日志

ir是为了平滑从树形到riscv的线性结构

一般不会选择树形的ir

ssa静态单变量赋值形式

尽可能的拿掉ast的特征 达到线性的目的

tac 三地址码

while for的抽象都可以 用if来表示

clang -llvm

ir 无限寄存器

11. 30学习llvm

```
clang -S -emit-llvm test.c
```

生成中间代码

```
clang -S -emit-llvm -O3 test.c
```

o3优化

```
llc test.ll
```

生成汇编代码

然后用操作系统自带的汇编器和链接器生成可执行文件

这是一个基于llvm的编译器

```
.c --frontend--> AST --frontend--> LLVM IR --LLVM opt--> LLVM IR --LLVM llc--> .s
Assembly --OS Assembler--> .o --OS Linker--> executable
```

- clang

```
# 生成可执行文件
$ clang main.c -o main
# 查看编译的过程
$ clang -ccc-print-phases main.c

# 生成 tokens
$ clang -E -Xclang -dump-tokens main.c
# 生成语法树
$ clang -fsyntax-only -Xclang -ast-dump main.c
# 生成 llvm ir（不开优化）
$ clang -S -emit-llvm main.c -o main.ll -O0
```

```
# 生成汇编（在本实验中用处不大）
$ clang -S main.c -o main.s
# 生成目标文件（在本实验中用处不大）
$ clang -c main.c -o main.o
```

clang -S -emit-llvm main.c 可以生成.ll文件

```
# 1. 生成 main.c 对应的 .ll 格式的文件
$ clang -S  -emit-llvm main.c -o main.ll -O0

# 2. 用 lli 解释执行生成的 .ll 文件
$ lli main.ll

$clang main.ll -o code
```

12.3

学llvm

我们的ir 可以用lli来查看结果

12.4

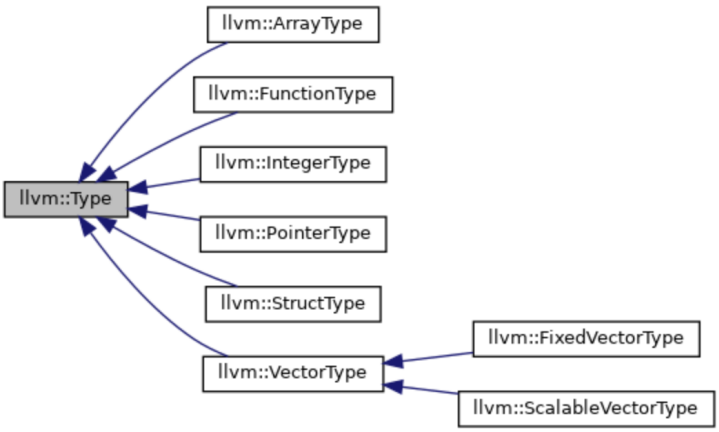
还在学 其实可以开始写了时间紧迫

12.5

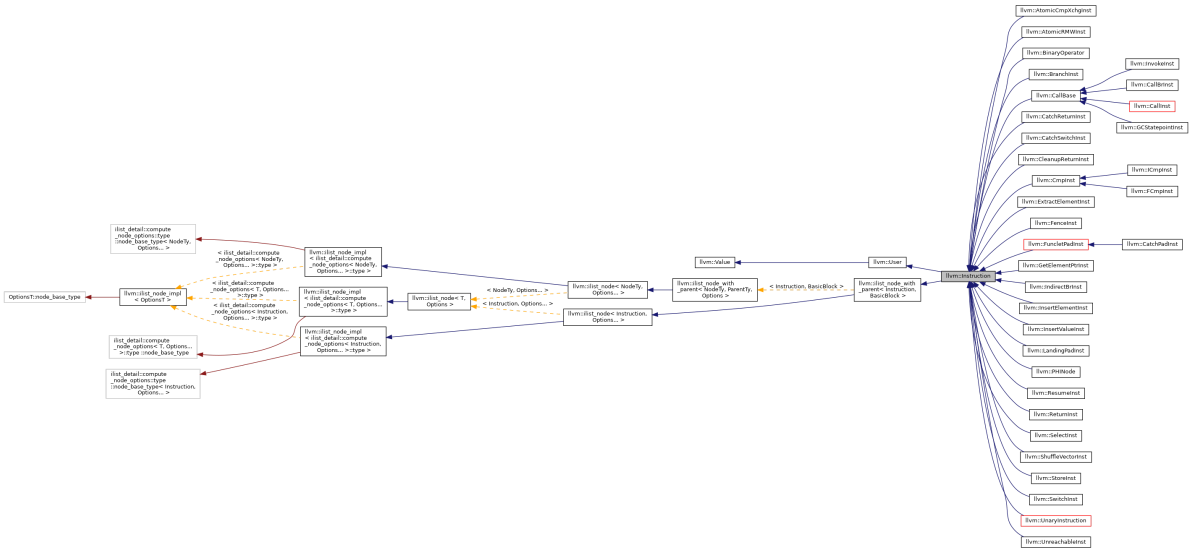
开始写ir

bool 存储与load的时候长度变化

之后可以加上use链用于优化...?也可以现在加



[legend]



12.6

```
const Use &getOperandUse(unsigned i) const {
  assert(i < NumUserOperands && "getOperandUse() out of range!");
  return getOperandList()[i];
}
Use &getOperandUse(unsigned i) {
  assert(i < NumUserOperands && "getOperandUse() out of range!");
  return getOperandList()[i];
}
```

赵一龙说要allign 4

经过测试struct 可以后面直接跟i32 i8 不用像clang那样子采用复杂的形式

//todo加上use value链

12.7开始设计bb function module

12.8

那些instruction里面用到的操作数 暂时不用处理 因为这些都已经存储在具体节点内部 br也不用存储bb 也存储在内部节点里面了

begin to write irbuilder

bool 采取mallocate 1的写法算了

争取这周内实现能跑的llvm!!!

所有的expr里面都加上operand 然后lvalueoperand可以后面加

一个函数的entrance 块进来如果有return值 我都需要先给return allocate i32

所有的returnblock都只有两行 load 和 ret

开始实现它吧

然后弄完之后去把print 搞出来

先不用管那么多细节 大致实现自己清楚就好

先写着后面更改起来也是挺容易的

每一个expr结束都需要把自己的operand设置为一个register

return 块在构造的时候先出现 但是在所有函数的stat都访问完之后才放入linkedlist 这样能够保持linkedlist的顺序性

//todo

emmm

1.加上内建函数 （部分内建函数还没有实现）emmm两边都存储一下好了emmm这样子可以保证所有外部的声明都打印在一起2.研究irprinter的写法**（特别是计数器问题）** 尝试今天能够输出

- [ ] 思考指针的层级问题是如何处理的

编译器设计（书籍）

先实现部分的内建函数（print 相关）

计数问题的解决方法是在currentfunction里面维护一个symboltable()如果存在的话就改变名字（后面数字增加）不对这个是维护bb的名字问题

```
olTable 〉 put                                                    Main
IRBuilder.java   SymbolTable.java   Function.java   FuncCallExprNode.java   BasicBlock.java
            arrayList = symbolTable.get(name);   name: "call"   symbolTable:   size
            label = arrayList.size();
        } else {
            arrayList = new ArrayList<>();
            symbolTable.put(name, arrayList);
            label = 0;
        }

        if (object instanceof BasicBlock = false )
            ((BasicBlock) object).setName(name + "." + label);
        else if (object instanceof Parameter = false )
            ((Parameter) object).setName(name + "." + label);
        else if (object instanceof Register = true )
            ((Register) object).setName(name + "." + label);
        else
```

