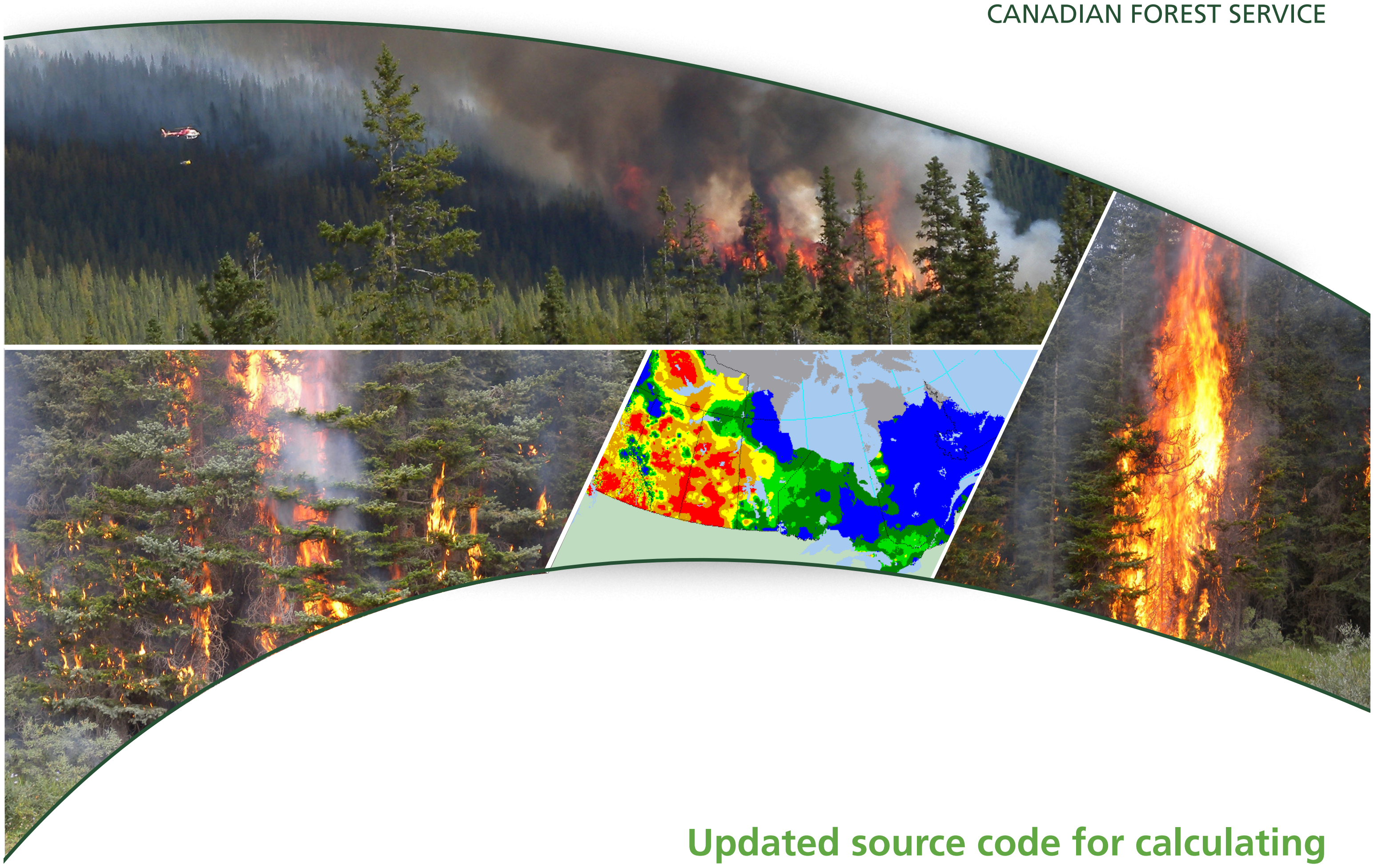
Natural Resources Canada Ressources naturelles Canada

CANADIAN FOREST SERVICE



# Updated source code for calculating fire danger indices in the Canadian Forest Fire Weather Index System

Y. Wang, K.R. Anderson, and R.M. Suddaby

INFORMATION REPORT
NOR-X-424

2015

Canada

# Updated source code for calculating fire danger indices in the Canadian Forest Fire Weather Index System

Y. Wang[1], K.R. Anderson[1], and R.M. Suddaby[1]

Information Report NOR-X-424

Canadian Forest Service
Northern Forestry Centre
2015

[1] Natural Resources Canada, Canadian Forest Service, 5320–122 Street, Edmonton, AB  T6H 3S5.

