自动类型推断（auto）

能力

auto 可以认为是一个关键字，也可以理解成一个类型占位符。当我们写下 auto x = 20; 这条语句时，在编译期会由编译期自动地对 x 进行类型推断，并使用推断结果来替换掉 auto 这一占位符

因此，auto 定义变量必须立即初始化，这样编译器才能推断出它的实际类型。并且，自动推断发生在编译期，不会影响运行时性能

auto 常规类型推断

常规类型推断应该是我们在日常开发中使用最多的一种自动类型推断，通常我们会在 for 循环中使用

string s = "Hello";

for (auto c : s) { /* ..... */ }

在 for-each 遍历中我们可以使用 auto 来自动地对遍历内容进行类型推断，在该例中 c 被自动的推断成 char 类型

```
using boost::typeindex::type_id_with_cvr;

string s = "Hello";
for (auto c : s) {
    cout << type_id_with_cvr<decltype(c)>().pretty_name() << endl;
}
```

借助 boost 库我们可以非常方便的查看推断结果

map、multimap 等 key-value 容器使用 auto

类似于 key-value 的容器在迭代时可能会比较复杂，我们可以使用迭代器迭代，也可以使用 for-each 的方式进行迭代

```
map<string, string> stu = {"name", "smart"};

for (const auto &item : stu) {
        cout << type_id_with_cvr<decltype(item)>().pretty_name() << endl;
    }
```

这时候会发现 stdout 会输出很长一串儿，如果我们自己写的话，emmm，非常麻烦

常见情况下的推断

值类型推断

传值方式的 auto 会抛弃掉引用和 const 限定符

```
int x = 1024;
int &y = x;
const int z = x;
```

```
auto m = x;
auto n = y;
auto u = z;
```

m 、n 和 u 的推断结果均为 int

实际上值类型的 auto 推断完全可以看作是 void (T value) 这样的函数模板，将其当成值传递就好

引用或指针类型推断

```
int x = 1024;
int &y = x;
const int z = y;

auto &m = y;
auto &n = z;
```

引用类型

auto &m = y; — y 本身的类型为 int &，因此，auto 将被替换成 int，那么 m 的类型则为 int &

auto &n = z; — **z 本身的类型为 const int，因此，auto 被替换成 const int，那么 n 的类型则为 const int &**

auto &p = q; 可以看作是 void (T& value) 这一函数模板，那么引用将会被抛弃，但是会保留 const 属性

```
const int x = 1024;

auto *m = &x;
auto n = m;
```

指针类型

auto *m = &x; — auto 将被替换成 const int，那么 m 的类型即为 const int *

auto n = m; — auto 将被替换成 const int *，那么 n 的类型即为 const int *

对于指针类型来说，既然我们能够使用另一个变量初始化，那么等号两边儿的类型就得保持一致

万能引用类型推断

auto &&p = q;

当 q 为左值时，auto 将被推断为左值引用，加之引用折叠，p 的类型也为左值引用

当 q 为左值时，auto 将被推断为右值的类型，p 的类型则为右值引用

const 属性保留

发生引用折叠的地方有两个: 万能引用作为函数模板形参，auto &&

数组和函数指针的类型推断

数组类型

当我们对一个数组变量进行 auto 自动类型推断时，数组将退化成指针

```
const char name[] = "smart";
auto s = name;
```

此时 auto 将被替换成 cosnt char *，也就是一个常量指针

函数类型

当我们对一个函数变量进行 auto 推断时，推断结果要么是指针，要么是引用

```
int func(int x, int y);

auto m = func;
auto &n = func;
```

auto m = func; — 推断结果为 int (*)(int, int)

auto &n = func; — 推断结果为 int (&)(int, int)

对 std::initializer_list 的特殊推断

在 C++11 中，我们可以使用 {}，也就是大括号来初始化变量，最常见的就是初始化 vector 。但是，我们也可以用来初始化基本类型

```
int x = {30};
int y{30};
```

x 和 y 的值均被初始化成 30

但是，如果使用 auto 的话就会出现不一样的情况

```
auto m = {30};
auto n{30};
```

此时 m 将被推断为 std::initializer_list<int>，n 则被推断为 int