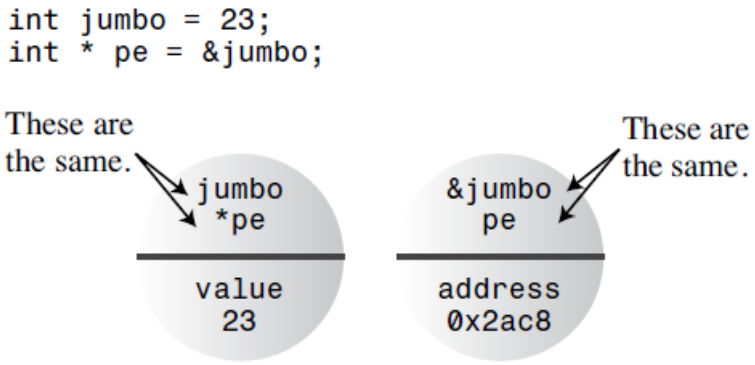
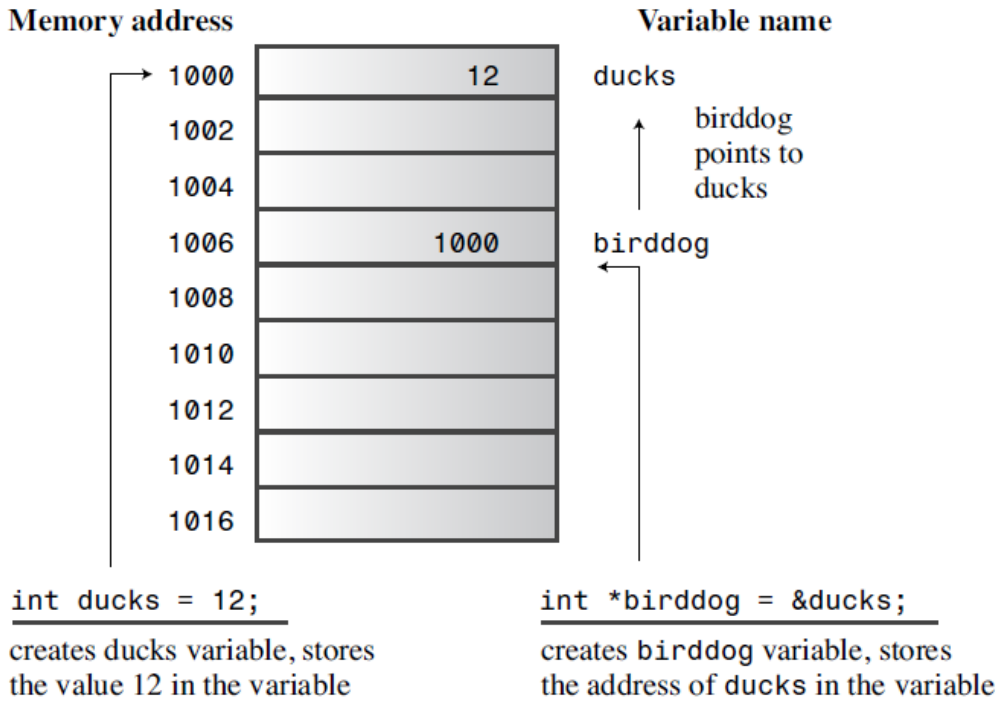
指针变量

- 指针也是一种数据类型，指针变量也是一种变量
- 指针变量指向谁，就把谁的地址赋值给指针变量
- `*`操作符操作的是指针变量指向的内存空间
- 使用`sizeof()`测量指针的大小，取决于操作系统




```
char ch = 'A';
int a = 1;

printf("%p %p\n", &ch, &a);
```




```
int a = 1;
int* b = &a;
*b = 2;
cout << a << " " << *b << endl;
```

野指针和空指针

野指针：C++中创建指针时，计算机将分配用来存储地址的内存，但是不会分配用来存储指针所指向数据的内存。比如：p指针指向的空间是不确定的，通过间接操作修改了p指向空间的数据，很可能是操作系统中重要的数据，就发生不可预知的危险，因此，声明指针时一定要初始化。

```
int *p;
*p = 100;
cout << *p << endl;
```

空指针：野指针和有效指针变量保存的都是数值，为了标志此指针变量没有指向任何变量(空闲可用)，C语言中，可以把NULL赋值给此指针，这样就标志此指针为空指针，没有任何指针。

- `int *p = NULL`
- NULL是一个值为`0`的宏常量：`#define NULL((void *)0)`

万能指针`void *`

有时候，一个指针根据不同的情况，指向的内容是不同类型的值，我们可以先不明确定义它的类型，只是定义一个`无类型的指针`，以后根据需要再用强制类型转换的方法明确它的类型。

```
#include <iostream>
using namespace std;

int a = 10;
double b = 3.5;
void* p;
int main(){
    p = &a;
    cout << *(int*)p << endl;
    p = &b;
    cout << *(double*)p << endl;
    cout << *(long long*)p << endl;
    return 0;
}

输出：
10
3.5
4615063718147915776
```

const修饰的指针变量

```
int a = 100, b = 200;

// 指向常量的指针
// 修饰*，指针指向可以改变，指针指向内存区域不能修改
const int * p1 = &a;
// *p1 = 111;
p1 = &b;
cout << *p1 << endl;

