```
//                          M     M    A    RRRRRR   CCCCC  EEEEEEE L
//                          MM   MM   A A   R     R C     C E       L
//                          M M M M  A   A  R     R C       E       L
//                          M  M  M A     A RRRRRR  C       EEEEE   L
//                          M     M AAAAAAA R   R   C       E       L
//                          M     M A     A R    R  C     C E       L
//                          M     M A     A R     R  CCCCC  EEEEEEE LLLLLLL

//                            J  OOOOO  H     H N     N  SSSSS   OOOOO  N     N
//                            J O     O H     H NN    N S     S O     O NN    N
//                            J O     O H     H N N   N S       O     O N N   N
//                            J O     O HHHHHHH N  N  N  SSSSS  O     O N  N  N
//                        J   J O     O H     H N   N N       S O     O N   N N
//                        J   J O     O H     H N    NN S     S O     O N    NN
//                         JJJJJ   OOOOO  H     H N     N  SSSSS   OOOOO  N     N
```

```
// ** statistics ** 2 subprograms, no errors, no warnings
// ** statistics ** object size 258712 bytes
```

```
#include <vif.h>

static int_4 _km1 = -1, _k0 = 0, _k1 = 1;


#define _dc_0 " "
#define _dc_1 "| ROOF    | TIME TO ROOF/LOW/LANDING |"
#define _dc_2 " MAX SPEED   | FINAL SPEED | FUEL LEFT | GAS  |"
#define _dc_3 "|"
#define _dc_4 " M |"
#define _dc_5 "/"
#define _dc_6 " S   |"
#define _dc_7 " M/S   |"
#define _dc_8 " KG |"
#define _dc_9 "% |"
```

```
typedef char *FORMAT;
static FORMAT *__fmt_a = NULL;
typedef char char_1[2];
typedef char char_2[3];
```

```
#define __ncalls 3
static CALLS __calls[__ncalls] = {
  {"johnson", 0}, // program
  {"euler", 0}, // function
  {NULL, 0}
};
```

```
// common /_common/
static struct {
  real_8 deltat_;
} _common;
```

```
prototype static void _johnson (void);
prototype static inline real_8 _euler (real_8 _p_ currnt_, real_8 _p_ rate_);
```

```
//                          PPPPPP  RRRRRR   OOOOO   GGGGG  RRRRRR     A    M     M
//                          P     P R     R O     O G     G R     R   A A   MM   MM
//                          P     P R     R O     O G       R     R  A   A  M M M M
//                          PPPPPP  RRRRRR  O     O G  GGGG RRRRRR  A     A M  M  M
//                          P       R   R   O     O G     G R   R   AAAAAAA M     M
//                          P       R    R  O     O G     G R    R  A     A M     M
//                          P       R     R  OOOOO   GGGGG  R     R A     A M     M

//                                J  OOOOO  H     H N     N  SSSSS   OOOOO  N     N
//                                J O     O H     H NN    N S     S O     O NN    N
//                                J O     O H     H N N   N S       O     O N N   N
//                                J O     O HHHHHHH N  N  N  SSSSS  O     O N  N  N
//                          J     J O     O H     H N   N N       S O     O N   N N
//                          J     J O     O H     H N    NN S     S O     O N    NN
//                           JJJJJ   OOOOO  H     H N     N  SSSSS   OOOOO  N     N
```

```
// ** johnson    ** end of compilation 1
```

```
// I Line    ISN *....*....|....1....|....2....|....3....|....4....|....5....|....6....|....7..*.|....8
//      2        C THIS PROGRAM WAS A DEMO FOR A HIGH SCHOOL MATH GRADUATION PROJECT.
//      3        C
//      4        C NASA (FORMERLY NACA) USED LARGE 'MAINFRAME' COMPUTERS IN FOR EXAMPLE
//      5        C THE MERCURY, GEMINI AND APOLLO PROJECTS.
//      6        C IN THE MOVIE 'HIDDEN FIGURES' YOU CAN SEE AN IBM 7094 FROM 1962.