Natural Resources Canada
Canadian Forest Service
Northern Forestry Centre
5320 – 122 Street
Edmonton, AB T6H 3S5

Catalogue No.: Fo133-1/424E-PDF
ISBN 978-0-660-03779-0
ISSN 0831-8247

For a printed version of this report, visit the Canadian Forest Service publications website at http://cfs.nrcan.gc.ca/publications

Cette publication est également disponible en français sous le titre *Adaptation de l'aménagement forestier durable aux changements climatiques, rapport exhaustif sur les scénarios de la vulnérabilité.*

TTY : 613-996-4397 (Teletype for the hearing-impaired)
ATS: 613-996-4397 (appareil de télécommunication pour sourds)

Cover photos and map: Kerry Anderson

## Library and Archives Canada Cataloguing in Publication

Wang, Y, 1956-
Updated source code for calculating fire danger indices in the Canadian Forest Fire Weather Index System / Y. Wang, K.R. Anderson, and R.M. Suddaby.

(Information report ; NOR-X-424)

Electronic monograph in PDF format.

ISBN 978-0-660-03779-0
Cat. no.: Fo133-1/424E-PDF

1. Fire weather--Canada.
2. Forest fire forecasting--Canada.
3. Forest fires--Canada--Prevention and control.
I. Anderson, Kerry Robert, 1960-
II. Suddaby, Rod M
III. Northern Forestry Centre (Canada)
IV. Title.
V. Series: Information report (Northern Forestry Centre (Canada)); NOR-X-424

SD421.37 W36 2015 634.9'6180971 C2015-980050-1

**Wang, Y.; Anderson, K.R.; Suddaby, R.M. 2015. Updated source code for calculating fire danger indices in the Canadian Forest Fire Weather Index System. Nat. Resour. Can., Can. For. Serv., North. For. Cent., Edmonton, AB. Inf. Rep. NOR-X-424 .**

## ABSTRACT

This report presents updated versions of the FORTRAN 77 program originally published by the Canadian Forest Service in 1985 and used to calculate the Canadian forest fire weather indices from daily weather observations. The updated program is presented here in FORTRAN 95, C, C++, Python, Java and SAS/IML programming languages to meet the needs of various users. The updated versions are easier to understand and use than the original source code. The updated source codes were written in a modular programming style, consisting of a main program and several subroutines, to enable easy incorporation into other programs. The updated programs were tested against the original FORTRAN 77 code to ensure that consistent results would be obtained. The availability of these source codes allows users to confidently incorporate the Canadian forest fire weather indices into their own programs.

## RÉSUMÉ

Ce rapport présente les versions à jour du programme FORTRAN 77 publié à l'origine par le Service canadien des forêts en 1985 et utilisé pour calculer les indices forêt météo du Canada à partir des observations météorologiques quotidiennes. Le programme à jour est présenté ici en langage de programmation FORTRAN 95, C, C++, Python, Java et SAS/IML pour répondre aux besoins de divers utilisateurs. Les versions à jour sont plus faciles à comprendre et à utiliser que le code source d'origine. Les codes sources à jour ont été rédigés dans un style de programmation modulaire, qui consiste en un programme principal et plusieurs sous programmes, afin de permettre une intégration facile à d'autres programmes. Les programmes à jour ont été mis à l'essai par rapport au code FORTRAN 77 d'origine afin d'assurer l'obtention de résultats cohérents. La disponibilité de ces codes sources permet aux utilisateurs d'intégrer avec confiance les indices forêt-météo du Canada à leurs propres programmes.

## CONTENTS

INTRODUCTION . . . . . . . . . . . . . . . . . . . . . . . . . . . .1

VARIABLES USED IN THE FWI SYSTEM . . . . . . . . . . . . . . . . .2

FORTRAN 95, C, C++, PYTHON, JAVA, AND SAS/IML SOURCE CODES . .3

DISCUSSION AND CONCLUSIONS . . . . . . . . . . . . . . . . . . . 25

ACKNOWLEDGMENTS . . . . . . . . . . . . . . . . . . . . . . . . 25

LITERATURE CITED . . . . . . . . . . . . . . . . . . . . . . . . . 25

## TABLES

1. Source code for Canadian Forest Fire Weather Index System calculations in the FORTRAN 95 programming language . . . . . . .4

2. Source code for Canadian Forest Fire Weather Index System calculations in the C programming language . . . . . . . . . . . . .8

3. Source code for Canadian Forest Fire Weather Index System calculations in the C++ programming language . . . . . . . . . . 11

4. Source code for Canadian Forest Fire Weather Index System calculations in the Python programming language . . . . . . . . . 14

5. Source code for Canadian Forest Fire Weather Index System calculations in the Java programming language . . . . . . . . . . 17

6. Source code for Canadian Forest Fire Weather Index System calculations in the SAS/IML programming language . . . . . . . . 20

7. Sample of input data and result output . . . . . . . . . . . . . . 23

# INTRODUCTION

The Canadian Forest Fire Weather Index (FWI) System (Van Wagner and Pickett 1985; Van Wagner 1987) is a weather-based means of calculating potential fire conditions in Canada's forests. Canadian research on forest fire danger rating was started in the 1920s by scientists such as Wright (1933, 1937). The research focused mainly on tracking the day-to-day susceptibility of forests to fire. In the 1970s, several standard components of the FWI System were published in tabular format as the successor to a series of systems dating back to 1933 (Canadian Forestry Service 1984). Simard (1970) wrote a computer program to calculate the fire weather indices within the FWI System, and Engisch and Walker (1971), Kean (1975), and Van Wagner and Pickett (1975, 1985) subsequently modified and improved the program. Van Wagner (1987) published an updated version of the FWI System for calculating the fire weather indices, which was used in the FORTRAN 77 program published by Van Wagner and Pickett (1985).

