第七章 指针(一)

一、指针的定义

二、指针变量

三、访问指针指向的变量

四、声明指针变量

五、未被初始化的指针

六、空指针

一、指针的定义

1.1 地址与指针

变量的本质是什么？ 内存中的存储单元，即每一个变量都需要在内存中分配一块空间来保存它的值，如 int a = 5;编译器需要在内存中分配4个字节来存储5这个值。

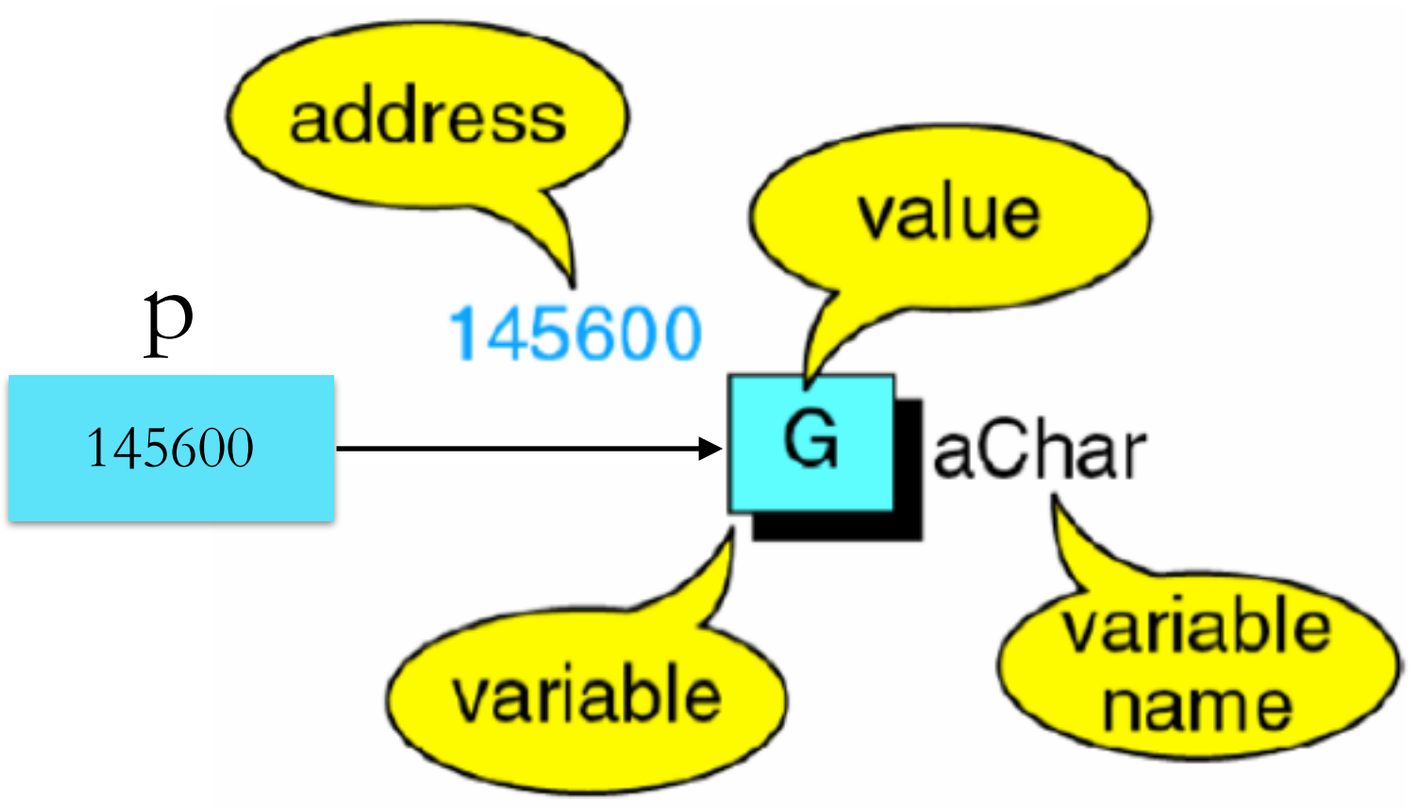
数据保存在内存中，而每一块内存空间都有一个编号，称为内存地址，如 0x12FF3C，用来存储这个地址编号的变量称为指针变量。