重新命名的方法

12.9

- [x] 处理重命名问题...? （用function里面的一个register map 一个bb map就可以解决）同时处理parament 作为register的情况 emmm在构造函数的时候可以解决这个问题（register 加入function里面的reg map里面）//todo emmm或者用cnt解决

使用renaming map处理 需要在new reg 和new bb的时候手动调用

- [x] 处理这个东西...?思考

- `%call.0 = call i32 @nmd(i32 1)` 归于call

- [x] 完成println(1) （先完成这个！）
- [x] 完成binary unary operation 提高代码的复用率
- [x] 加上global 以及init块 一个大的问题就是 valdecl init函数

- [x] 完成 print（"hello world"）这里要处理get_elementptr 和 string　//todo 简单的内部string emmm这边采用简单的写法 直接往print里面传 getelementptr ? 问xt emmm 我的处理时 存一个string globalvar map

  %stringLiteral.0 = getelementptr [12 x i8], [12 x i8]* @.str.0, i32 0, i32 0 后面的i32 0是写死的 emm 第一个i32 0代表是地址的偏移量 在string里面一直一直也是i32 0 一个type 一个type * 然后后面加上偏移量

- [x] 思考指针的层级问题是如何处理的

```
; Function Attrs: noinline nounwind optnone uwtable
define dso_local i32 @_Z5tets1i(i32 %0) #0 {
  %2 = alloca i32, align 4
  store i32 %0, i32* %2, align 4
  %3 = load i32, i32* %2, align 4
  ret i32 %3
}
```

不同function的寄存器可以使用同一个名字 所以我需要在function里面维护一个使用过register的名字类似的还有 bb

争取这周能开始跑过大部分点（甚至过掉llvm ）

本质就是ast 到ir 线性结构的转化 所以有一大堆的转化函数 本质就是线性变成非线性

12.10

函数进来时候如果有数值的定义emmm 然后在entrance块的头部进行空间的分配 init 操作都在当前块

- [x] scope 一个id 比如x=10 要先去找给id 分配的空间 emmm 可以在astnode里面记录信息 lrh是varentity emmm也可以在scope里面记录信息 string到 reg的映射(emmm function里面记录scope emm还是其他方法 自己新建立维护一个scope) emm临时变量不需要记录在regmap里面 **只有valdecl需要考虑**....? emmm临时值也没有地址 故不需要考虑 地址到寄存器的映射 load store 的寄存器存储的均是地址
- [x] 加上前面块的注释

之后可以用 思维导图画一幅 结构图java里面的继承关系

emmm 为了输出而加功能 可以分块测试

12.11

关于get element ptr 的理解

1.

```
long *nums = {1, 2, 3};
long index_first(void) {
    return nums[0];
}
```

```
@nums = dso_local global i64* inttoptr (i64 1 to i64*), align 8

define dso_local i64 @index_first() #0{
  %0 = load i64*, i64** @nums, align 8
  %arrayidx = getelementptr inbounds i64, i64* %0, i64 0
  %1 = load i64, i64* %arrayidx, align 8
  ret i64 %1
}
```

2.

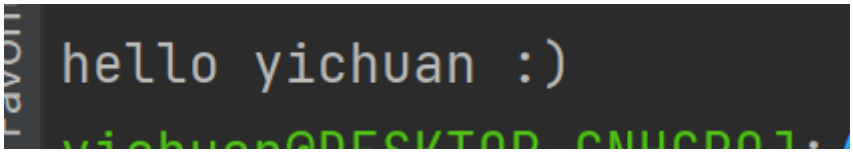
```
long nums[3][3] = { {1, 2, 3}, {2, 3, 4}, {3, 4, 5} };
long i;
long index_i2(void) {
  return nums[i][i];
}
```

```
@nums = dso_local local_unnamed_addr global [3 x [3 x i64]] [[3 x i64] [i64 1,
i64 2, i64 3], [3 x i64] [i64 2, i64 3, i64 4], [3 x i64] [i64 3, i64 4, i64 5]],
align 16
@i = common dso_local local_unnamed_addr global i64 0, align 8

define dso_local i64 @index_i2() local_unnamed_addr #0 {
  %0 = load i64, i64* @i, align 8
  %arrayidx1 = getelementptr inbounds [3 x [3 x i64]], [3 x [3 x i64]]* @nums,
i64 0, i64 %0, i64 %0
  %1 = load i64, i64* %arrayidx1, align 8
  ret i64 %1
}
```

注意到depoint 总是解引用一层 emmm后面描述的树不同维数的偏移量 emmm

new 作为一个loop出现




```
int main() {
int i=0;
{
int i=9;
printlnInt(i);
}
printlnInt(i);
    return 0;
}
```

- [x] finish the above

```
#include "stdio.h"
int k=9;
int p=k;
int main() {
int i=1;    printf("%d",i);
    return 0;
}
```

c库的头文件

- [x] make the following init

```
@k = dso_local global i32 4, align 4
@p = dso_local global i32 0, align 4
@.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
@llvm.global_ctors = appending global [1 x { i32, void ()*, i8* }] [{ i32, void
()*, i8* } { i32 65535, void ()* @_GLOBAL__sub_I_main.mx, i8* null }]


; Function Attrs: noinline norecurse optnone uwtable
define  i32 @main() #1 {
  %1 = alloca i32, align 4
  store i32 0, i32* %1, align 4
  %2 = load i32, i32* @p, align 4
  %3 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8], [3 x
i8]* @.str, i64 0, i64 0), i32 %2)
  ret i32 0
}

declare dso_local i32 @printf(i8*, ...) #2

define  void @_GLOBAL__sub_I_main.mx()  {
  %1 = load i32, i32* @k, align 4
  store i32 %1, i32* @p, align 4
  ret void
}
```

```
@k = dso_local global i32 9, align 4
@p = dso_local global i32 0, align 4
@.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
@llvm.global_ctors = appending global [1 x { i32, void ()*, i8* }] [{ i32, void ()*, i8* } { i32 65535, void ()* @_GLOBAL__sub_I_main.mx, i8* null }]

; Function Attrs: noinline norecurse optnone uwtable
define dso_local i32 @main() #1 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  store i32 0, i32* %1, align 4
  store i32 1, i32* %2, align 4
  %3 = load i32, i32* %2, align 4
  %4 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8], [3 x i8]* @.str, i64 0, i64 0), i32 %3)
  ret i32 0
}

declare dso_local i32 @printf(i8*, ...) #2

; Function Attrs: noinline uwtable
define  void @_GLOBAL__sub_I_main.mx() {
  %1 = load i32, i32* @k, align 4
  store i32 %1, i32* @p, align 4
  ret void
}
```

```
int k=9;
int p=k;
int main() {
printlnInt(p);
   return 0;
}
```