The FWI System is used daily (and, in some cases, hourly) by all Canadian fire management agencies, and its outputs form the basis for describing the changing fire environment (e.g., Van Wagner 1987, Taylor and Alexander 2006). The system is also used in countries and regions outside Canada, including Europe (Šturm et al. 2012), New Zealand (Fogarty et al. 1998, Simpson et al. 2014), China (Tian et al. 2014), Indonesia (Field et al. 2004), Malaysia (de Groot et al. 2006), and the United States (Horel et al. 2014, Jandt et al. 2005).

The FWI System depends solely on weather measurements taken each day at noon local time. Daily inputs to the system consist of temperature (°C), relative humidity (%), wind speed (km/h), and precipitation (mm) over the past 24 hours. The FWI System outputs three moisture codes: the Fine Fuel Moisture Code (FFMC), the Duff Moisture Code (DMC), and the Drought Code (DC), with higher values indicating drier conditions and greater fire danger. The FWI System also generates three fire behavior outputs based on the above indices: the Initial Spread Index (ISI), the Build-Up Index (BUI), and the Fire Weather Index (FWI), with higher values indicating elevated fire danger. Detailed description of these indices can be found in Van Wagner (1987). It should be noted that the ISI calculated in the Technical Report by Van Wagner (1987) is different from the ISI calculated using the equations in the Information Report by Forestry Canada (1992).

The original program by Van Wagner and Pickett (1985) was written in the FORTRAN 77 programming language. Although the program accurately computes the fire danger rating indices, it has limitations, and its source code could be significantly improved to meet a higher programming standard. In this report, we present an updated version of the original FORTRAN 77 code. The updated source code has been written in a modular programming style, consisting of a main program and several subroutines, to enable easy incorporation into other programs. In total, we have coded the FWI System calculations in six widely used programming languages (FORTRAN 95, C, C++, Python, Java, and SAS/IML) to meet the needs of users with different programming backgrounds.

## VARIABLES USED IN THE FWI SYSTEM

The following variables were discussed and defined in Van Wagner and Pickett (1985) and used in their FORTRAN 77 source code:

| | | |
|---|---|---|
| T | = | Noon temperature (°C) |
| H | = | Noon relative humidity (%) |
| W | = | Noon wind speed (km/h) |
| Ro | = | Rain fall in open over previous 24 hours, at noon (mm) |
| Rf | = | Effective rain fall for calculating FFMC |
| Re | = | Effective rainfall for calculating DMC |
| Rd | = | Effective rainfall for calculating DC |
| Mo | = | Fine Fuel Moisture Content from previous day |
| Mr | = | Fine Fuel Moisture Content after rain |
| M | = | Fine Fuel Moisture Content after drying |
| Ed | = | Fine Fuel equilibrium moisture content (EMC) for drying |
| Ew | = | Fine Fuel EMC for wetting |
| Ko | = | Intermediate step in calculation of Kd |
| Kd | = | Log drying rate, FFMC drying rate, FFMC ln (M)/day |
| K1 | = | Intermediate step in calculation of Kw |
| Kw | = | Natural log wetting rate, ln (M)/day |
| Fo | = | Previous day's FFMC |
| F | = | Fine Fuel Moisture Code |
| DMo | = | Duff Moisture Content from previous day |
| DMr | = | Duff moisture content after rain |
| DM | = | Duff moisture content after drying |
| K | = | Log drying rate in DMC, ln (M)/day |
| Le | = | Effective day length in DMC, hours |
| B | = | Slope variable in DMC rain effect |
| Po | = | Previous day's DMC |
| Pr | = | DMC after rain |
| P | = | DMC |
| Q | = | Moisture equivalent of DC, units of 0.254 mm |
| Qo | = | Moisture equivalent of previous day's DC |
| Qr | = | Moisture equivalent after rain |
| V | = | Potential evapotranspiration, units of 0.254 mm water/day |
| Lf | = | Day-length adjustment in DC |
| Do | = | Previous day's DC |
| Dr | = | DC after rain |
| D | = | DC |
| Fw | = | Wind function |
| Ff | = | Fine fuel moisture function |
| Fd | = | Duff moisture function |
| R | = | Initial spread index |
| U | = | Buildup index |
| B | = | Intermediate FWI |
| S | = | Final FWI |

## FORTRAN 95, C, C++, PYTHON, JAVA, AND SAS/IML SOURCE CODES

New source codes were written according to the equations and methodology described by Van Wagner (1987). These codes are presented in Tables 1 to 6, respectively. To the extent possible, variable names follow the convention used by Van Wagner (1987). Comment lines throughout the codes show the correspondence between equations used for calculating indices in the FWI System and key statements, to make the statements as self-explanatory as possible even for those with limited programming knowledge.