通过指针可以访问和处理指针所指向的变量，增加了访问数据的手段，使程序更加灵活。

1.2 指针的概念

指针本身也是一个变量，它存储的是另一个变量的地址。

存放变量的地址的变量是指针变量，因此，一个指针变量的值就是某个变量的地址。为了表示指针变量和它所指向的变量之间的关系，在程序中用“*”符号表示“指向”。




```
char aChar = 'G';
char *p = &aChar;
```

- p 表示指针变量本身的名字；
- * 表示p是一个指针变量，可以存储一个变量的地址；
- & 表示取地址运算符，返回变量的地址编号；

1.3 输出指针变量

```
/* Print character addresses */
#include <stdio.h>

int  main  (void)
{
/* Local Definitions */
   char   a;
   char   b;
/* Statements */
   printf ("%p %p\n", &a, &b);

   return 0;
}  /* main */
```

(1) &：取地址符，如&a表示获取变量a在内存中的地址；

(2) %p：用于打印指针变量的值；

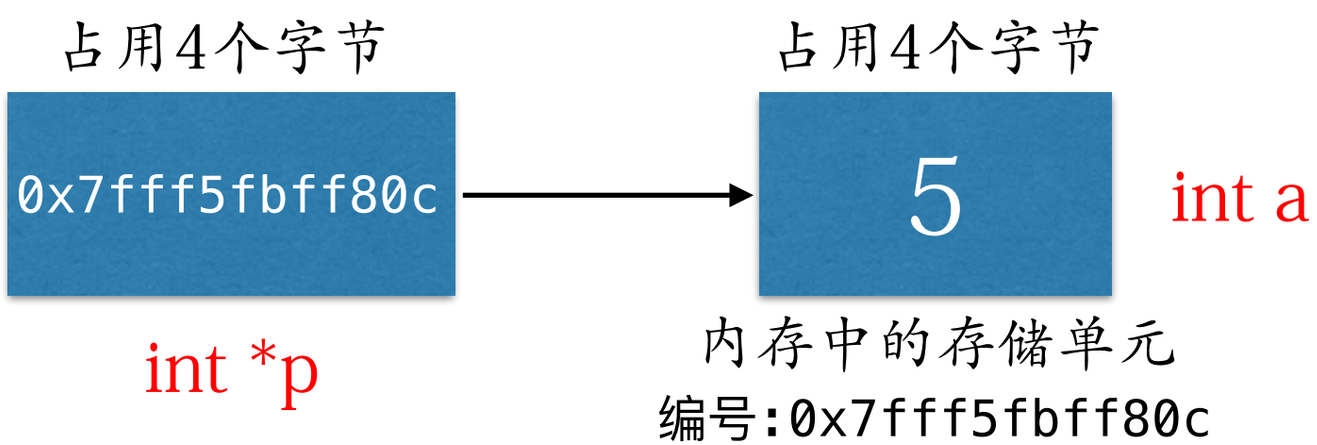
(3) 所有的指针变量在内存中都是占用4个字节；

二、指针变量

指针变量是用于存储另一变量的地址，它具有以下四个特点：

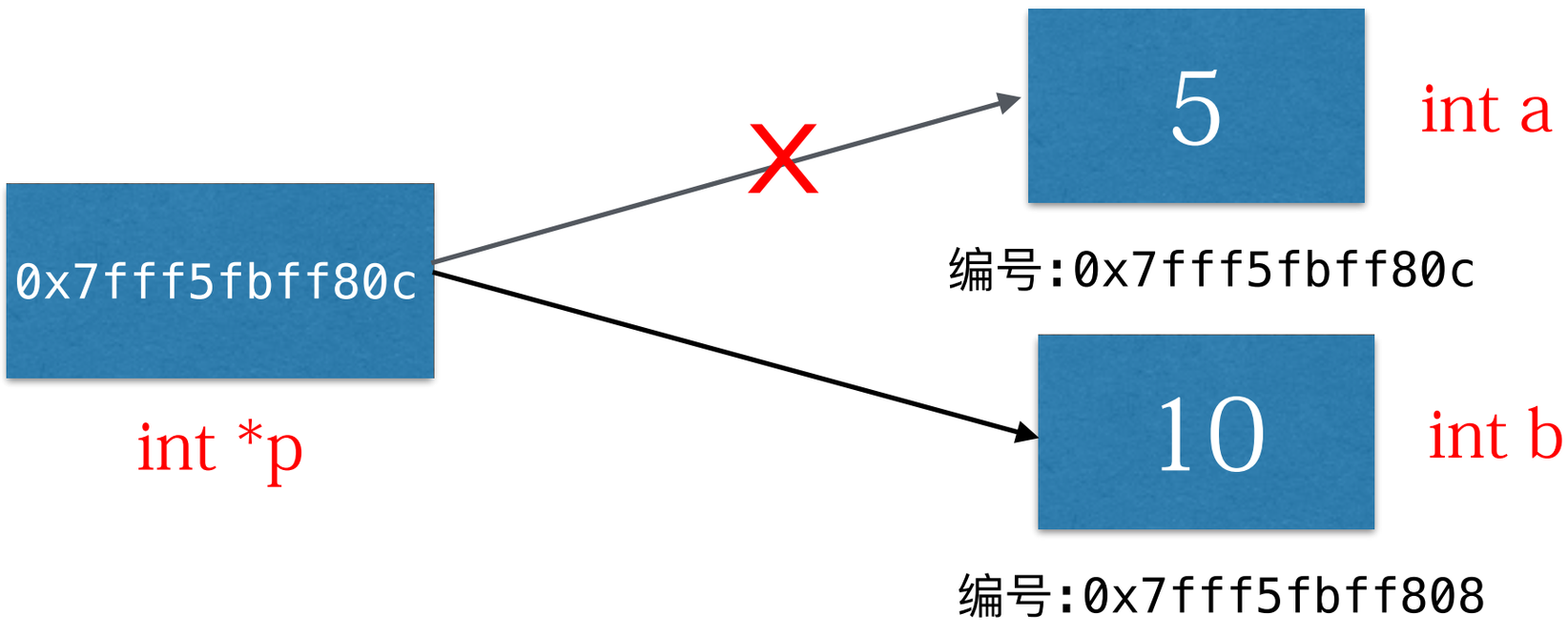
⑴ 它的值是一个地址；

```c