缓存不要太大 一步一步走

还是采用main函数call init 的方法吧 emmm这样子可以 便于后续的codegen而尽量减少clang的调用

我不能做到return void很多函数只有一个Block那样 我没有实现它的水平qwqq

我只能跳转到return快

bingo

总结todo

- [ ] 控制流 for 循环 if 等变化
- [x] 函数调用
- [ ] 处理复杂return情况
- [x] 进入函数的时候要求先给parament allocate空间 然后进行操作 给参数分配空间 结论是需要分配
- [x] 测试一下bool 增加拓展性...?maybe
- [ ] 数组（数组长度存在-1位）
- [x] 单目运算符

12.12

```
void calltest(int i){
printlnInt(i);
}
int main() {
calltest(7);
    return 0;
}
```

测试点

用main.c clang只能编译.c文件

```
define dso_local i32 @k(i32 %0) #0 {
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  store i32 %0, i32* %3, align 4
  %4 = load i32, i32* %3, align 4
  %5 = call i32 (i8*, ...) @printf(i8* getelementptr inbounds ([3 x i8], [3 x
i8]* @.str, i64 0, i64 0), i32 %4)
  %6 = load i32, i32* %2, align 4
  ret i32 %6
}
```

%3是存储内存地址的一个寄存器 %0是一个具体的数值 先把para记录下来 记录string+"para"和寄存器的映射 然后string 和一个新的寄存器建立映射

注意我的para 寄存器名字都加xxx_para

1213

then 肯定存在 else 不一定存在

todo 我的现在这个不行qwqq

```
int foo(int k){
printlnInt(k);}
int main() {
int c=9;
c=c+8;
int b=7;
printlnInt(b);
printlnInt(c);

    return 0;
}
```

处理 重复赋值 会出问题

完成赋值和一元表达式

id出问题 emmm idexpr

关于右移计算 mx采用的是算数右移 由于mx用的都是有符号数

12.14

- [x] return 现在的问题是有两个branch指令 解决惹
- [x] for while
- [ ] array
- [x] 短路求值

短路求值采用 创建新的block的方法 不用按位or或者按位与 直接用icmp实现

condition

if(a==1 && b>0)

else

current bb

condition

false

a==1

RHS

条件跳

强制br

iroperand

short cir

end

iroperand

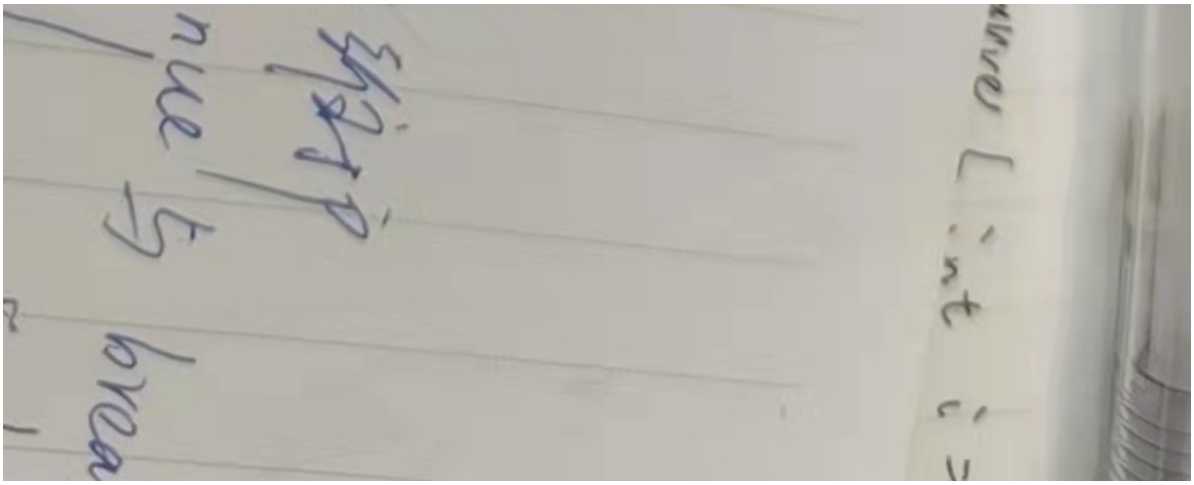
true

强制跳

找来

conti

指向



在irbuilder阶段始终需要访问irlhs但事实上运行中不一定会跳转到branch的块

store bool pointer--->i1

emmm 先在stack中alloca 在两个块都可以store这边的值

- [x] continue break（用一个stack来维护 就挺简单的emmm但不太好想）

```
/*
Test Package: Codegen
Author: Admin
Time: 2020-02-02
Input:
=== input ===
=== end ===
Output:
=== output ===
=== end ===
ExitCode: 10
InstLimit: -1
Origin Package: Codegen Pretest-577
*/


int main() {
int i;
    for(i=0;i<6;i++){printlnInt(i);
    if(i>4)break;
    }
    return 0;
}
```

//to pass it

- [x] string 加法 大小比较 先写 可以跑点

```
12 字符串:
12.1 字符串对象
字符串对象赋值为null是语法错误。

12.2 字符串的双目运算
+表示字符串拼接
```

```
==, !=比较两个字符串是否完全一致（不是内存地址）

<, >, <=, >=用于比较字典序大小

剩余双目运算符都是语法错误，字符串双目运算符要求两边类型相同，不满足则语法错误。

12.3 字符串的内建方法
函数：int length();

使用：<StringIdentifier>.length();

作用：返回字符串的长度。

函数：string substring(int left, int right);

使用：<StringIdentifier>.substring(left, right);

作用：返回下标为[left, right)的子串。

函数：int parseInt();

使用：<StringIdentifier>.parseInt();

作用：返回一个整数，这个整数应该是该字符串的最长前缀。如果该字符串没有一个前缀是整数，结果未定义。如果该整数超界，结果也未定义。

函数：int ord(int pos);

使用：<StringIdentifier>.ord(pos);

作用：返回字符串中的第pos位上的字符的ASCII码。下标从0开始编号。

常量字符串不具有内建方法，使用内建方法的常量字符串未定义。
```

由于semantic写的过于屎山 我的type判断不能使用instance of 而只能用type.typename来判断string
注意这边可能导致typetrans 产生问题 可以用typeneme特判一下