The new codes were tested against the standard FWI System test dataset provided by Van Wagner and Pickett (1985). The text data file used as the input file contains 49 lines of daily weather observations (corresponding to 49 days from April 13 to May 31, including data for month and day, temperature, relative humidity, wind speed, and precipitation). The output FFMC, DMC, DC, ISI, BUI, and FWI values produced by the six program source codes, as run on a Linux platform, were identical with those produced by the FORTRAN 77 source code. The test data from Van Wagner and Pickett (1985) were included in Table 7 so that users can test these codes. An input data file is required for running all six programs, which is named data.txt in this report. The data file consists of six data fields of month, day, temperature, relative humidity, wind speed, and precipitation. The results of running the programs are written to the output file fwioutput.txt for all programs.

**Table 1. Source code for Canadian Forest Fire Weather Index System calculations in the FORTRAN 95 programming language**

```
module fwiindices !!!************ Beginning of Module ***********!!!
implicit none
real, save :: Mo,Rf,Ed,Ew,M,Kl,Kw,K,Pr,Ra,Re,B,Mr,V,Rd,Dr,Qo
real, save :: Fw,Ff,P,El(12),Fl(12),Qr,D0,Kd,Ko,Fd

data El/6.5,7.5,9.0,12.8,13.9,13.9,12.4,10.9,9.4,8.0,7.0,6.0/
data Fl/-1.6,-1.6,-1.6,0.9,3.8,5.8,6.4,5.0,2.4,0.4,-1.6,-1.6/

private :: Mo,Rf,Ed,Ew,M,Kl,Kw,K,Pr,Ra,Re,B,Mr,V,Rd,Dr,Qo
private :: Fw,Ff,P,Qr,D0,El,Fl,Kd,Ko,Fd

Contains

! Subroutine FFMC calculation

subroutine FFMCcalc(T,H,W,Ro,Fo, ffmc)
implicit none
   real, intent(in)  :: T,H,W,Ro,Fo
   real, intent(out) :: ffmc

   Mo = (147.2*(101.0 - Fo))/(59.5 + Fo)                                   !*Eq. 1*!
   if (Ro > 0.5) then
      rf = Ro - 0.5                                                         !*Eq. 2*!
      if(Mo <= 150.0) then
          Mr = Mo+42.5*Rf*exp(-100.0/(251.0-Mo))*(1.0 - exp(-6.93/Rf))     !*Eq. 3a*!
      else

          Mr = (Mo+42.5*Rf*exp(-100.0/(251.0-Mo))*(1.0 - exp(-6.93/Rf))) + &
           (.0015*(Mo - 150.0)**2)*sqrt(Rf)                                 !*Eq.3b*!
      endif

      if (Mr > 250.0) Mr = 250.0
      Mo = Mr
   endif

   Ed =.942*(H**.679) + (11.0*exp((H-100.0)/10.0))+0.18*(21.1-T) &
   (1.0 - 1.0/exp(.1150 * H))                                               !*Eq. 4*!

   if(Mo > Ed) then
      Ko = .424*(1.0-(H/100.0)**1.7)+(.0694*sqrt(W))*(1.0- (H/100.0)**8)   !*Eq. 6a*!
      Kd = Ko * (.581*exp(.0365*T))                                        !*Eq. 6b*!
      M = Ed + (mo-ed)/10.0 ** Kd                                          !*Eq. 8*!
   else
     Ew = .618*(H**.753) + (10.0*exp((H-100.0)/10.0)) + .18*(21.1-T) * &
     (1.0 - 1.0/exp(.115 * H))                                              !*Eq. 5*!
     if(Mo < Ew) then

        Kl = .424*(1.0-((100.0 - H)/100.0)**1.7) + (.0694*sqrt(W)) * & (1.0
        - ((100.0 - H)/100.0)**8)
        Kw = Kl * (.581 * exp(.0365 * T))
        M = Ew - (Ew - Mo)/10.0**Kw
      else
        M = Mo                                                              !*Eq. 7a*!
      endif                                                                 !*Eq. 7b*!
   endif                                                                    !*Eq. 9*!

   ffmc = (59.5 * (250.0 - m)) / (147.2 + m)                               !*Eq. 10*!
   if (ffmc  > 101.0)  ffmc = 101.0
   if (ffmc  <=  0.0)  ffmc =   0.0

   return

end subroutine
```