//      7        C LATER, NASA USED IBM 360'S IN THE APOLLO PROJECT.
//      8        C THIS PROGRAM IS AN EXAMPLE OF HOW A PROGRAM FROM THAT TIME MIGHT
//      9        C HAVE LOOKED LIKE.
//     10        C THIS IS VINTAGE FORTRAN CODE, AS WAS USED IN THE 1960'S. ALL TEXT
//     11        C IS UPPER-CASE ONLY AND VARIABLE NAMES ARE AT MOST SIX CHARACTERS.
//     12        C THIS PROGRAM USES THE EULER METHODE FOR INTEGRATION THAT WAS USED
//     13        C IN 'HIDDEN FIGURES'. IT IS MENTIONED WHEN KATHERINE WORKS ON THE
//     14        C FIRST MANNED FLIGHT IN THE MERCURY PROJECT:
//     15        C KATHERINE: 'THE PROBLEM IS WHEN THE CAPSULE MOVES FROM
//     16        C             AN ELLIPTICAL ORBIT TO A PARABOLIC ORBIT.
//     17        C             THERE IS NO MATHEMATICAL FORMULA FOR THAT.
//     18        C             ...
//     19        C HARRISON:  'MAYBE WE'RE THINKING ABOUT THIS ALL WRONG.'
//     20        C STAFFORD:  'HOW'S THAT?'
//     21        C HARRISON:  'MAYBE IT'S NOT NEW MATH AT ALL.'
//     22        C KATHERINE: 'MAYBE IT'S OLD MATH.
//     23        C             SOMETHING THAT LOOKS AT THE PROBLEM NUMERICALLY.
//     24        C             AND NOT THEORETICALLY. MATH IS ALWAYS DEPENDABLE.'
//     25        C HARRISON:  'FOR YOU IT IS.' (EXITS)
//     26        C KATHERINE: 'EULER'S METHOD.'
//     27        C STAFFORD:  'THAT'S ANCIENT.'
//     28        C KATHERINE: 'YES. BUT IT WORKS. IT WORKS NUMERICALLY.' (EXITS)
//     29        C THIS PROGRAM IS A SIMPLE DEMO FOR FINDING A STRATEGY TO SAFELY
//     30        C LAND A 'LUNAR EXCURSION MODULE' (LEM) ON THE MOON.
//     31        C A LEM STARTED ITS DESCENT FROM AN ALTITUDE OF 15 KILOMETER.
//     32        C IT WEIGHED 16 METRIC TONS AND HAD TO LAND WITH A SPEED NOT
//     33        C EXCEEDING 1 METER PER SECOND. OTHERWISE THE LEM WOULD GET
//     34        C DAMAGED, LEAVING ITS ASTRONAUTS MAROONED ON THE MOON.
//     35        C TO ASCEND, THE LEM HAD A SEPARATE ENGINE WITH ITS OWN FUEL.
//     36        C DURING DESCENT, FUEL COULD NOT BE EXHAUSTED SINCE A RESERVE
//     37        C WAS NEEDED FOR LAST-SECOND MANEUVERING TO FIND A GOOD LANDING SPOT.
//     38        C (THIS HAPPENED TO APOLLO 11).
//     39        C HERE THE PROBLEM IS SIMPLIFIED FOR THE SAKE OF DEMONSTRATION.
//     40        C THE STRATEGY IS THIS:
//     41        C DESCENT WITH IGNITED ENGINE TO REDUCE SPEED UNTIL SOME HEIGHT 'ROOF'.
//     42        C UP TO 'ROOF' WE ACCEPT A SAFE HIGH SPEED.
//     43        C THEN SPEED IS REDUCED.
//     44        C THE FINAL 300M WE REDUCE SPEED TO A LOW FINAL SPEED.
//     45        C IF DURING SPEED FUEL RUNS OUT WE ARE IN FREE FALL, AND THE SIMULATION
//     46        C GENERALLY FAILS SINCE THE LEM LIKELY CRASHES.