#include <stdio.h>

int main(int argc, const char * argv[]) {
    int a = 5;
    int *p = &a;//指针变量p存储的是a的地址
    printf("p = %p\n",p);
    printf("&a = %p\n",&a);
    return 0;
}
```



(2) 指针变量的值可以改变，存储一个新的地址，即指向别的变量；

```c
#include <stdio.h>
int main(int argc, const char * argv[]) {
    int a = 5;int b = 10;
    int *p = &a;//指针指向变量a
    printf("p = %p\n",p);//p = 0x7fff5fbff80c

    p = &b;      //指针的值改变,指针指向别的变量
    printf("p = %p\n",p);//p = 0x7fff5fbff808
    return 0;
}
```



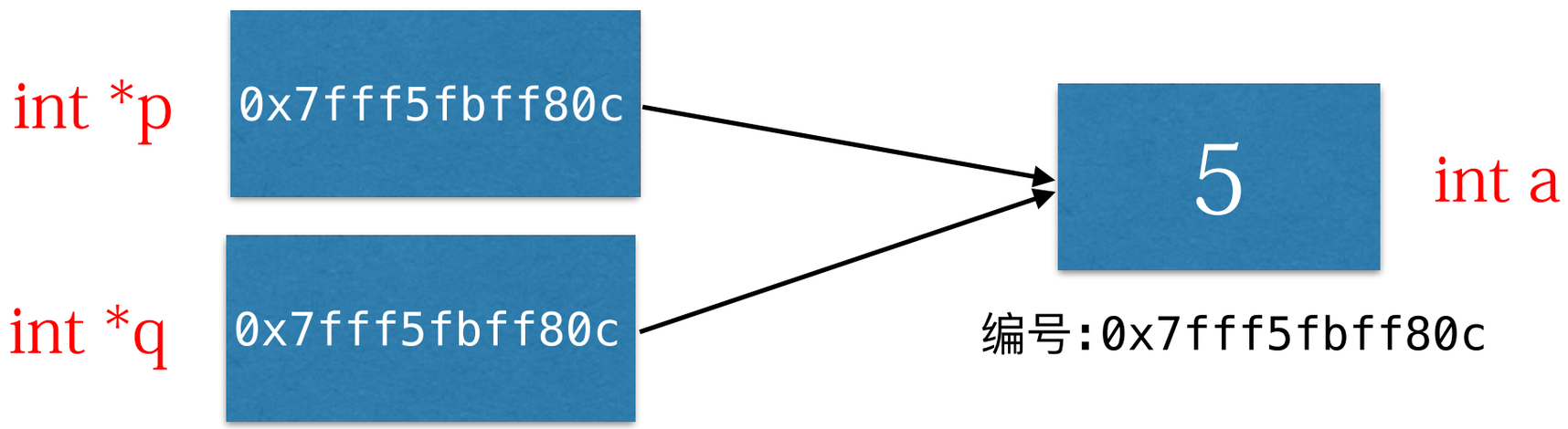
(3) 指针指向一个特定的类型，即指针本身也是有类型的；

```c
#include <stdio.h>

int main(int argc, const char * argv[]) {
    int a = 5;
    int *p = &a;//指针指向整型变量a
    printf("p = %p\n",p);
    return 0;
}
```

