C++11起引入了参数包,但是在C++11时对包展开比较繁琐

当我想要实现一个print函数,其功能是将我传入的参数全部输出,在**C++11**时需要这样实现:

```
#include <iostream>

void print() {} //需要定义一个出口函数

template<typename T, typename...Args>
void print(T a, Args...args) {
    std::cout << a << " ";
    print(args...);
}

int main() {
    print(1, 2, "test", 3); // 1 2 test 3
}
```

在参数包类型全部都相同时可以进行如下操作

```
#include <iostream>

template<typename...Args>
void print(Args...args) {
    int a[sizeof...(Args)]{args...}; // sizeof...对参数包大小进行求值
    for (auto i : a) std::cout << i << " ";
}

int main() {
    print(1, 2, 3, 4); // 1 2 3 4
}
```

C++17起引入折叠表达式,可以简化繁琐的写法

一元折叠

一元折叠分为左折叠和右折叠,示例如下

```
template<typename...Args>
decltype(auto) left(Args &&... args)//一元左折叠
{
    return (...-args);
}

template<typename...Args>
decltype(auto) right(Args &&...args)//一元右折叠
{
    return (args-...);
}

int a = left(1, 2, 3); // 等价((1-2)-3) = -4
int b = right(1, 2, 3); // 等价1-(2-3) = 2
```

利用一元折叠表达式实现print函数

```
template<typename...Args>
void print(Args &&...args)
{
    ((std::cout << args << " "), ...);
}

template<typename...Args>
void foo(Args &&...args)
{
    int a[sizeof...(args)]{(std::cout << args << std::endl, 0)...};
    // sizeof...获取包的大小,逗号表达式A,B,丢弃A的值,保留B
    for (auto i : a) std::cout << i << " ";
}
```

二元折叠

```
template<typename...Args>
decltype(auto) sum(Args &&...args)
{
    //return (args+...+1*2); 编译器报错...优先级低于*
    return (args+...+(1 * 2)); //OK
}

std::cout << sum(1, 2, 3) << std::endl;
    // 等价((1+2)+3)+(1*2) 而不是(1+1*2)+(2+1*2)+(3+1*2)
```