- [x] todo the next point

```
int A = 1;
int B = 1;
int C = 1;

int main(){
    //while (C < (1 << 29) && C > -(1 << 29)){
    //  A = ((((((((C - A + B) - (A + B)) + ((C - A + B) - (A + B))) + (((C - A
+ B) - (A + B)) + (C - A + B))) - ((((A + B) + (C - A + B)) - (A + B)) + (((C - A
+ B) - (A + B)) + (C - A + B)))) - (((((A + B) + (C - A + B)) - ((A + B) + (C - A
+ B))) - (((A + B) + (C - A + B)) - (A + B))) + ((((C - A + B) - (A + B)) + (C -
A + B)) - (((A + B) + (C - A + B)) - (A + B))))) + ((((((C - A + B) - (A + B)) +
((C - A + B) - (A + B))) + (((C - A + B) - (A + B)) + (C - A + B))) - ((((A + B)
+ (C - A + B)) - (A + B)) + (((C - A + B) - (A + B)) + (C - A + B)))) - (((((A +
B) + (C - A + B)) - (A + B)) + (((C - A + B) - (A + B)) + (C - A + B))) - ((((A +
B) + (C - A + B)) - (A + B)) + (((C - A + B) - (A + B)) + (C - A + B))))))) -
(((((((A + B) + (C - A + B)) - ((A + B) + (C - A + B))) - (((A + B) + (C - A +
B)) - (A + B))) + ((((C - A + B) - (A + B)) + (C - A + B)) - (((A + B) + (C - A +
B)) - (A + B)))) + (((((C - A + B) - (A + B)) + (C - A + B)) - (((A + B) + (C - A
+ B)) - (A + B))) + ((((C - A + B) - (A + B)) + (C - A + B)) - (((A + B) + (C - A
+ B)) - (A + B))))) + ((((((C - A + B) - (A + B)) + ((C - A + B) - (A + B))) +
(((C - A + B) - (A + B)) + (C - A + B))) - ((((A + B) + (C - A + B)) - (A + B)) +
(((C - A + B) - (A + B)) + (C - A + B)))) - (((((A + B) + (C - A + B)) - (A + B))
+ (((C - A + B) - (A + B)) + (C - A + B))) - ((((A + B) + (C - A + B)) - (A + B))
+ (((C - A + B) - (A + B)) + (C - A + B)))))));
    //  B = ((((((((C - A + B) - (A + B)) + ((C - A + B) - (A + B))) + (((C - A
+ B) - (A + B)) + (C - A + B))) - ((((A + B) + (C - A + B)) - (A + B)) + (((C - A
+ B) - (A + B)) + (C - A + B)))) - (((((A + B) + (C - A + B)) - ((A + B) + (C - A
+ B))) - (((A + B) + (C - A + B)) - (A + B))) + ((((C - A + B) - (A + B)) + (C -
A + B)) - (((A + B) + (C - A + B)) - (A + B))))) + ((((((C - A + B) - (A + B)) +
((C - A + B) - (A + B))) + (((C - A + B) - (A + B)) + (C - A + B))) - ((((A + B)
+ (C - A + B)) - (A + B)) + (((C - A + B) - (A + B)) + (C - A + B)))) - (((((A +
B) + (C - A + B)) - (A + B)) + (((C - A + B) - (A + B)) + (C - A + B))) - ((((A +
B) + (C - A + B)) - (A + B)) + (((C - A + B) - (A + B)) + (C - A + B))))))) -
(((((((A + B) + (C - A + B)) - ((A + B) + (C - A + B))) - (((A + B) + (C - A +
B)) - (A + B))) + ((((C - A + B) - (A + B)) + (C - A + B)) - (((A + B) + (C - A +
B)) - (A + B)))) + (((((C - A + B) - (A + B)) + (C - A + B)) - (((A + B) + (C - A
+ B)) - (A + B))) + ((((C - A + B) - (A + B)) + (C - A + B)) - (((A + B) + (C - A
+ B)) - (A + B))))) + ((((((C - A + B) - (A + B)) + ((C - A + B) - (A + B))) +
(((C - A + B) - (A + B)) + (C - A + B))) - ((((A + B) + (C - A + B)) - (A + B)) +
(((C - A + B) - (A + B)) + (C - A + B)))) - (((((A + B) + (C - A + B)) - (A + B))
+ (((C - A + B) - (A + B)) + (C - A + B))) - ((((A + B) + (C - A + B)) - (A + B))
+ (((C - A + B) - (A + B)) + (C - A + B)))))));
```

```
    //  C = ((((((((C - A + B) - (A + B)) + ((C - A + B) - (A + B))) + (((C - A
+ B) - (A + B)) + (C - A + B))) - ((((A + B) + (C - A + B)) - (A + B)) + (((C - A
+ B) - (A + B)) + (C - A + B)))) - (((((A + B) + (C - A + B)) - ((A + B) + (C - A
+ B))) - (((A + B) + (C - A + B)) - (A + B))) + ((((C - A + B) - (A + B)) + (C -
A + B)) - (((A + B) + (C - A + B)) - (A + B))))) + ((((((C - A + B) - (A + B)) +
((C - A + B) - (A + B))) + (((C - A + B) - (A + B)) + (C - A + B))) - ((((A + B)
+ (C - A + B)) - (A + B)) + (((C - A + B) - (A + B)) + (C - A + B)))) - (((((A +
B) + (C - A + B)) - (A + B)) + (((C - A + B) - (A + B)) + (C - A + B))) - ((((A +
B) + (C - A + B)) - (A + B)) + (((C - A + B) - (A + B)) + (C - A + B))))))) -
(((((((A + B) + (C - A + B)) - ((A + B) + (C - A + B))) - (((A + B) + (C - A +
B)) - (A + B))) + ((((C - A + B) - (A + B)) + (C - A + B)) - (((A + B) + (C - A +
B)) - (A + B)))) + (((((C - A + B) - (A + B)) + (C - A + B)) - (((A + B) + (C - A
+ B)) - (A + B))) + ((((C - A + B) - (A + B)) + (C - A + B)) - (((A + B) + (C - A
+ B)) - (A + B))))) + ((((((C - A + B) - (A + B)) + ((C - A + B) - (A + B))) +
(((C - A + B) - (A + B)) + (C - A + B))) - ((((A + B) + (C - A + B)) - (A + B)) +
(((C - A + B) - (A + B)) + (C - A + B)))) - (((((A + B) + (C - A + B)) - (A + B))
+ (((C - A + B) - (A + B)) + (C - A + B))) - ((((A + B) + (C - A + B)) - (A + B))
+ (((C - A + B) - (A + B)) + (C - A + B)))))));
    //}
    println(toString(A) + " " + toString(B) + " " + toString(C));
    return 0;
}
```

char * 和string 都是i8*

- [x] string =//感觉是不会用到的?
- [x] 加上一个print()的 built in 貌似漏了一个

12.15

- [x] 拓展一下for
- [ ] 开始array 弄完array（开始！！！！）
- [ ] class

```
int main() {
    int []a=new int[5];
    int i=0;
    for(i=0;i<5;i++)a[i]=i;
    for(i=0;i<5;i++)printlnInt(a[i]);

    return 0;
}
```

to pass it

```
/*
Test Package: Codegen
Author: 11' Hang Wu
Time: 2020-01-25
Input:
=== input ===
=== end ===
Output:
=== output ===
```

```
2
=== end ===
ExitCode: 0
InstLimit: -1
Origin Package: Codegen Pretest-538
*/

//int[] a = new int[4];
int main()
{
    int[][] b = new int[4][2];
    b[2][1]=2;
    //a=b;
    //println(toString(a[2]));
    println(toString(b[2][1]));

    return 0;
}
```

```
/*
Test Package: Codegen
Author: 11' Hang Wu
Time: 2020-01-25
Input:
=== input ===
=== end ===
Output:
=== output ===
2
=== end ===
ExitCode: 0
InstLimit: -1
Origin Package: Codegen Pretest-538
*/

//int[] a = new int[4];
int main()
{
    int[] b = new int[4];
    b[2]=2;
    //a=b;
    //println(toString(a[2]));
    println(toString(b[2]));

    return 0;
}
```

- [x] to work it
- [x] array 作为bianry expr的左值的时候

采用特判的办法 在arrayexp记录额外的数据

我的getelement ptr是每次都解引用一层的emmm 所以后面的参数列表都只有一个 采用多行多次的方法 而不是一步到位

12 16完成多维数组和class

- [x] todo

```
/*
Test Package: Codegen
Author: 11' Hang Wu
Time: 2020-01-25
Input:
=== input ===
=== end ===
Output:
=== output ===
2
=== end ===
ExitCode: 0
InstLimit: -1
Origin Package: Codegen Pretest-538
*/

//int[] a = new int[4];
int main()
{
    int[][] b = new int[4][2];
    b[2][1]=2;
    //a=b;
    //println(toString(a[2]));
    println(toString(b[2][1]));

    return 0;
}
```