//     47        C THIS PROGRAM FINDS VALUES FOR 'ROOF' WHERE WE CAN LAND AT A SAFE
//     48        C SPEED OF AT MOST 1 METER PER SECOND.
//     49        C WE SOLVE IT HERE BY TAKING INCREASING VALUES FOR 'ROOF' AND
```

```
// I Line    ISN *....*....|....1....|....2....|....3....|....4....|....5....|....6....|....7..*.|....8
//      50        C SIMULATE THE LANDING - HOPING THE LEM LANDS SAFELY.
//      51        C ALTHOUGH THIS IS AN OVERSIMPLIFICATION OF THE ACTUAL LANDING
//      52        C PROCEDURES, RESULTS ARE COMPARABLE TO THOSE ACHIEVED IN ACTUAL
//      53        C LANDINGS ON THE MOON.
//      54       1       PROGRAM JOHNSON
//      55       2       IMPLICIT REAL*8 (A-Z)
//      56       3       COMMON DELTAT
//      57        C EULER INTEGRATION STEP DETERMINES ACCURACY OF COMPUTATION.
//      58        C 1/20 SECOND APPEARS GOOD ENOUGH.
//      59       4       DELTAT = 0.01
//      60        C FUEL COMSUMPTION IN KG/S AT FULL THROTTLE (ESTIMATED FROM DOCUMENTS).
//      61       5       CONSUM = 8
//      62        C THRUST IS 45 KILO-NEWTON (FROM LEM DOCUMENTATION).
//      63       6       MAXTHR = 45000
//      64        C GRAVITATIONAL ACCELARATION ON THE MOON IN M/S/S.
//      65       7       G = 1.62
//      66        C DESCENT STARTS AT 15 KM.
//      67       8       ORBIT = 15000
//      68        C THE LEM HAD 8200 KG FUEL FOR DESCENT. SOME OF THAT WAS USED TO GO
//      69        C FROM HIGH ORBIT AT 110 KM TO THE LOW ORBIT AT 15 KM.
//      70        C WE MAKE AN EDUCATED GUESS OF THE AMOUNT OF FUEL NEEDED TO GO FROM
//      71        C 110 KM DOWN TO 15 KM.
//      72       9       FULL = 8200 - 2000
//      73        C FROM 'LOW' ON, WE LAND CAREFULLY.
//      74      10       LOW = 300
//      75        C WE INCREASE 'ROOF' EVERY ITERATION, START LOW.
//      76      11       ROOF = ORBIT / 2
//      77        C DESCENT BEGINS HERE.
//      78      12     1 SPEED = 0
//      79      13       FASTST = 0
//      80      14       TIME = 0
//      81      15       HEIGHT = ORBIT
//      82        C FUEL IS FULL WHEN WE START DESCENDING.
//      83      16       FUEL = FULL
//      84        C WE RECORD PASSING 'ROOF' AND 'LOW' HEIGHTS.
//      85      17       ROFTIM = 0
//      86      18       LOWTIM = 0
//      87        C GAS IS ACTUAL THRUST.
//      88        C FOR THE LEM THAT WAS 10-60% OF FULL THROTTLE (FROM DOCUMENTATION).
//      89      19       GAS = 0.1
//      90        C LEM MASS IS 16 TON WITH FUEL.
//      91        C THE LEM LOOSES WEIGHT DURING DESCENT (BURNING FUEL).
//      92      20       EMPTY = 16000 - FULL
//      93        C ENGINE WILL BURN WHILE THERE IS FUEL.
//      94      21     2 IF (FUEL .GT. 0) GOTO 3
//      95        C OOPS! NO FUEL, FREE FALL, HELP!
//      96      22       SPEED = EULER (SPEED, G)
//      97      23       GOTO 4
```

```
// I Line    ISN *....*....|....1....|....2....|....3....|....4....|....5....|....6....|....7..*.|....8
//     98        C WE DO HAVE FUEL.
//     99        COMPUTE 'SAFE' SPEED DEPENDING ON HEIGHT.
