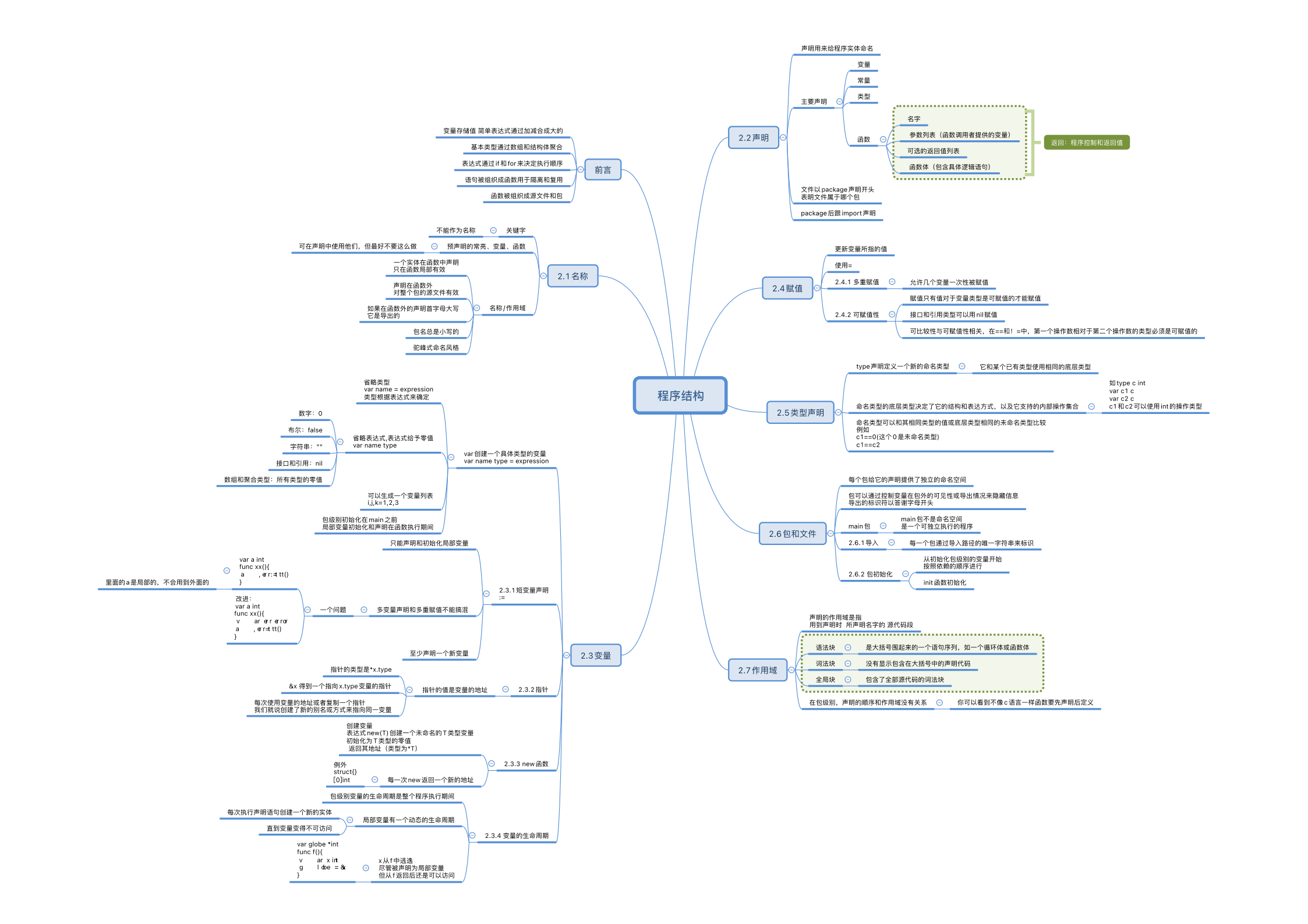
程序结构
前言
变量存储值 简单表达式通过加减合成大的
基本类型通过数组和结构体聚合
表达式通过 if 和 for 来决定执行顺序
语句被组织成函数用于隔离和复用
函数被组织成源文件和包
2.1 名称
关键字
不能作为名称
预声明的常亮、变量、函数
可在声明中使用他们，但最好不要这么做
名称/作用域
一个实体在函数中声明
只在函数局部有效
声明在函数外
对整个包的源文件有效
如果在函数外的声明首字母大写
它是导出的
包名总是小写的
驼峰式命名风格
2.2 声明
声明用来给程序实体命名
主要声明
变量
常量
类型
函数
名字
参数列表（函数调用者提供的变量）
可选的返回值列表
函数体（包含具体逻辑语句）
返回：程序控制和返回值
文件以 package 声明开头
表明文件属于哪个包
package 后跟 import 声明
2.3 变量
var 创建一个具体类型的变量
var name type = expression
省略类型
var name = expression
类型根据表达式来确定
省略表达式,表达式给予零值
var name type
数字：0
布尔：false
字符串：""
接口和引用：nil
数组和聚合类型：所有类型的零值
可以生成一个变量列表
i,j,k=1,2,3
包级别初始化在 main 之前
局部变量初始化和声明在函数执行期间
2.3.1 短变量声明 :=
只能声明和初始化局部变量
多变量声明和多重赋值不能搞混
一个问题
var a int
func xx(){
a , err := tt()
}
里面的a是局部的，不会用到外面的
改进：
var a int
func xx(){
v ar err error
a , err = tt()
}
至少声明一个新变量
2.3.2 指针
指针的值是变量的地址
指针的类型是*x.type
&x 得到一个指向 x.type 变量的指针
每次使用变量的地址或者复制一个指针
我们就说创建了新的别名或方式来指向同一变量
2.3.3 new 函数