- [ ] class

12.17

```
/*
Test Package: Codegen
Author: 10' Huan Yang
Time: 2020-01-24
Input:
=== input ===
102
=== end ===
Output:
=== output ===
68
=== end ===
ExitCode: 0
InstLimit: -1
Origin Package: Codegen Pretest-523
*/

bool check(int a, int N) {
```

```
    return ((a < N) && (a >= 0));
}

int main() {
    int N;
    int head;
    int startx;
    int starty;
    int targetx;
    int targety;
    int tail;
    int ok;
    int now;
    int x;
    int y;
    int[] xlist;
    int[] ylist;
    int[][] step;
    int i;
    int j;

    N = getInt();
    head =0;
    tail = 0;
    startx = 0;
    starty = 0;
    targetx = N-1;
    targety = N-1;
    x = 0;
    y = 0;
    now = 0;
    ok = 0;
    xlist = new int[N * N];
    for (i = 0; i < N * N; i ++ )
        xlist[i] = 0;
    ylist = new int[N * N];
    for (i = 0; i < N * N; i ++ )
        ylist[i] = 0;
    step = new int[N][];
    for (i = 0; i < N;  i ++ ) {
        step[i] = new int[N];
        for (j = 0; j < N; j ++ )
        step[i][j] = -1;
    }
    xlist[0] = startx;
    ylist[0] = starty;
    step[startx][starty] = 0;
    while (head <= tail)
    {
        now = step[xlist[head]][ylist[head]];
        x = xlist[head] - 1;
        y = ylist[head] - 2;
        if (check(x, N) && check(y, N) && step[x][y] == -1)
        {
            tail = tail + 1;
            xlist[tail] = x;
            ylist[tail] = y;
            step[x][y] = now + 1;
```

```
            if (x == targetx && y == targety) ok = 1;
        }
        x = xlist[head] - 1;
        y = ylist[head] + 2;
        if (check(x, N) && check(y, N) && step[x][y] == -1)
        {
            tail = tail + 1;
            xlist[tail] = x;
            ylist[tail] = y;
            step[x][y] = now + 1;
            if (x == targetx && y == targety) ok = 1;
        }
        x = xlist[head] + 1;
        y = ylist[head] - 2;
        if (check(x, N) && check(y, N) && step[x][y] == -1)
        {
            tail = tail + 1;
            xlist[tail] = x;
            ylist[tail] = y;
            step[x][y] = now + 1;
            if (x == targetx && y == targety) ok = 1;
        }
        x = xlist[head] + 1;
        y = ylist[head] + 2;
        if (check(x, N) && check(y, N) && step[x][y] == -1)
        {
            tail = tail + 1;
            xlist[tail] = x;
            ylist[tail] = y;
            step[x][y] = now + 1;
            if (x == targetx && y == targety) ok = 1;
        }
        x = xlist[head] - 2;
        y = ylist[head] - 1;
        if (check(x, N) && check(y, N) && step[x][y] == -1)
        {
            tail = tail + 1;
            xlist[tail] = x;
            ylist[tail] = y;
            step[x][y] = now + 1;
            if (x == targetx && y == targety) ok = 1;
        }
        x = xlist[head] - 2;
        y = ylist[head] + 1;
        if (check(x, N) && check(y, N) && step[x][y] == -1)
        {
            tail = tail + 1;
            xlist[tail] = x;
            ylist[tail] = y;
            step[x][y] = now + 1;
            if (x == targetx && y == targety) ok = 1;
        }
        x = xlist[head] + 2;
        y = ylist[head] - 1;
        if (check(x, N) && check(y, N) && step[x][y] == -1)
        {
            tail = tail + 1;
            xlist[tail] = x;
```

```
            ylist[tail] = y;
            step[x][y] = now + 1;
            if (x == targetx && y == targety) ok = 1;
        }
        x = xlist[head] + 2;
        y = ylist[head] + 1;
        if (check(x, N) && check(y, N) && step[x][y] == -1)
        {
            tail = tail + 1;
            xlist[tail] = x;
            ylist[tail] = y;
            step[x][y] = now + 1;
            if (x == targetx && y == targety) ok = 1;
        }
        if (ok == 1) break;
        head = head + 1;
    }
    if (ok == 1) println(toString(step[targetx][targety]));
    else print("no solution!\n");
    return 0;
}
```

```
/*
Test Package: Sema_Local_Preview
Test Target: String
Author: 15' Caomeng Yao
Time: 2019-10-20
Verdict: Success
Origin Package: Semantic Extended
*/

string[] str_arr = null;

int main() {
    int la = getInt();
    str_arr = new string[la];

    int i;
    int cnt = 0;
    for (i = 0; i < la; i++) {
        str_arr[i] = getString();
        cnt = cnt + str_arr[i].length();
    }

    string str = "";
    int sum = 0;
    for (i = 0; i < la; ++i){
        str = str + str_arr[i].substring(0, str_arr[i].length() - 1);
        sum = sum + str_arr[i].ord(0);
    }
    println(str);
    print(toString(sum));
    if (cnt == str.length()) return 0;

    else return 1;
}
```

- [ ] t1 t3 短路求值烂了qwqqq
- [ ] class!!

12.18

- [ ] class完成

在getelementptr上面有一个写死的1 第一位

```
class m{
int k;
int m;
};
int main(){
m M=new m;
M.k=9;
printlnInt(M.k);
return 0;
}

/*
Test Package: Codegen
Author: Admin
Time: 2020-02-02
Input:
=== input ===
=== end ===
Output:
=== output ===
=== end ===
ExitCode: 70
InstLimit: -1
Origin Package: Codegen Pretest-574
*/
class C2 {
    int x;
    int y;
    int z;
};

int main() {
    C2 obj = new C2;
    obj.x = 10;
    obj.y = 20;
    obj.z = 40;
    return obj.x + obj.y + obj.z;
}




class m{
int k;
int m;
MMM o;
m(){
o=new MMM;
```

```
}
};
class MMM{
int g=8;
MMM(){
g=6;
}

};
int main(){
m M=new m;
M.k=9;
printlnInt(M.k);
printlnInt(M.o.g);

return 0;
}
```

如果是struct type则需要转化为指针类型

llvm是强类型语言 alloca出来指针还是数据都是32位的emmm class*也是这样的

- [ ] 加上align qwqqqq
- [ ] 数组的size
- [ ] wsl java环境配置
- [ ] return在前 有些细节还没处理

12.19

class llvm 的gep 位置是该元素在class中的第几个元素

```
%class.C2 = type { i32, i32, i32 }  这个应该理解成一个type而不是一个寄存器
```

- [ ]
-
  ```
  /*
  Test Package: Codegen
  Author: Admin
  Time: 2020-02-02
  Input:
  === input ===
  === end ===
  Output:
  === output ===
  === end ===
  ExitCode: 2
  InstLimit: -1
  Origin Package: Codegen Pretest-587-Modifiy
  */
  int main() {
      string s = "hahaha";
      return s.substring(2, 4).length();
  }
  ```