上述代码中指针变量指向int类型的变量a，所以声明指针p的类型也为int类型，即声明指针的类型和指针所指向的变量的类型保持一致。

(4) 多个指针变量可以指向同一个值（变量）；

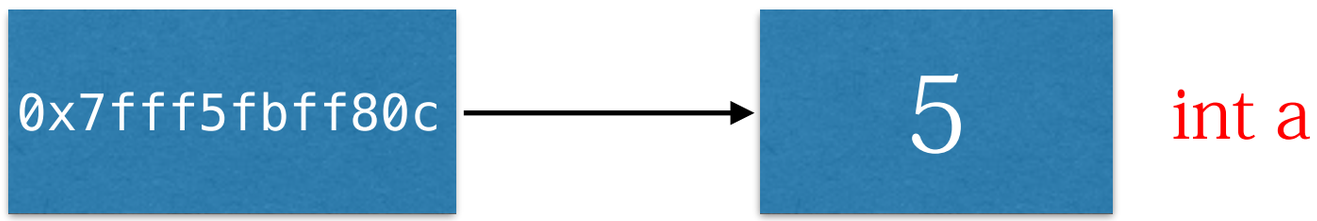
```c
#include <stdio.h>

int main(int argc, const char * argv[]) {
    int a = 5;
    int *p;
    int *q;
    p = &a;//指针p指向变量a
    q = &a;//指针q也指向变量a
    printf("p = %p\n",p);//p = 0x7fff5fbff80c
    printf("q = %p\n",q);//q = 0x7fff5fbff80c
    return 0;
}
```



三、访问指针指向的变量

通过指针变量可以间接访问指针所指向的变量的值。

对于右图所示的存储结构，想要访问内存中的5这个值，有两种方式：



(1) 通过变量名a;

(2) 通过*p对指针解引用;

```c
#include <stdio.h>

int main(int argc, const char * argv[]) {
    int a = 5;
    int *p = &a;
    printf("%d,%d\n",a,*p);
    return 0;
}
```

间接运算：

把运算符*放在指针名前面就可以访问指针所指向的变量

设已定义好整型变量a和整型指针pa，且pa指向变量a，则

printf(“%d”,*pa);//输出pa指向变量的值，即输出a的值

*pa = 100;//给pa所指向变量赋值100，即给a赋值100，等价于a = 100;

课堂练习：

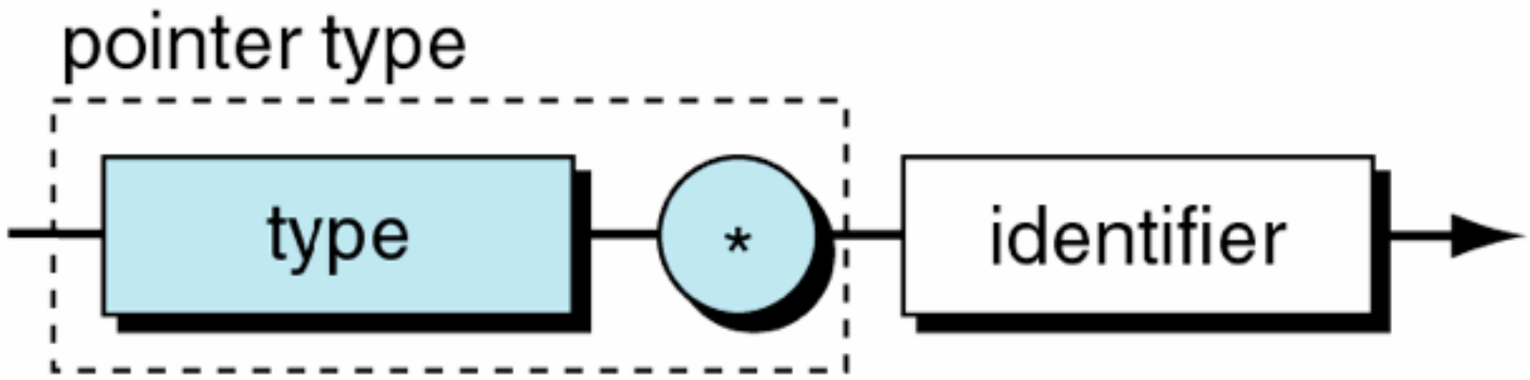
```c
#include <stdio.h>

int main(int argc, const char * argv[]) {
    int a = 5;
    int *p = &a;

    *p += 3;
    printf("a = %d\n",a);
    return 0;
}
```

