

这个阶段还是多关心 Cyly 的前端。

闭包我更倾向于 C++ 或 PHP 那种显式指定的风格，因为隐式的需要“解压”，作用域的推导就已经对用户造成不小的负担了（这里的用户不止包括人类，它们的推理资源很宝贵，不应该仅用于解压）

显式指定时不需要搜索当前函数能访问的所有变量，而只需要推理指定的变量在哪个作用域中。

这似乎不仅意味着让函数作为函数返回值时使用闭包特性，而是也包括了函数定义时。

这个 idea 还有很多待完善的地方，需要考虑更多材料。

总之，在确立函数和 OOP 的具体细节之后，就直接开始实现第一版 Cyly 前端。

```
int a = 500;
float b = 12.34;

func func_builder(char c) [&a, b] {

    func ret_func() [c, double d = *a + b] {
        return [c, d];
    }

    return ret_func;
}

func recursion(int n) [recursion] {
    if (n > 10 || n < 0) return n;
    return recursion(n + 1);
}

main(int argc, char** argv) [func_builder] {
    char c = 'c';
    func new_func = func_builder(c);

    List[] = new_func();
    print List[0], List[1]; // 'c', 512.34
}
```