```cpp
//(1.2) decltype后的圆括号中非变量（是表达式）
//decltype会返回表达式的结果对应的类型。
decltype(8) kkk = 5;  //kkk = int
int i = 0;
int *pi = &i;
int &iy = i;
decltype(iy + 1); //j = int,因为iy+1得到一个整型表达式
decltype(pi) k;  //k = int *: pi是个变量。
*pi = 4;
decltype(*pi) k2 = i; //k2 = int &;
                      //*pi是指针pi所指向的对象,而且能够给这个对象赋值，所以*pi是个左值。
                        //*pi是个表达式不是个变量，因为它有*号，
                        //如果表达式结果能够作为赋值语句左边的值,那decltype后返回的就是个引用。
                        //所以这种情况要专门记：decltype后边是个非变量的表达式，并且表达式能够作为等号左边内容，，返回的就是一个类似int &一个左值引用。
```

```cpp
//decltype(i) k3;   //k3 = int,i只是个变量

//decltype((i)) iy3 = i;  //如果在变量名外额外增加了一层或者多层括号()，那么编译器就会把这个变量当成一个表达式，
//                        //又因为 变量(表达式)可以作为等号左边的内容；最终iy3 = int &
//(i) = 6;
////结论decltype((变量))的结果永远都是个引用。decltype(变量)
```

```cpp
//(1.3) decltype后的圆括号中是函数
decltype(testf()) tmpv = 14; //tmpv的类型就是函数testf()的返回类型 int。
                             //这里编译器没有去调用过函数testf()。只是使用函数testf()的返回值类型作为tmpv的类型。
decltype(testf) tmpv2;  //tmpv2 = int(void)，这个有返回类型，有参数类型，代表一种可调用对象。
//标准库function用法，类模板：
function <decltype(testf)> ftmp = testf;  //声明了一个function(函数)类型，用来代表一个可调用对象。
        //它所代表的可调用对象是一个int(void);
cout << ftmp() << endl;  //10
```

```cpp
//(2.2)通过变量表达式抽取变量类型
vector<int> ac;
ac.push_back(1);
ac.push_back(2);
vector<int>::size_type mysize = ac.size();
cout << mysize << endl; //2
decltype(ac)::size_type mysize2 = mysize; //抽取ac的类型 vector<int>,所以这行相当于 vector<int>::size_type mysize2 = mysize;
```

```cpp
typedef decltype(sizeof(0)) size_t; // ==>size_t int
//typedef decltype(sizeof(int)) size_t;
//typedef unsigned int size_t;
size_t abc;
abc = 1;
```

```
//a)用于函数返回类型
//int a = 100;
//mydouble(a) = 20; //int &
//                    //把mydouble的返回类型修改为auto后,那么mudouble返回的类型就是int(右值);
//                          //不能给右值赋值，会提示 ：“=”：左操作数必须为左值
//cout << a << endl; //a = 20;
//decltype (mydouble(a)) b = a; //b = int &

//b)用于变量声明中
int x = 1;
const int &y = 1;
auto z = y; //z = int ,const和引用都没了
decltype(auto) z2 = y; //z2 = const int & ;
                    //auto丢掉的东西(const,引用)，能够通过decltype(auto)捡回来。
```