**Table 1. Continued**

```
! Subroutine DMC calculation

subroutine DMCcalc(T, H, Ro, Po, I, dmc)
implicit none
   integer,  intent(in)  :: I
   real,     intent(in)  :: T,H,Ro,Po
   real,     intent(out) :: dmc

   if(Ro <= 1.5) then
      Pr = Po
   else
      Re = 0.92*Ro - 1.27                                              !*Eq. 11*!
      Mo = 20.0 + 280.0/exp(0.023*Po)                                  !*Eq. 12*!
      if(Po <= 33.0) then
         B = 100.0 /(0.5 + 0.3*Po)                                          !*Eq.
      else                                                                  13a*!
         B =  6.2 * log(Po) - 17.2
         if(Po-65.0 <= 0.0)  B = 14.0 - 1.3*log(Po)                         !*Eq.
      endif                                                                 13c*!
                                                                            !*Eq.
                                                                            13b*!

      Mr = Mo + (1000.0*Re) / (48.77+B*Re)
      Pr = 43.43 * (5.6348 - log(Mr-20.0))                             !*Eq. 14*!
   endif                                                               !*Eq. 15*!

   if(T >= -1.1) then
      K = 1.894*(T+1.1) * (100.0-H) * (El(I)*0.0001)                   !*Eq. 15*!
   else
      K = 0.0                                                          !*Eq. 17*!
   endif

   if(Pr <  0.0) Pr = 0.0
   dmc = Pr + K
   if(dmc <= 0.0) dmc = 0.0                                               return

   return

end subroutine

!   Subroutine DC calculation

subroutine DCcalc (T, Ro, D0, I, dc)
implicit none
   real, intent(in)    :: T, Ro
   real, intent(inout) :: D0
   real, intent(out)   :: dc
   integer,intent(in)  :: I

   if (Ro > 2.8) then
      Rd = 0.83*Ro - 1.27                                              !*Eq. 18*!
      Qo = 800.0 * exp(-D0/400.0)                                      !*Eq. 19*!
      Qr = Qo + 3.937*Rd                                               !*Eq. 20*!
      Dr = 400.0*log(800.0/Qr)                                         !*Eq. 21*!
      if(Dr > 0.0) then
         D0 = Dr
      else                                                                  else
         D0 = 0.0
      endif
   endif

   if(T < -2.8) then
      V = Fl(I)
   else
      V = (0.36*(T+2.8) + Fl(I))                                       !*Eq. 22*!
   endif
   if (V <= 0.0) V = 0.0
   dc = D0 + 0.5*V

   return

end subroutine
```

**Table 1. Continued**

```
! Subroutine for ISI

subroutine ISICalc(F,W, isi)
implicit none

      real,intent(in) :: F, W
      real,intent(out):: isi

      Mo  =  (147.2*(101.0-F)) / (59.5+F)                                !*Eq. 1*!
      Fw  =  exp(0.05039*W)                                             !*Eq. 24*!
      Ff  =  91.9*exp(-0.1386*Mo) * (1.0+(Mo**5.31)/4.93e7)             !*Eq. 25*!
      isi =  0.208 * Fw * Ff                                            !*Eq. 26*!

      return

end subroutine

! Subroutine for BUI

subroutine BUICalc(P,D,bui)
implicit none

   real, intent(in) :: P,D
   real, intent(out):: bui

   if (P <= 0.4*D) then
      bui = (0.8*P*D) / (P+0.4*D)                                          !*Eq.
   else                                                                    27a*!
      bui = P-(1.0-0.8*D/(P+0.4*D))*(0.92+(0.0114*P)**1.7)
   endif                                                                   !*Eq.
                                                                           27b*!

   if (bui<0.0) bui = 0.0

   return

end subroutine

! Subroutine for FWI

subroutine FWICalc(R,U,fwi)
implicit none

   real, intent(in) :: R,U
   real, intent(out):: fwi

   if (U <= 80.0) then
      Fd = 0.626*U**0.809 + 2.0                                            !*Eq.
   else                                                                    28a*!
      Fd = 1000.0/(25.0+108.64*exp(-0.023*U))
   endif                                                                   !*Eq.
                                                                           28b*!

   B = 0.1*R*Fd                                                         !*Eq. 29*!
   if(B > 1.0) then
      fwi = exp(2.72*(0.434*log(B))**0.647)                                !*Eq.
   else                                                                    30a*!
      fwi = B
   endif                                                                   !*Eq.
                                                                           30b*!

   return

end subroutine

end module !!!****** End of Module ***********!!!
```

**Table 1. Concluded**

```
!        Main program

program  fwicalculation
use fwiindices
implicit none
   real     ::  temp,rhum,prcp,wind                          !!! four weather variables
   real     ::  ffmc, dmc, dc                    !!! FFMC, DMC, and DC to be calculated
   real     ::  isi, bui, fwi, dsr          !!! ISI, BUI, FWI, and DSR to be calculated
   real     ::  ffmc0, dmc0, dc0                 !!! initial or yesterday's FFMC, DMC,
                                                                         and DC values
   integer  ::  mth, day                         !!! month and day for each observation
   integer  ::  io                                                !!! utilit y integer
   ffmc0 = 85.0                                                 !!! First z initialize
   dmc0  =  6.0                                                  !!! FFMC, DMC, and DC
   dc0   = 15.0                                                               !!! here

   open(unit = 1, file = 'data.txt', status='old')                  !!! open data file
   open(unit = 2, file = 'fwioutput.txt',status='new')                 !!! output file
   do

      read(1,*,iostat=io) mth,day,temp,rhum,wind,prcp      !!! read in data day by day
      if(io/=0) exit                                          !!! reach the end of file
      if (rhum>= 100.0) rhum=100.0                         !!! make sure RHUM <= 100.0
      call FFMCcalc(temp,rhum,wind,prcp,ffmc0,ffmc)                 !!! calculate FFMC
      call DMCcalc (temp,rhum,prcp,dmc0,mth,dmc)                     !!! calcualte DMC
      call DCcalc  (temp,prcp,dc0,mth,dc)                             !!! calcualte DC
      call ISIcalc (ffmc,wind,isi)                                   !!! calcualte ISI
      call BUIcalc (dmc,dc,bui)                                      !!! calculate BLL
      call FWIcalc (isi,bui,fwi)                                     !!! calculate FWI

      ffmc0 = ffmc                                              !!! use today's values
      dmc0  = dmc                                                   !!! for tomorrow's
      dc0   = dc                                                       !!! calculation

      write(2,10) mth,day,temp,rhum,wind,prcp,&                     !!! output results
            ffmc, dmc, dc, isi, bui, fwi                  !!! to scereen (or to file)
10    format(i4,i5,4f6.1,6f8.1,f9.2)                             !!! output data format

   enddo
   close (1)
   close (2)

end program fwi calculation
```