//    100     24     3 SAFE = 1
//    101     25       IF (HEIGHT .GE. LOW) SAFE = 10
//    102     26       IF (HEIGHT .GE. ROOF) SAFE = 100
//    103        COMPUTE NEW STATE.
//    104        CURRENT MASS OF THE LEM.
//    105     27       MASS = EMPTY + FUEL
//    106        COMPUTE THRUST FROM NEWTON'S LAW F=M*A.
//    107     28       THRUST = MAXTHR / MASS * GAS
//    108     29       SPEED = EULER (SPEED, G - THRUST)
//    109     30       FUEL = EULER (FUEL, -CONSUM * GAS)
//    110     31       IF (FUEL .LT. 0) FUEL = 0
//    111     32     4 HEIGHT = EULER (HEIGHT, -SPEED)
//    112        C RECORD THINGS WE WANT TO KNOW LATER.
//    113     33       IF (HEIGHT .LT. ROOF .AND. ROFTIM .EQ. 0) ROFTIM = TIME
//    114     34       IF (HEIGHT .LT. LOW .AND. LOWTIM .EQ. 0) LOWTIM = TIME
//    115     35       IF (SPEED .GT. FASTST) FASTST = SPEED
//    116        CLOCK PROGRESSES A TINY BIT.
//    117     36       TIME = TIME + DELTAT
//    118        C IF THERE IS FUEL LEFT WE CORRECT THRUST.
//    119     37       IF (FUEL .EQ. 0) GO TO 5
//    120        C DO WE ASCEND? THEN REDUCE THRUST.
//    121     38       IF (SPEED .LE. 0) GAS = GAS - 0.05
//    122        C DO WE DESCEND TOO FAST? THEN INCREASE THRUST.
//    123     39       IF (SPEED .GE. SAFE) GAS = GAS + 0.05
//    124        C KEEP THRUST WITHIN LEM PARAMETERS (10-60%).
//    125     40       IF (GAS .GT. 0.6) GAS = 0.6
//    126     41       IF (GAS .LT. 0.1) GAS = 0.1
//    127        C WHILE NOT LANDED, PERFORM NEXT EULER ITERATION.
//    128     42     5 IF (HEIGHT .GT. 0) GO TO 2
//    129        C REPORT WHEN WE LANDED SAFELY.
//    130     43       IF (SPEED .GT. SAFE .OR. TIME .GT. 1500) GO TO 7
//    131     44       WRITE (6, 6) ROOF, ROFTIM, LOWTIM, TIME,
//    132     44      .             FASTST, SPEED, FUEL, GAS * 100
//    133     45     6 FORMAT (X, '| ROOF    | TIME TO ROOF/LOW/LANDING |',
//    134     45      .           ' MAX SPEED   | FINAL SPEED | FUEL LEFT | GAS  |'/
//    135     45      .        X, '|', F6.0, ' M |',
//    136     45      .        X, 2(F6.1, '/'), F6.1, ' S   |',
//    137     45      .        X, F5.1, ' M/S   |', X, F5.1, ' M/S   |',
//    138     45      .        X, F6.0, ' KG |' ,
//    139     45      .        X, F3.0, '% |')
//    140        CONSIDER A HIGHER ALTITUDE IN THE NEXT ATTEMPT.
//    141     46     7 ROOF = ROOF + 25
//    142        CONTINUE WHILE 'ROOF' IS BELOW 'ORBIT' (QUITE LOGICAL, CAPTAIN).
//    143     47       IF (ROOF .LT. ORBIT) GO TO 1
//    144        C WE'RE DONE, NO ATTEMPTS LEFTS.