```

```cpp
// 指针常量
// 修饰p2，指针指向内存区域可以改变，指针指向不可以改变
int * const p2 = &a;
// p2 = &b;
*p2 = 222;
cout << *p2 << endl;
```

指针和数组

```cpp
#include <iostream>
#include <cstdio>
using namespace std;

int a[5];
int* pa = a;

int main() {

    for(int i = 0; i < 5; i++)
        scanf("%d", a + i);

    for(int i = 0; i < 5; i++)
        printf("a[%d]=%d\n", i, *(a + i));

    return 0;
}
```

```cpp
#include <iostream>
#include <cstdio>
using namespace std;

int a[5];
int* pa = a;

int main() {

    for(int i = 0; i < 5; i++)
        scanf("%d", a + i);

    for(int i = 0; i < 5; i++){
        printf("%d ", *pa);
        pa++;
    }

    return 0;
}
```

指针数组

数组的每个元素都是指针类型。

```cpp
#include <iostream>
#include <cstdio>
using namespace std;

int main() {

    int a = 1, b = 2, c = 3;
    int* p[] = {&a, &b, &c};

    for(int i = 0; i < sizeof(p) / sizeof(p[0]); i++)
        cout << *p[i] << " ";

    return 0;
}
```

多重指针

```cpp
#include <cstdio>

int a = 10;
int* p;
int** pp;   // 定义双重指针
int*** ppp; // 定义三重指针

int main() {
    p = &a;     // 将p指向a
    pp = &p;    // 将pp指向p
    ppp = &pp;  // 将ppp指向pp
    printf("a=%d=%d=%d\n", *p, **pp, ***ppp);
    return 0;
}
```

引用

> 引用是给变量起别名，比指针更加简洁。

```cpp
#include <iostream>
#include <cstdio>
using namespace std;

int main() {

    // 引用的本质是常指针，因此必须初始化
    int a = 10;
    int& b = a; // int* const b = &a;

    b = 20; // *b = 20;

    cout << a << " " << b << endl;

    return 0;
```

```
}
```

```cpp
#include <iostream>
#include <cstdio>
using namespace std;

int f(int& x){
    x *= 2;
    return x * 3;
}

int main() {

    int a = 10;
    cout << f(a) << " " << a << endl;

    return 0;
}
```

数组引用

```cpp
int a[5] = { 1, 2, 3, 4, 5 };
int(&aref)[5] = a;

aref[0] = 6;

for (int i = 0; i < 5; i++) cout << a[i] << " "; // 6 2 3 4 5
```

函数参数实现变量交换

```cpp
#include <iostream>
#include <cstdio>
using namespace std;

void swap1(int a, int b){
    int t = a;
    a = b;
    b = t;
    cout << "函数内部: " << a << " " << b << endl;
}

void swap2(int* a, int* b){
    int t = *a;
    *a = *b;
    *b = t;
}

void swap3(int& a, int& b){
    int t = a;
```

```
    a = b;
    b = t;
}

int main() {

    int a = 1, b = 2;

    // swap1(a, b);
    // swap2(&a, &b);
    swap3(a, b);

    cout << "函数外部：" << a << " " << b << endl;

    return 0;
}
```

动态开辟空间

```
#include <iostream>
#include <cstdio>
using namespace std;

/*
const int N = 1e8 + 10;
int a[N];
*/

int main() {

    int* p = new int(3);
    cout << *p << endl;

    delete p;   // 释放空间

    cout << *p << endl;

    int* p2 = new int;
    cout << *p2 << endl;

    return 0;
}
```

```
#include <iostream>
#include <cstdio>
using namespace std;

int main() {

    int n;
    cin >> n;
    int* a = new int[n + 1];
```

```
10
11      for(int i = 1; i <= n; i++)
12          cin >> a[i];
13
14      for(int i = 1; i <= n; i++)
15          cout << a[i] << " ";
16
17      delete [] a;
18
19      return 0;
20  }
```

数组做函数参数

```
1   #include <iostream>
2   #include <cstdio>
3   using namespace std;
4
5   int n;
6   int a[110];
7
8   void f(const int a[]){
9       // a[1] *= 2;
10      for(int i = 1; i <= n; i++)
11          cout << a[i] << " ";
12  }
13
14  void f2(const int* a){
15      // a[1] *= 2;
16      for(int i = 1; i <= n; i++)
17          cout << a[i] << " ";
18  }
19
20  int main() {
21
22      cin >> n;
23
24      for(int i = 1; i <= n; i++)
25          cin >> a[i];
26
27      f(a);
28
29      cout << endl << "---------------" << endl;
30
31      for(int i = 1; i <= n; i++)
32          cout << a[i] << " ";
33
34      return 0;
35  }
```

字符串指针

```cpp
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;

int main() {

    // char str[] = "hello cat";
    const char * str = "program";

    printf("%s\n", str);
    cout << str << endl;

    str = "hello";

    printf("%s\n", str);
    cout << str << endl;

    int len = strlen(str);
    for(int i = 0; str[i]; i++)
        cout << *(str + i);

    // str[0] = 'z';

    return 0;
}
```