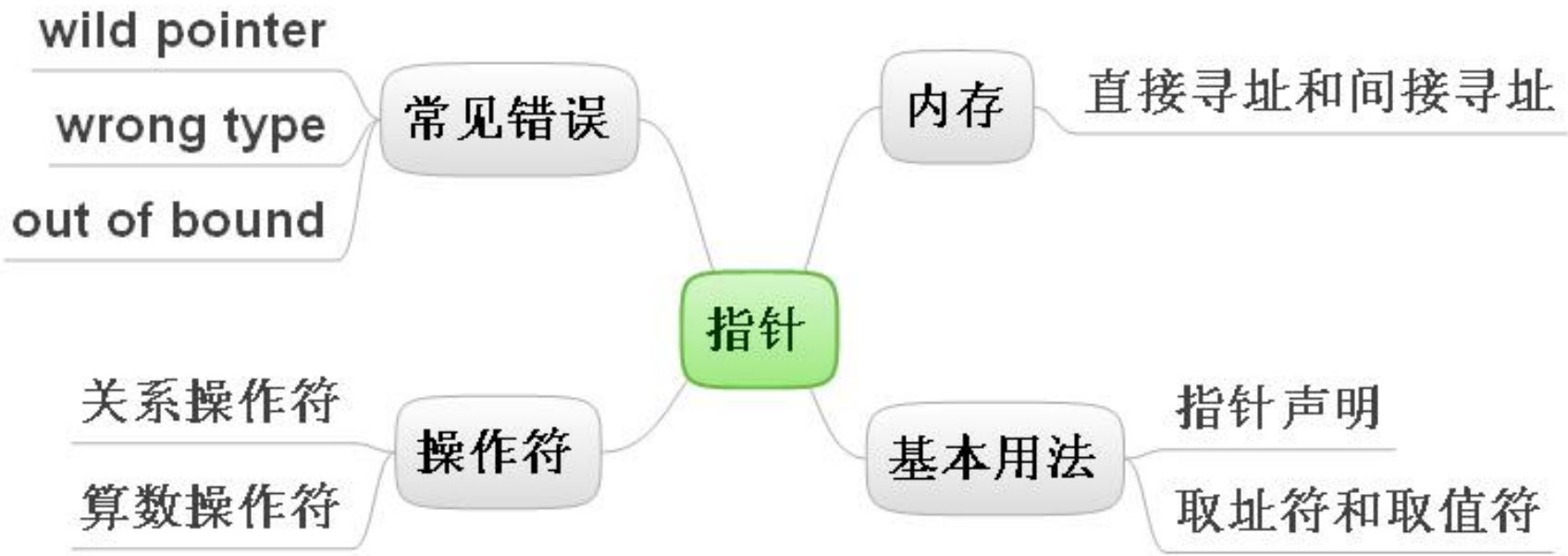
问题

*debug*版本与*release*版本有什么区别？

问题

*demo5*与*demo6*在*release*版本下会发生什么事情？

总结


指针
常见错误
wild pointer
wrong type
out of bound
内存
直接寻址和间接寻址
操作符
关系操作符
算数操作符
基本用法
指针声明
取址符和取值符

谢谢大家，欢迎提问！