//    145     48       STOP
```

```
// I Line    ISN *....*....|....1....|....2....|....3....|....4....|....5....|....6....|....7..*.|....8
//    146     49       END
```

```
// line     55 save real*8       deltat_
// line     60 save real*8       consum_
// line     62 save real*8       maxthr_
// line     64 save real*8       g_
// line     66 save real*8       orbit_
// line     71 save real*8       full_
// line     73 save real*8       low_
// line     75 save real*8       roof_
// line     77 save real*8       speed_
// line     78 save real*8       fastst_
// line     79 save real*8       time_
// line     80 save real*8       height_
// line     82 save real*8       fuel_
// line     84 save real*8       roftim_
// line     85 save real*8       lowtim_
// line     88 save real*8       gas_
// line     91 save real*8       empty_
// line     99 save real*8       safe_
// line    104 save real*8       mass_
// line    106 save real*8       thrust_

// label     1     1 in line    77, goto
// label     2     2 in line    93, goto
// label     3     3 in line    99, goto
// label     4     4 in line   110, goto
// label     5     5 in line   127, goto
// label     6     6 in line   132, non-executable
// label     7     7 in line   140, goto
```

```
// 8200 - 2000 = 6200
```

```
static void _johnson (void)
{
# line 53 "test-set/johnson.f" // PROGRAM JOHNSON
  static real_8 consum_, maxthr_, g_, orbit_, full_, low_, roof_, speed_, fastst_, time_, height_, fuel_, roftim_,
  lowtim_, gas_, empty_, safe_, mass_, thrust_;
  int_4 __fcnt, __rc;
  real_8 _t_0, _t_1, _t_2, _t_3;
# line 132 "test-set/johnson.f" // FORMAT (X, '| ROOF    | TIME TO ROOF/LOW/LANDING |',

  static FORMAT __fmt_6[] = {
    FMT_TEXT, _dc_0, _dc_0,
    FMT_TEXT, _dc_1, _dc_1,
    FMT_TEXT, _dc_2, _dc_2,
    FMT_TEXT, "\n", "\n",
    FMT_TEXT, _dc_0, _dc_0,
    FMT_TEXT, _dc_3, _dc_3,
    FMT_REAL, "%6f", "%6.0f",
    FMT_TEXT, _dc_4, _dc_4,
    FMT_TEXT, _dc_0, _dc_0,
    FMT_REAL, "%6f", "%6.1f",
    FMT_TEXT, _dc_5, _dc_5,
    FMT_REAL, "%6f", "%6.1f",
    FMT_TEXT, _dc_5, _dc_5,
    FMT_REAL, "%6f", "%6.1f",
    FMT_TEXT, _dc_6, _dc_6,
    FMT_TEXT, _dc_0, _dc_0,
    FMT_REAL, "%5f", "%5.1f",
    FMT_TEXT, _dc_7, _dc_7,
    FMT_TEXT, _dc_0, _dc_0,
    FMT_REAL, "%5f", "%5.1f",
    FMT_TEXT, _dc_7, _dc_7,
    FMT_TEXT, _dc_0, _dc_0,
    FMT_REAL, "%6f", "%6.0f",
    FMT_TEXT, _dc_8, _dc_8,
    FMT_TEXT, _dc_0, _dc_0,
    FMT_REAL, "%3f", "%3.0f",
    FMT_TEXT, _dc_9, _dc_9,
    NULL, NULL, NULL
  };
# line 58 "test-set/johnson.f" // DELTAT = 0.01
  _common.deltat_ = 0.01;
# line 60 "test-set/johnson.f" // CONSUM = 8
  consum_ = 8;
# line 62 "test-set/johnson.f" // MAXTHR = 45000
  maxthr_ = 45000;
# line 64 "test-set/johnson.f" // G = 1.62
  g_ = 1.62;
# line 66 "test-set/johnson.f" // ORBIT = 15000
  orbit_ = 15000;
# line 71 "test-set/johnson.f" // FULL = 8200 - 2000
```

```
  full_ = 6200;
# line 73 "test-set/johnson.f" // LOW = 300
  low_ = 300;
# line 75 "test-set/johnson.f" // ROOF = ORBIT / 2
  roof_ = orbit_ / 2;
  _l1:;
# line 77 "test-set/johnson.f" // SPEED = 0
  speed_ = 0;
# line 78 "test-set/johnson.f" // FASTST = 0
  fastst_ = 0;
# line 79 "test-set/johnson.f" // TIME = 0
  time_ = 0;
# line 80 "test-set/johnson.f" // HEIGHT = ORBIT
  height_ = orbit_;
# line 82 "test-set/johnson.f" // FUEL = FULL
  fuel_ = full_;
# line 84 "test-set/johnson.f" // ROFTIM = 0
  roftim_ = 0;
# line 85 "test-set/johnson.f" // LOWTIM = 0
  lowtim_ = 0;
# line 88 "test-set/johnson.f" // GAS = 0.1
  gas_ = 0.1;
# line 91 "test-set/johnson.f" // EMPTY = 16000 - FULL
  empty_ = 16000 - full_;
  _l2:;
# line 93 "test-set/johnson.f" // IF (FUEL .GT. 0) GOTO 3
  if (fuel_ > 0) {
    goto _l3;
  }