创建变量
表达式 new(T) 创建一个未命名的 T 类型变量
初始化为 T 类型的零值
返回其地址（类型为 *T）
每一次 new 返回一个新的地址
例外
struct{}
[0]int
2.3.4 变量的生命周期
包级别变量的生命周期是整个程序执行期间
局部变量有一个动态的生命周期
每次执行声明语句创建一个新的实体
直到变量变得不可访问
var globe *int
func f(){
v ar x int
g l obe = &x
}
x从 f 中逃逸
尽管被声明为局部变量
但从 f 返回后还是可以访问
2.4 赋值
更新变量所指的值
使用=
2.4.1 多重赋值
允许几个变量一次性被赋值
2.4.2 可赋值性
赋值只有值对于变量类型是可赋值的才能赋值
接口和引用类型可以用 nil 赋值
可比较性与可赋值性相关，在==和！=中，第一个操作数相对于第二个操作数的类型必须是可赋值的
2.5 类型声明
type 声明定义一个新的命名类型
它和某个已有类型使用相同的底层类型
命名类型的底层类型决定了它的结构和表达方式，以及它支持的内部操作集合
如 type c int
var c1 c
var c2 c
c1 和 c2 可以使用 int 的操作类型
命名类型可以和其相同类型的值或底层类型相同的未命名类型比较
例如
c1==0(这个 0 是未命名类型)
c1==c2
2.6 包和文件
每个包给它的声明提供了独立的命名空间
包可以通过控制变量在包外的可见性或导出情况来隐藏信息
导出的标识符以答谢字母开头
main 包
main 包不是命名空间
是一个可独立执行的程序
2.6.1 导入
每一个包通过导入路径的唯一字符串来标识
2.6.2 包初始化
从初始化包级别的变量开始
按照依赖的顺序进行
init 函数初始化
2.7 作用域
声明的作用域是指
用到声明时 所声明名字的 源代码段
语法块
是大括号围起来的一个语句序列，如一个循环体或函数体
词法块
没有显示包含在大括号中的声明代码
全局块
包含了全部源代码的词法块
在包级别，声明的顺序和作用域没有关系
你可以看到不像 c 语言一样函数要先声明后定义