**Table 2. Source code for Canadian Forest Fire Weather Index System calculations in the C programming language**

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/*         FFMC calculation             */
void FFMCcalc(float T,float H,float W,float Ro,float Fo, float *ffmc) {
     float Mo,Rf,Mr,Ed,Ew,Ko,Kd,Kl,Kw,M,F;

     Mo = 147.2*(101.-Fo)/(59.5+Fo);                                   /*Eq. 1  in */
     if(Ro > 0.5) {                                  /*van Wagner and Pickett (1985) */
        Rf = Ro - 0.5;                                                       /*Eq. 2*/
        if(Mo <= 150.)
           Mr = Mo +
           42.5*Rf*(exp(-100./(251.-Mo)))*(1-exp(-6.93/Rf));                /*Eq. 3a*/
        else
           Mr = Mo +
                42.5*Rf*(exp(-100./(251.-Mo)))*(1-exp(-6.93/Rf)) +
                .0015*pow(Mo-150.,2.)*pow(Rf,.5);                           /*Eq. 3b*/
        if(Mr > 250.) Mr = 250.;
        Mo = Mr;
     }
     Ed = 0.942*pow(H,.679)
          + 11.*exp((H-100.)/10.)+.18*(21.1-T)*(1.-exp(-.115*H));            /*Eq. 4*/
     if(Mo > Ed) {
        Ko = 0.424*(1.-pow(H/100.,1.7))
             + 0.0694*pow(W,.5)*(1.-pow(H/100.,8.));                        /*Eq. 6a*/
        Kd = Ko*.581*exp(0.0365*T);                                         /*Eq. 6b*/
        M = Ed+(Mo-Ed)*pow(10.,-Kd);                                         /*Eq. 8*/
     }
     else {
        Ew = 0.618*pow(H,.753)
          + 10.*exp((H-100.)/10.)
          + .18*(21.1-T)*(1.-exp(-.115*H));                                  /*Eq. 5*/
        if(Mo < Ew) {
           Kl = 0.424*(1.-pow((100.-H)/100.,1.7))
            + 0.0694*pow(W,.5)*(1-pow((100.-H)/100.,8.));                   /*Eq. 7a*/
           Kw = Kl*.581*exp(0.0365*T);                                      /*Eq. 7b*/
           M = Ew - (Ew - Mo)*pow(10.,-Kw);                                  /*Eq. 9*/
        }
        else M = Mo;
     }
     *ffmc = 59.5*(250.-M)/(147.2+M);                                       /*Eq. 10*/
     if (*ffmc > 101.0) *ffmc=101.0;
}
```

**Table 2. Continued**

```
/*                DMC calculation           */
void DMCcalc(float T,float H,float Ro,float Po,int I, float *dmc)  {
     float  Re,Mo,Mr,K,B,P,Pr;
     static float Le[] = {6.5,7.5,9.,12.8,13.9,13.9,12.4,10.9,9.4,8.,7.,6.};

     if(T >= -1.1) K = 1.894*(T +1.1)*(100.-H)*Le[I-1]*0.0001;             /*Eq. 16*/
     else K = 0.;                                                           /*Eq. 17*/

     if(Ro <= 1.5) Pr = Po;
     else {
        Re = 0.92*Ro-1.27;                                                  /*Eq. 11*/
        Mo = 20. + 280.0/exp(0.023*Po);                                     /*Eq. 12*/
        if(Po <= 33.) B = 100./(.5+.3*Po);                                 /*Eq. 13a*/
        else {
           if(Po <= 65.) B = 14. - 1.3*log(Po);                            /*Eq. 13b*/
           else B = 6.2*log(Po) - 17.2;
        }                                                                  /*Eq. 13c*/
        Mr = Mo +1000.*Re/(48.77+B*Re);                                     /*Eq. 14*/
        Pr =  43.43*(5.6348 -log(Mr - 20.));                                /*Eq. 15*/
     }
     if(Pr < 0.) Pr=0.0;
     *dmc = Pr + K;
     if(*dmc <= 0.0) *dmc = 0.0;
}
/*          DC calculation            */
void DCcalc(float T,float Ro,float Do,int I, float *dc) {
     float Rd,Qo,Qr,V,D,Dr;
     static float Lf[] = {-1.6,-1.6,-1.6,.9,3.8,5.8,6.4,5.,2.4,.4,-1.6,-1.6};