# line 95 "test-set/johnson.f" // SPEED = EULER (SPEED, G)
  speed_ = _euler (&speed_, &g_);
# line 96 "test-set/johnson.f" // GOTO 4
  goto _l4;
  _l3:;
# line 99 "test-set/johnson.f" // SAFE = 1
  safe_ = 1;
# line 100 "test-set/johnson.f" // IF (HEIGHT .GE. LOW) SAFE = 10
  if (height_ >= low_) {
    safe_ = 10;
  }
# line 101 "test-set/johnson.f" // IF (HEIGHT .GE. ROOF) SAFE = 100
  if (height_ >= roof_) {
    safe_ = 100;
  }
# line 104 "test-set/johnson.f" // MASS = EMPTY + FUEL
  mass_ = empty_ + fuel_;
# line 106 "test-set/johnson.f" // THRUST = MAXTHR / MASS * GAS
  thrust_ = maxthr_ / mass_ * gas_;
# line 107 "test-set/johnson.f" // SPEED = EULER (SPEED, G - THRUST)
  speed_ = _euler (&speed_, (_t_0 = g_ - thrust_, &_t_0));
```

```
# line 108 "test-set/johnson.f" // FUEL = EULER (FUEL, -CONSUM * GAS)
  fuel_ = _euler (&fuel_, (_t_1 = -consum_ * gas_, &_t_1));
# line 109 "test-set/johnson.f" // IF (FUEL .LT. 0) FUEL = 0
  if (fuel_ < 0) {
    fuel_ = 0;
  }
  _l4:;
# line 110 "test-set/johnson.f" // HEIGHT = EULER (HEIGHT, -SPEED)
  height_ = _euler (&height_, (_t_2 = -speed_, &_t_2));
# line 112 "test-set/johnson.f" // IF (HEIGHT .LT. ROOF .AND. ROFTIM .EQ. 0) ROFTIM = TIME
  if (height_ < roof_ && roftim_ == 0) {
    roftim_ = time_;
  }
# line 113 "test-set/johnson.f" // IF (HEIGHT .LT. LOW .AND. LOWTIM .EQ. 0) LOWTIM = TIME
  if (height_ < low_ && lowtim_ == 0) {
    lowtim_ = time_;
  }
# line 114 "test-set/johnson.f" // IF (SPEED .GT. FASTST) FASTST = SPEED
  if (speed_ > faststt_) {
    faststt_ = speed_;
  }
# line 116 "test-set/johnson.f" // TIME = TIME + DELTAT
  time_ = time_ + _common.deltat_;
# line 118 "test-set/johnson.f" // IF (FUEL .EQ. 0) GO TO 5
  if (fuel_ == 0) {
    goto _l5;
  }
# line 120 "test-set/johnson.f" // IF (SPEED .LE. 0) GAS = GAS - 0.05
  if (speed_ <= 0) {
    gas_ = gas_ - 0.05;
  }
# line 122 "test-set/johnson.f" // IF (SPEED .GE. SAFE) GAS = GAS + 0.05
  if (speed_ >= safe_) {
    gas_ = gas_ + 0.05;
  }
# line 124 "test-set/johnson.f" // IF (GAS .GT. 0.6) GAS = 0.6
  if (gas_ > 0.6) {
    gas_ = 0.6;
  }
# line 125 "test-set/johnson.f" // IF (GAS .LT. 0.1) GAS = 0.1
  if (gas_ < 0.1) {
    gas_ = 0.1;
  }
  _l5:;
# line 127 "test-set/johnson.f" // IF (HEIGHT .GT. 0) GO TO 2
  if (height_ > 0) {
    goto _l2;
  }
# line 129 "test-set/johnson.f" // IF (SPEED .GT. SAFE .OR. TIME .GT. 1500) GO TO 7
  if (speed_ > safe_ || time_ > 1500) {
```

```
    goto _l7;
  }
# line 130 "test-set/johnson.f" // WRITE (6, 6) ROOF, ROFTIM, LOWTIM, TIME,
  _fcheck ("johnson.f:johnson:130", 6, action_write, form_formatted);
  __fcnt = 0;
  while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
    __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
    _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
    __fcnt += 3;
  }
  if (__fmt_6[__fcnt] == NULL) {
    __fcnt = 0;
    __rc = fprintf (_ffile[6].unit, "\n");
    while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
      __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
      _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
      __fcnt += 3;
    }
  };
  __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], &roof_, REAL, 8);
  _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
  __fcnt += 3;
  while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
    __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
    _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
    __fcnt += 3;
  }
  if (__fmt_6[__fcnt] == NULL) {
    __fcnt = 0;
    __rc = fprintf (_ffile[6].unit, "\n");
    while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
      __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
      _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
      __fcnt += 3;