程序运行结果如下：

a = 8

结论：指针指向谁，解引用就是谁！

四、声明指针变量

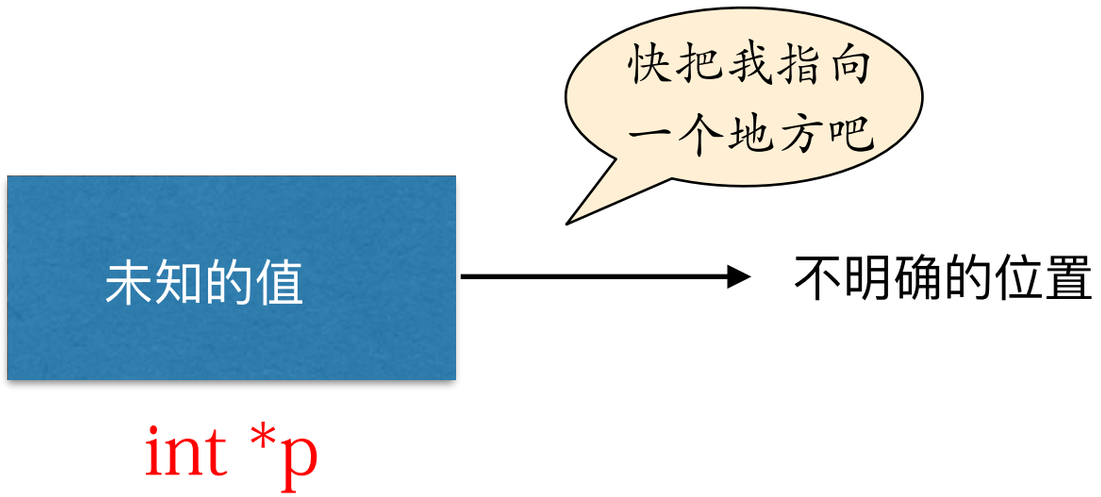



```
Examples:
  char *p;
  int *q;
  float *r;
  long double *s;
  long int *t;
```

注意：声明指针变量的类型应和指针所指向的变量的类型保持一致。

五、未被初始化的指针

任何指针变量刚被创建时不会自动成为NULL指针，它的缺省值是随机的，它会乱指一气。如定义 int *p；则指针变量p未被初始化，它的指向是不明确的。



注意：因为野指针指向的内存空间位置不明确，因此不要对野指针解引用。

当一个指针成为“野指针”时，它指向哪里就是不可预知的了。当使用这个野指针时，即使程序运行没有问题，那也是非常危险的，因为这个”野指针”指向的内存空间，可能是某个重要的数据或其它程序，甚至是系统的重要内存位置，这样造成的危害是不可预知的，这个不可预知包括危害程度的不可预知和危害时间的不可预知的，像一颗不知何时会爆的定时炸弹.

六、空指针

指针变量在创建的同时应当被初始化，要么将指针设置为NULL，要么让它指向合法的内存，否则它就是一个野指针。所以当我们不知道应该将指针指向何处时，应将指针置为NULL。

```
int *p = NULL;
```

空指针是指向为NULL，它的值是0x0，即0号地址单元，永远不要试图去访问空指针指向的内容，因为它是系统内存单元，不允许用户级别的程序去访问，即对空指针解引用是非法的，会造成segmentation fault（段错误）。

空指针与野指针的区别：

野指针：未被初始化的指针，里面的内容是垃圾地址，它的值是不明确的；

空指针：被初始化为NULL的指针，里面的地址是0；

注意：不要解引用空指针或野指针。

The End

渥瑞达·教学部