     if(Ro > 2.8) {
        Rd = 0.83*(Ro) - 1.27;                                              /*Eq. 18*/
        Qo = 800.*exp(-Do/400.);                                            /*Eq. 19*/
        Qr = Qo +3.937*Rd;                                                  /*Eq. 20*/
        Dr = 400.*log(800./Qr);                                             /*Eq. 21*/
        if(Dr > 0.) Do=Dr;
        else        Do=0.0;
     }
     if(T > -2.8) V = 0.36*(T+2.8)+Lf[I-1];                                 /*Eq. 22*/
     else V = Lf[I-1];
     if(V < 0.) V=.0;
     *dc = Do + 0.5*V;                                                      /*Eq. 23*/

}
/*             ISI calculation         */

void ISIcalc(float F,float W, float *isi) {
     float Fw,M,Ff,R;
     M = 147.2*(101-F)/(59.5+F);                                             /*Eq. 1*/
     Fw = exp(0.05039*W);                                                   /*Eq. 24*/
     Ff = 91.9*exp(-.1386*M)*(1.+pow(M,5.31)/4.93E7);                       /*Eq. 25*/
     *isi = 0.208*Fw*Ff;                                                    /*Eq. 26*/
}
/*             BUI calculation         */

void BUIcalc(float P,float D, float *bui)  {
     float U;
     if(P <= .4*D) *bui = 0.8*P*D/(P+.4*D);                                /*Eq. 27a*/
     else *bui = P - (1.-.8*D/(P+.4*D))*(.92+pow(.0114*P,1.7));            /*Eq. 27b*/
     if (*bui < 0.0) *bui = 0.0;
}
```

**Table 2. Concluded**

```
/*               FWI calculation         */

void FWIcalc(float R,float U, float *fwi)  {
  float Fd,B,S;
     if(U <= 80.) Fd = .626*pow(U,.809)+2.;                                /*Eq. 28a*/
     else Fd = 1000./(25.+108.64*exp(-.023*U));                            /*Eq. 28b*/
     B = .1*R*Fd;                                                           /*Eq. 29*/
     if(B > 1.) *fwi = exp(2.72*pow(.434*log(B),.647));                    /*Eq. 30a*/
     else *fwi = B;                                                        /*Eq. 30b*/
}

/*              Main Program                  */

int main(){
   float   temp, rhum, wind, prcp, ffmc0, dmc0, dc0;
   float   ffmc, dmc, dc, isi, bui, fwi;
   int      mth, day;

   FILE *weather_data = fopen("data.txt","r");
   if(weather_data == NULL) {
      printf("Data file is not opened \n");
      return -1;
   }
   FILE *output_data = fopen("fwioutput.txt","a");
   if(output_data == NULL) {
      printf("Output file is not opened \n");
      return -1;
   }

   ffmc0 = 85.0; dmc0 = 6.0; dc0 = 15.0;   /* FFMC, DMC, and DC initialization */

   while(!feof(weather_data)) {
      fscanf(weather_data,"%d" "%d" "%f" "%f" "%f"
                        "%f",&mth,&day,&temp,&rhum,&wind,&prcp);
      if (rhum > 100.0) rhum = 100.0;
      if (feof(weather_data)) break;