    }
  };
  __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], &roftim_, REAL, 8);
  _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
  __fcnt += 3;
  while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
    __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
    _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
    __fcnt += 3;
  }
  if (__fmt_6[__fcnt] == NULL) {
    __fcnt = 0;
    __rc = fprintf (_ffile[6].unit, "\n");
    while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
      __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
      _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
```

```
      __fcnt += 3;
    }
  };
  __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], &lowtim_, REAL, 8);
  _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
  __fcnt += 3;
  while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
    __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
    _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
    __fcnt += 3;
  }
  if (__fmt_6[__fcnt] == NULL) {
    __fcnt = 0;
    __rc = fprintf (_ffile[6].unit, "\n");
    while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
      __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
      _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
      __fcnt += 3;
    }
  };
  __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], &time_, REAL, 8);
  _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
  __fcnt += 3;
  while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
    __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
    _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
    __fcnt += 3;
  }
  if (__fmt_6[__fcnt] == NULL) {
    __fcnt = 0;
    __rc = fprintf (_ffile[6].unit, "\n");
    while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
      __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
      _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
      __fcnt += 3;
    }
  };
  __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], &faststt_, REAL, 8);
  _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
  __fcnt += 3;
  while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
    __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
    _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
    __fcnt += 3;
  }
  if (__fmt_6[__fcnt] == NULL) {
    __fcnt = 0;
    __rc = fprintf (_ffile[6].unit, "\n");
    while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
      __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
```

```
      _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
      __fcnt += 3;
    }
  };
  __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], &speed_, REAL, 8);
  _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
  __fcnt += 3;
  while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
    __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
    _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
    __fcnt += 3;
  }
  if (__fmt_6[__fcnt] == NULL) {
    __fcnt = 0;
    __rc = fprintf (_ffile[6].unit, "\n");
    while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
      __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
      _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
      __fcnt += 3;
    }
  };
  __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], &fuel_, REAL, 8);
  _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
  __fcnt += 3;
  while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
    __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
    _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
    __fcnt += 3;
  }
  _t_3 = gas_ * 100;
  if (__fmt_6[__fcnt] == NULL) {
    __fcnt = 0;