- [ ] class 内函数 加上this指针的传入 再加上functioncall 注意function里面的参数全部存储在fuctiontype里面 自己的设计

```
class K{

int m;
void test(){
printlnInt(m);
}
};
int main(){
K tmp=new K;
tmp.m=9;
tmp.test();
return 0;

}
```

```
    public void visit(IdExp_ASTnode it) {
        //naive type but it work now don't find bug
        BaseOperand id_reg = current_ir_scope.find_id_to_reg(it.index);
        //single val decl before
        if (id_reg != null) {
            Register load_reg = new Register(type_trans.asttype_to_irtype(it.type), it.index);
            current_function.renaming_add(load_reg);
            current_basicblock.instruction_add(new LoadInstruction(current_basicblock, load_reg, id_reg));
            it.ir_operand = load_reg;
        } else {
            //in class where don't have value decl before so we load the data from the heap using gep
//todo
        }
    }
```

我的设计是刚开始class里面最先遍历到的id gep一下 之后都可以用这个公用的值 目前看来没问题

- [ ] class内部函数调用 emmm this指针使用

```
/*
Test Package: Codegen
Author: 14' Rongyu You
Time: 2020-02-03
Input:
=== input ===
=== end ===
Output:
=== output ===
vector x: ( 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 )
excited!
vector y: ( 9, 8, 7, 817, 5, 4, 3, 2, 1, 0 )
x + y: ( 18, 16, 14, 823, 10, 8, 6, 4, 2, 0 )
x * y: 0
(1 << 3) * y: ( 72, 64, 56, 6536, 40, 32, 24, 16, 8, 0 )
=== end ===
ExitCode: 0
InstLimit: -1
Origin Package: Codegen Pretest-900
*/
//
// Naive vector class for Mx*.
// Without any guarantee for robustness.
```

```
//
class vector{
    int[] data;
    void init(int[] vec){
        // init the vector from an array
        if (vec == null) return;
        data = new int[vec.size()];
        int i;
        for (i = 0; i < vec.size(); ++i)
        {
            data[i] = vec[i];
        }
    }

    int getDim(){
        if (data == null) return 0;
        return data.size();
    }

    int dot(vector rhs){
        int i = 0;
        int result = 0;
        while(i < getDim()){
            //result = data[i] * rhs[i];
            result = data[i] * rhs.data[i];
            ++i;
        }
        return result;
    }

    vector scalarInPlaceMultiply(int c){
        if (data == null) return null;
        int i;
        for (i = 0; i < getDim(); ++i) {
            this.data[i] = c * this.data[i];
        }
        return this;
    }

    vector add(vector rhs){
        if (getDim() != rhs.getDim() || getDim() == 0)
            return null;
        vector temp = new vector;
        int i;
        temp.data = new int[getDim()];
        for (i = 0; i < getDim(); ++i){
            temp.data[i] = data[i] + rhs.data[i];
        }
        return temp;
    }

    bool set(int idx, int value){
        if (getDim() < idx) return false;
        data[idx] = value;
        return true;
    }

    string tostring(){
```

```
        string temp = "( ";
        if (getDim() > 0) {
            temp = temp + toString(data[0]);
        }
        int i;
        for (i = 1; i < getDim(); ++i) {
            temp = temp + ", " + toString(data[i]);
        }
        temp = temp + " )";
        return temp;
    }

    bool copy(vector rhs){
        if (rhs == null) return false;
        if (rhs.getDim() == 0) {
            data = null;
        } else {
            data = new int[rhs.getDim()];
            int i;
            for (i = 0; i < getDim(); ++i) {
                data[i] = rhs.data[i];
            }
        }
        return true;
    }
};

int main(){
    vector x = new vector;
    int[] a = new int[10];
    int i;
    for (i = 0; i < 10; ++i){
        a[i] = 9 - i;
    }
    x.init(a);
    print("vector x: ");
    println(x.tostring());

    vector y = new vector;
    y.copy(x);
    if (y.set(3, 817)){
        println("excited!");
    }
    print("vector y: ");
    println(y.tostring());
    print("x + y: ");
    println((x.add(y)).tostring());
    print("x * y: ");
    println(toString(x.dot(y)));
    print("(1 << 3) * y: ");
    println(y.scalarInPlaceMultiply(1 << 3).tostring());
    return 0;
}
```

- [ ] return this && string equal

我使用 module_in_irbuilder.Module_Struct_Map.get(current_class_detail.classname) 来找到当前class的类型

- [ ] to pass constructer

```
/*
Test Package: Codegen
Author: 14' Xingyuan Sun
Time: 2020-02-03
Input:
=== input ===
=== end ===
Output:
=== output ===
(0, 0, 0)
28716325
7421636
9980404
38464544
1854392
(7616, 1666188, -1232986)
(-508, 4119, 3390)
(562, 1584, 2144)
(-920, 768, -524)
(612, -469, -630)
=== end ===
ExitCode: 0
InstLimit: -1
Origin Package: Codegen Pretest-901
*/
class point {
    int x;
    int y;
    int z;
    point() {
        x = 0;
        y = 0;
        z = 0;
    }
    void set(int a_x, int a_y, int a_z){
        x = a_x;
        y = a_y;
        z = a_z;
    }
    int sqrLen(){
        return x * x + y * y + z * z;
    }
    int sqrDis(point other) {
        return (x - other.x) * (x - other.x) + (y - other.y) * (y - other.y) +
(z - other.z) * (z - other.z);
    }
    int dot(point other) {
        return x * other.x + y * other.y + z * other.z;
    }
    point cross(point other) {
        point retval = new point;
```

```
        retval.set(y * other.z - z * other.y, z * other.x - x * other.z, x *
other.y - y * other.x);
        return retval;
    }
    point add(point other) {
        x = x + other.x;
        y = y + other.y;
        z = z + other.z;
        return this;
    }
    point sub(point other) {
        x = x - other.x;
        y = y - other.y;
        z = z - other.z;
        return this;
    }
    void printPoint() {
        println("(" + toString(x) + ", " + toString(y) + ", " + toString(z) +
")");
    }
};

int main() {
    point a = new point;
    point b = new point;
    point c = new point;
    point d = new point;
    a.printPoint();
    a.set(849, -463, 480);
    b.set(-208, 585, -150);
    c.set(360, -670, -742);
    d.set(-29, -591, -960);
    a.add(b);
    b.add(c);
    d.add(c);
    c.sub(a);
    b.sub(d);
    d.sub(c);
    c.add(b);
    a.add(b);
    b.add(b);
    c.add(c);
    a.sub(d);
    a.add(b);
    b.sub(c);
    println(toString(a.sqrLen()));
    println(toString(b.sqrLen()));
    println(toString(b.sqrDis(c)));
    println(toString(d.sqrDis(a)));
    println(toString(c.dot(a)));
    b.cross(d).printPoint();
    a.printPoint();
    b.printPoint();
    c.printPoint();
    d.printPoint();
    return 0;
}
```

