编译原理 · hw4

计01　容逸朗　2020010869

A1

```
E →    { E1.true := newlabel; E1.false := E.false } E1 ?
       { E2.true := newlabel; E2.false := E.false } E2 :
       { E3.true := E.false; E3.false := E.true } E3
       { E.code := E1.code || gen(E1.true ':') ||
           E2.code || gen(E2.true ':') || E3.code }
```

A2

```
S →    repeat { S1.next := newlabel } S1 until
       { E.true := S.next; E.false := S1.next } E
       { S.code := gen(S1.next ':') || S1.code || E.code || gen('goto' S1.next) }
```

A3

(a)

```
P →    D ; { S.inloop := 0 } S
       { P.type := if D.type = ok and S.type = ok then ok else type_error }

S →    if E then { S1.inloop := S.inloop } S1
       { S.type := if E.type = bool then S1.type else type_error }

S →    while E then {S1.inloop := 1 } S1
       { S.type := if E.type = bool then S1.type else type_error }

S →    { S1.inloop := S.inloop; S2.inloop := S.inloop } S1 ; S2
       { S.type := if S1.type = ok and S2.type = ok then ok else type_error }

S →    continue when E
       { S.type := if E.type = bool and S.inloop = 1 then ok else type_error }
```

(b)

```
P →   D ; { S.next := newlabel; S.continue := newlabel } S { gen(S.next ':') }

S →   if { E.true := newlabel; E.false := S.next } E then
      { S1.next := S.next; S1.continue := S.continue } S1
      { S.code := E.code || gen(E.true ':') || S1.code }

S →   while { E.true := newlabel; E.false := S.next } E do
      { S1.next := newlabel; S1.continue := S1.next } S1
      { S.code := gen(S1.next ':') || E.code || gen(E.true ':') ||
          S1.code || gen ('goto' S1.next) }

S →   { S1.next := newlabel; S1.countinue := S.continue } S1 ;
      { S2.next := S.next; S2.countinue := S.continue } S2
      { S.code := S1.code || gen(S1.next ':') || S2.code }

S →   continue when { E.true := S.continue; E.false := S.next }
      E { S.code := E.code }
```

A4

```
S →   switch A of { L.next := S.next } L end { S.code := A.code || L.code }

L →   case V : { S.next := newlabel } S ; { L1.next := L.next } L1
      { L.code := V.code || gen('if' A.place '≠' V.place 'goto' S.next) ||
          S.code || gen('goto' L.next) || gen(S.next ':') }

L →   default { S.next := L.next } S { L.code := S.code }

V →   d { V.place := newtemp; V.code := gen(V.place ':=' d.lexval) }
```

A5

(1)

```
A →   A1 + A2 { A.instr := A2.instr || A1.instr || Plus }

A →   A1 - A2 { A.instr := A2.instr || A1.instr || Minus }

A →   (A1) { A.instr := A1.instr }

A →   int { A.instr := Push int.val }
```

(2)

```
E →    E1 if B { E.instr := E1.instr || B.instr || Cond }
A →    id { A.instr := Load id.val}
B →    A1 > A2 { B.instr := A1.instr || A2.instr || Minus || Cmp || Push 1 || Minus }
B →    B1 & B2 { B.instr := B2.instr || B1.instr || Cond }
B →    !B1 { B.instr := B1.instr || Push 1 || Minus }
B →    true { B.instr := Push 1 }
B →    false { B.instr := Push 0 }
S →    id := E { S.instr := E.instr || Store id.val }
S →    S1 ; S2 { S.instr := S1.instr || S2.instr }
```

A6

```
E →    E1 ↑ E2 { E2.label := E.label; E2.case = not E.case; E1.case = false;
                 if E.case { E1.label := E.label; E.code := E1.code || E2.code }
                 else { E1.label := newlabel; E.code := E1.code ||
                 E2.code || gen(E1.label ':') }
               }
S →    repeat S1 until E
       { S1.next := newlabel; E.case := false; E.label := S1.next;
           S.code := gen(S1.next ':') || S1.code || E.code || gen(S.next ':') }
```