    __rc = fprintf (_ffile[6].unit, "\n");
    while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
      __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
      _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
      __fcnt += 3;
    }
  };
  __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], &_t_3, REAL, 8);
  _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
  __fcnt += 3;
  while (__fmt_6[__fcnt] != NULL && __fmt_6[__fcnt] == FMT_TEXT) {
    __rc = _vif_fprintf (6, __fmt_6[__fcnt + 2], NULL, NOTYPE, 0);
    _write_err (__rc, 6, _ioerr_write ("johnson.f:johnson:130", 6));
    __fcnt += 3;
  }
  _write_eol (6);
  _l7:;
# line 140 "test-set/johnson.f" // ROOF = ROOF + 25
```

```
  roof_ = roof_ + 25;
# line 142 "test-set/johnson.f" // IF (ROOF .LT. ORBIT) GO TO 1
  if (roof_ < orbit_) {
    goto _l1;
  }
# line 144 "test-set/johnson.f" // STOP
  exit (EXIT_SUCCESS);
# line 145 "test-set/johnson.f" // END
  __calls[0].calls++;
  return;
}
```

```
//                          FFFFFFF U     U N     N  CCCCC  TTTTTTT   III    OOOOO  N     N
//                          F       U     U NN    N C     C    T       I    O     O NN    N
//                          F       U     U N N   N C          T       I    O     O N N   N
//                          FFFFF   U     U N  N  N C          T       I    O     O N  N  N
//                          F       U     U N   N N C          T       I    O     O N   N N
//                          F       U     U N    NN C     C    T       I    O     O N    NN
//                          F        UUUUU  N     N  CCCCC     T      III    OOOOO  N     N

//                                   EEEEEEE U     U L       EEEEEEE RRRRRR
//                                   E       U     U L       E       R     R
//                                   E       U     U L       E       R     R
//                                   EEEEE   U     U L       EEEEE   RRRRRR
//                                   E       U     U L       E       R   R
//                                   E       U     U L       E       R    R
//                                   EEEEEEE  UUUUU  LLLLLLL EEEEEEE R     R
```

```
// ** euler      ** end of compilation 2
```

```
// I Line    ISN *....*....|....1....|....2....|....3....|....4....|....5....|....6....|....7..*.|....8
//    147     50       FUNCTION EULER (CURRNT, RATE)
//    148        C EULER INTEGRATION IS DONE WITH DOUBLE PRECISION.
//    149     51       IMPLICIT REAL*8 (A-Z)
//    150     52       COMMON DELTAT
//    151     53       EULER = CURRNT + RATE * DELTAT
//    152     54       RETURN
//    153     55       END
```

```
// line    146 save real*8       euler_
// line    146 save real*8       currnt_
// line    146 save real*8       rate_
// line    149 save real*8       deltat_
```

```
static inline real_8 _euler (real_8 _p_ currnt_, real_8 _p_ rate_)
{
# line 146 "test-set/johnson.f" // FUNCTION EULER (CURRNT, RATE)
  static real_8 euler_;
# line 150 "test-set/johnson.f" // EULER = CURRNT + RATE * DELTAT
  euler_ = *currnt_ + *rate_ * _common.deltat_;
# line 151 "test-set/johnson.f" // RETURN
  goto _l0;
# line 152 "test-set/johnson.f" // END
  _l0:;
  __calls[1].calls++;
  return euler_;
}
```

```
// Global entry point.
int_4 main (int_4 argc, char **argv)
{
  _vif_init ();
  _ffile[0] = NEW_FTN_FILE (NULL, form_formatted, action_readwrite, 0);
  _ffile[5] = NEW_FTN_FILE (stdin, form_formatted, action_read, MAX_LRECL);
  _ffile[5].buff = (char *) _malloc (MAX_LRECL + 1);
  _ffile[6] = NEW_FTN_FILE (stdout, form_formatted, action_write, 0);
  _ffile[7] = NEW_FTN_FILE (stdout, form_formatted, action_write, 0);
  _johnson (); // Fortran entry point.
  _vif_exit ();
  return EXIT_SUCCESS;
}
```