有构造函数的话就需要在new A 的时候显示调用构造函数

带参数的构造函数是未定义行为

//todo add constructer

```
/*
Test Package: Codegen
Author: 14' Xingyuan Sun
Time: 2020-02-03
Input:
=== input ===
=== end ===
Output:
=== output ===
(0, 0, 0)
28716325
7421636
9980404
38464544
1854392
(7616, 1666188, -1232986)
(-508, 4119, 3390)
(562, 1584, 2144)
(-920, 768, -524)
(612, -469, -630)
=== end ===
ExitCode: 0
InstLimit: -1
Origin Package: Codegen Pretest-901
*/
class point {
    int x;
    int y;
    int z;
    point() {
        x = 1;
        y = 0;
        z = 0;
    }
    void printPoint() {
        println("(" + toString(x) + ", " + toString(y) + ", " + toString(z) +
")");
    }
};

int main() {
    point a = new point;
    a.printPoint();

    return 0;
}
```

//todo

```
step = new int[N][];
    for (i = 0; i < N;  i ++ ) {
        step[i] = new int[N];
        for (j = 0; j < N; j ++ )
        step[i][j] = -1;
    }

    for (i = 0; i < N;  i ++ ) {
                for (j = 0; j < N; j ++ )
                printlnInt(step[i][j]);
    }
```

fuckkkkk! break at this pointer!!

```
/*
Test Package: Optim
Author: Zhekai Zhang, 15
Input:
=== input ===
1 acm2015
2 ABC64A57029F21F165A96BDB59F0351C7C7D1769
0

=== end ===
Output:
=== output ===
5B38674EB4BD02CEC1D41C8DE3CC14A9872A2656
ACM

=== end ===
ExitCode: 0
InstLimit: -1
*/

//Compute and Crack SHA-1
//by zzk

int hex2int(string x)
{
    int i;
    int result = 0;
    for(i=0;i<x.length();i++)
    {
        int digit = x.ord(i);
        if(digit >= 48 && digit <= 57)
            result = result * 16 + digit - 48;
        else if(digit >= 65 && digit <= 70)
            result = result * 16 + digit - 65 + 10;
        else if(digit >= 97 && digit <= 102)
            result = result * 16 + digit - 97 + 10;
        else
            return 0;
    }
    return result;
}
```

```
string asciiTable = " !\"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz{|}~";
string int2chr(int x)
{
    if(x >= 32 && x <= 126)
        return asciiTable.substring(x-32, x-31);
    return "";
}
string toStringHex(int x)
{
    string ret = "";
    int i;
    for(i=28;i>=0;i=i-4)
    {
        int digit = (x >> i) & 15;
        if(digit < 10)
            ret = ret + int2chr(48+digit);
        else
            ret = ret + int2chr(65+digit-10);
    }
    return ret;
}
int rotate_left(int x, int shift)
{
    if(shift == 1)
        return ((x & 2147483647) << 1) | ((x >> 31) & 1);
    if(shift == 31)
        return ((x & 1) << 31) | ((x >> 1) & 2147483647);
    return ((x & ((1 << (32-shift)) - 1)) << shift) | ((x >> (32-shift)) & ((1 <<
shift) - 1));
}
int add(int x, int y)   //to avoid possible undefined behaviour when overflow
{
    int low = (x & 65535) + (y & 65535);
    int high = (((x >> 16) & 65535) + ((y >> 16) & 65535) + (low >> 16)) &
65535;
    return (high << 16) | (low & 65535);
}
int lohi(int lo, int hi)
{
    return lo | (hi << 16);
}

int MAXCHUNK = 100;
int MAXLENGTH = (MAXCHUNK-1) * 64 - 16;
int[][] chunks = new int[MAXCHUNK][80];
int[] inputBuffer = new int[MAXLENGTH];
int[] outputBuffer = new int[5];
int[] sha1(int[] input, int length)
{
    int nChunk = (length + 64 - 56) / 64 + 1;
    if(nChunk > MAXCHUNK)
    {
        println("nChunk > MAXCHUNK!");
        return null;
    }
    int i;
    int j;
```

```
    for(i=0;i<nChunk;i++)
        for(j=0;j<80;j++)
            chunks[i][j] = 0;
    for(i=0;i<length;i++)
        chunks[i/64][i%64/4] = chunks[i/64][i%64/4] | (input[i] << ((3-i%4)*8));
    chunks[i/64][i%64/4] = chunks[i/64][i%64/4] | (128 << ((3-i%4)*8));
    chunks[nChunk-1][15] = length << 3;
    chunks[nChunk-1][14] = (length >> 29) & 7;

    int h0 = 1732584193;  //0x67452301
    int h1 = lohi(43913, 61389);  //0xEFCDAB89
    int h2 = lohi(56574, 39098); //0x98BADCFE
    int h3 = 271733878;   //0x10325476
    int h4 = lohi(57840, 50130); //0xC3D2E1F0
    for(i=0;i<nChunk;i++)
    {
        for(j=16;j<80;j++)
            chunks[i][j] = rotate_left(chunks[i][j-3] ^ chunks[i][j-8] ^
chunks[i][j-14] ^ chunks[i][j-16], 1);

        int a = h0;
        int b = h1;
        int c = h2;
        int d = h3;
        int e = h4;
        for(j=0;j<80;j++)
        {
            int f;
            int k;
            if(j<20)
            {
                f = (b & c) | ((~b) & d);
                k = 1518500249; //0x5A827999
            }
            else if(j<40)
            {
                f = b ^ c ^ d;
                k = 1859775393; //0x6ED9EBA1
            }
            else if(j<60)
            {
                f = (b & c) | (b & d) | (c & d);
                k = lohi(48348, 36635); //0x8F1BBCDC
            }
            else
            {
                f = b ^ c ^ d;
                k = lohi(49622, 51810); //0xCA62C1D6
            }
            int temp = add(add(add(rotate_left(a, 5), e), add(f, k)), chunks[i]
[j]);
            e = d;
            d = c;
            c = rotate_left(b, 30);
            b = a;
            a = temp;
        }
        h0 = add(h0, a);
```

```
        h1 = add(h1, b);
        h2 = add(h2, c);
        h3 = add(h3, d);
        h4 = add(h4, e);
    }
    outputBuffer[0] = h0;
    outputBuffer[1] = h1;
    outputBuffer[2] = h2;
    outputBuffer[3] = h3;
    outputBuffer[4] = h4;
    return outputBuffer;
}

void computeSHA1(string input)
{
    int i;
    for(i=0; i<input.length(); i++)
        inputBuffer[i] = input.ord(i);
    int[] result = sha1(inputBuffer, input.length());
    for(i=0; i<result.size(); i++)
        print(toStringHex(result[i]));
    println("");
}

int nextLetter(int now)
{
    if(now == 122) //'z'
        return -1;
    if(now == 90)  //'Z'
        return 97; //'a'
    if(now == 57)  //'9'
        return 65;
    return now + 1;
}

bool nextText(int[] now, int length)
{
    int i;
    for(i=length-1; i>=0; i--)
    {
        now[i] = nextLetter(now[i]);
        if(now[i] == -1)
            now[i] = 48;    //'0'
        else
            return true;
    }
    return false;
}

bool array_equal(int[] a, int[] b)
{
    if(a.size() != b.size())
        return false;
    int i;
    for(i=0; i<a.size(); i++)
        if(a[i] != b[i])
            return false;
    return true;
```

```
}

void crackSHA1(string input)
{
    int[] target = new int[5];
    if(input.length() != 40)
    {
        println("Invalid input");
        return;
    }
    int i;
    for(i=0;i<5;i++)
        target[i] = 0;
    for(i=0;i<40;i=i+4)
        target[i/8] = target[i/8] | (hex2int(input.substring(i, i+4)) << (1 - (i
/ 4) % 2) * 16);

    int MAXDIGIT = 4;
    int digit;
    for(digit=1; digit <= MAXDIGIT; digit++)
    {
        for(i=0;i<digit;i++)
            inputBuffer[i] = 48;
        while(true)
        {
            int[] out = sha1(inputBuffer, digit);
            if(array_equal(out, target))
            {
                for(i=0;i<digit;i++)
                    print(int2chr(inputBuffer[i]));
                println("");
                return;
            }
            if(!nextText(inputBuffer, digit))
                break;
        }
    }
    println("Not Found!");
}

int main()
{
    int op;
    string input;
    while(true)
    {
        op = getInt();
        if(op == 0)
            break;
        if(op == 1)
        {
            input = getString();
            computeSHA1(input);
        }
        else if(op == 2)
        {
            input = getString();
            crackSHA1(input);
```

```
        }
    }
    return 0;
}
```

to do add internal function emmm such as length ord and so on

> **12.3 字符串的内建方法**
>
> 函数：`int length();`
>
> 使用：`<StringIdentifier>.length();`
>
> 作用：返回字符串的长度。
>
> 函数：`string substring(int left, int right);`
>
> 使用：`<StringIdentifier>.substring(left, right);`
>
> 作用：返回下标为`[left, right)`的子串。
>
> 函数：`int parseInt();`
>
> 使用：`<StringIdentifier>.parseInt();`
>
> 作用：返回一个整数，这个整数应该是该字符串的最长前缀。如果该字符串没有一个前缀是整数，结果未定义。如果该整数超界，结果也未定义。
>
> 函数：`int ord(int pos);`
>
> 使用：`<StringIdentifier>.ord(pos);`
>
> 作用：返回字符串中的第pos位上的字符的ASCII码。下标从0开始编号。
>
> 常量字符串不具有内建方法，使用内建方法的常量字符串未定义。

emmm i will specially check it

emmm直接两边都用map find 一下就可以了 emmm 感觉很简单

在functioncall里面更改很简单就可以实现

```
/*
Test Package: Optim
Author: Yunwei Ren, 17
Input:
=== input ===
5

=== end ===
Output:
=== output ===
0: 4
1: 4
2: 6
3: 4
4: 4
6: 2
7: 3
8: 2
10: 1
11: 4
13: 2
14: 6
15: 2
```

```
18: 4
19: 3
20: 1
21: 5
22: 1
24: 3
25: 3
26: 2
27: 2
30: 5
33: 16
35: 2
36: 1
39: 4
40: 1
41: 8
42: 7
43: 2
44: 2
46: 5
47: 1
48: 2
51: 2
55: 5
57: 2
60: 1
63: 2
64: 1
65: 2
66: 2
67: 2
68: 1
69: 1
75: 1
76: 2
77: 2
78: 1
80: 4
81: 5
82: 2
83: 1
84: 2
86: 4
87: 2
89: 5
90: 6
91: 4
92: 6
93: 1
94: 1
97: 5
99: 1
102: 1
105: 1
106: 2
107: 5
108: 2
109: 5
```

```
111: 3
112: 7
115: 2
116: 5
117: 2
118: 1
119: 1
120: 3
121: 3
122: 8
126: 2
127: 1

=== end ===
ExitCode: 0
InstLimit: -1
*/

class Node {
  Node pnt;
  Node[] children;
  int key;
  int duplicate;
};

Node constructNode(int key, Node pnt, Node lchild, Node rchild) {
  Node node = new Node;
  node.children = new Node[2];
  node.key = key;
  node.duplicate = 1;
  node.pnt = pnt;
  node.children[0] = lchild;
  node.children[1] = rchild;
  return node;
}

Node root = null;

int insertImpl(Node cur, Node pnt, int childId, int key) {
  if (cur == null) {
    cur = constructNode(key, pnt, null, null);
    pnt.children[childId] = cur;
    return 0;
  }
  if (cur.key == key) {
    ++cur.duplicate;
    return 1;
  }
  int id = 0;
  if (cur.key < key)
    id = 1;
  return insertImpl(cur.children[id], cur, id, key);
}

// return 1 if isIn
int insert(int key) {
  if (root != null)
    return insertImpl(root, null, 0 - 1, key);
```

```
  root = constructNode(key, null, null, null);
  return 0;
}

Node findLargest(Node cur) {
  if (cur.children[1] == null)
    return cur;
  return findLargest(cur.children[1]);
}

int eraseImpl(Node cur, Node pnt, int childId, int key) {
  if (cur == null)
    return 0;
  if (cur.key > key)
    return eraseImpl(cur.children[0], cur, 0, key);
  if (cur.key < key)
    return eraseImpl(cur.children[1], cur, 1, key);
  --cur.duplicate;
  if (cur.duplicate > 0)
    return 1;
  //  assert(cur.duplicate == 0);
  if (cur.children[0] == null) {
    if (pnt != null)
      pnt.children[childId] = cur.children[1];
    if (cur.children[1] != null)
      cur.children[1].pnt = pnt;
    if (key == root.key)
      root = cur.children[1];
    return 1;
  }
  Node replacement = findLargest(cur.children[0]);
  if (key == root.key)
    root = replacement;
  //  assert(replacement.children[1] == null);
  if (replacement.key != cur.children[0].key) {
    replacement.pnt.children[1] = replacement.children[0];
    if (replacement.children[0] != null)
      replacement.children[0].pnt = replacement.pnt;
  }
  if (pnt != null)
    pnt.children[childId] = replacement;
  replacement.pnt = pnt;
  replacement.children[1] = cur.children[1];
  if (cur.children[1] != null)
    cur.children[1].pnt = replacement;
  if (replacement.key != cur.children[0].key) {
    replacement.children[0] = cur.children[0];
    cur.children[0].pnt = replacement;
  }
  return 1;
}

// return 1 if isIn
int erase(int key) {
  if (root == null)
    return 0;
  return eraseImpl(root, null, -1, key);
}
```

```
void printTree(Node cur) {
  if (cur == null)
    return;
  printTree(cur.children[0]);
  println(toString(cur.key) + ": " + toString(cur.duplicate));
  printTree(cur.children[1]);
}

int MAX = 128;
int MaxRandInt = ~(1 << 31);
int seed;

// In mx, we do not have unsigned int. Hence, we only use the least significant
// 31 bits of an integer here.
int randInt31() {
  int x = seed;
  x = x ^ (x << 13);
  x = x & ~(1 << 31);
  x = x ^ (x >> 17);
  x = x ^ (x << 5);
  x = x & ~(1 << 31);
  seed = x;
  return x;
}

// probability = p / PM
int randOp(int n) {
  if (randInt31() < n) {
    return 1;
  }
  return 2;
}

void generateOperations(int n, int t) {
  int i;
  for (i = 0; i < t; ++i) {
    int value = randInt31() % MAX;
    if (randOp(n) == 1) {
      insert(value);
    } else {
      erase(value);
    }
  }
}

int main() {
  seed = getInt();
  int m = 50000;
  generateOperations(8 * (MaxRandInt / 10), m);
  generateOperations(2 * (MaxRandInt / 10), 2 * m);
  generateOperations(4 * (MaxRandInt / 10), m);
  printTree(root);
  return 0;
}
寄
```