      FFMCcalc(temp,rhum,wind,prcp,ffmc0,&ffmc);
      DMCcalc(temp,rhum,prcp,dmc0,mth, &dmc);
      DCcalc(temp,prcp,dc0,mth,&dc);
      ISIcalc(ffmc,wind,&isi);
      BUIcalc(dmc,dc,&bui);
      FWIcalc(isi,bui,&fwi);
      ffmc0 = ffmc; dmc0 = dmc; dc0  = dc;
      fprintf(output_data,"%f %f %f %f %f %f \n",ffmc,dmc,dc,isi,bui,fwi);
   }
   fclose(weather_data);
   fclose(output_data);
}
```

**Table 3. Source code for Canadian Forest Fire Weather Index System calculations in the C++ programming language**

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <math.h>
#include <stdlib.h>
using namespace std;

void FFMCcalc(float T,float H,float W,float Ro,float Fo,float& ffmc) {
     float  Mo,Rf,Ed,Ew,M,Kl,Kw,Mr,Ko,Kd;

     Mo = 147.2*(101.-Fo)/(59.5+Fo);                                    /*Eq. 1  in */
     if(Ro > 0.5) {                                    /*van Wagner and Pickett (1985)*/
        Rf = Ro - 0.5;                                                         /*Eq.2*/
        if(Mo <= 150.)
           Mr = Mo +
           42.5*Rf*(exp(-100./(251.-Mo)))*(1-exp(-6.93/Rf));                 /*Eq. 3a*/
        else
           Mr = Mo +
                42.5*Rf*(exp(-100./(251.-Mo)))*(1-exp(-6.93/Rf)) +
                .0015*pow(Mo-150.,2.)*pow(Rf,.5);                            /*Eq. 3b*/
        if(Mr > 250.) Mr = 250.;
        Mo = Mr;
     }
     Ed = 0.942*pow(H,.679)
          + 11.*exp((H-100.)/10.)+.18*(21.1-T)*(1.-exp(-.115*H));             /*Eq. 4*/
     if(Mo > Ed) {
        Ko = 0.424*(1.-pow(H/100.,1.7))
             + 0.0694*pow(W,.5)*(1.-pow(H/100.,8.));                          /*Eq. 6a*/
        Kd = Ko*.581*exp(0.0365*T);                                          /*Eq. 6b*/
        M = Ed+(Mo-Ed)*pow(10.,-Kd);                                          /*Eq. 8*/
     }
     else {
        Ew = 0.618*pow(H,.753)
          + 10.*exp((H-100.)/10.)
          + .18*(21.1-T)*(1.-exp(-.115*H));                                   /*Eq. 5*/
        if(Mo < Ew) {
           Kl = 0.424*(1.-pow((100.-H)/100.,1.7))
            + 0.0694*pow(W,.5)*(1-pow((100.-H)/100.,8.));                    /*Eq. 7a*/
           Kw = Kl*.581*exp(0.0365*T);                                       /*Eq. 7b*/
           M = Ew - (Ew - Mo)*pow(10.,-Kw);                                   /*Eq. 9*/
        }
        else M = Mo;
     }
/*Finally calculate FFMC    */

      ffmc = (59.5 * (250.0 - M)) / (147.2 + M)   ;

/*..............................*/
/*Make sure 0. <= FFMC <= 101.0 */
/*..............................*/
      if (ffmc  > 101.0)  ffmc = 101.0;
      if (ffmc  <=  0.0)  ffmc = 0.0;
}
```

**Table 3. Continued**

```
/*               DMC calculation            */

void DMCcalc(float T,float H,float Ro,float Po,int I, float& dmc) {
     float  Re,Mo,Mr,K,B,P,Pr;
     float Le[] = {6.5,7.5,9.,12.8,13.9,13.9,12.4,10.9,9.4,8.,7.,6.};

     if(T >= -1.1) K = 1.894*(T +1.1)*(100.-H)*Le[I-1]*0.0001;          /*Eq. 16*/
     else K = 0.;                                                        /*Eq. 17*/

     if(Ro <= 1.5) Pr = Po;
     else {
        Re = 0.92*Ro-1.27;                                               /*Eq. 11*/
        Mo = 20. + 280.0/exp(0.023*Po);                                  /*Eq. 12*/
        if(Po <= 33.) B = 100./(.5+.3*Po);                              /*Eq. 13a*/
        else {
           if(Po <= 65.) B = 14. - 1.3*log(Po);                         /*Eq. 13b*/
           else B = 6.2*log(Po) - 17.2;
        }                                                               /*Eq. 13c*/
        Mr = Mo +1000.*Re/(48.77+B*Re);                                  /*Eq. 14*/
        Pr =  43.43*(5.6348 -log(Mr - 20.));                             /*Eq. 15*/
     }
     If (Pr < 0.) Pr = 0.0;
     P   = Pr + K;
     If (P   <=0.0) P = 0.0;
     dmc = P;
}

/*         DC calculation             */

void DCcalc(float T,float Ro,float Do,int I, float& dc) {
     float Rd,Qo,Qr,V,D,Dr;
     float Lf[] = {-1.6,-1.6,-1.6,.9,3.8,5.8,6.4,5.,2.4,.4,-1.6,-1.6};

     if(Ro > 2.8) {

        Rd = 0.83*(Ro) - 1.27;                                           /*Eq. 18*/
        Qo = 800.*exp(-Do/400.);                                         /*Eq. 19*/
        Qr = Qo +3.937*Rd;                                               /*Eq. 20*/
        Dr = 400.*log(800./Qr);                                          /*Eq. 21*/
        If (Dr > 0.) Do = Dr;
        else        Do = 0.0;
     }
     If (T > -2.8) V = 0.36*(T+2.8)+Lf[I-1];                             /*Eq. 22*/
     else V = Lf[I-1];
     if (V < 0.) V=.0;                                                   /*Eq. 23*/
     dc = Do + 0.5*V;
}

/*            ISI calculation         */

void ISIcalc(float F,float W,float& isi) {
     float  Fw,M,Ff;
     M = 147.2*(101-F)/(59.5+F);                                          /*Eq. 1*/
     Fw = exp(0.05039*W);                                                /*Eq. 24*/
     Ff = 91.9*exp(-.1386*M)*(1.+pow(M,5.31)/4.93E7);                    /*Eq. 25*/
     isi = 0.208*Fw*Ff;                                                  /*Eq. 26*/
}

/*            BUI calculation         */

void BUIcalc(float P,float D, float& bui) {
     if(P <= .4*D) bui = 0.8*P*D/(P+.4*D);                              /*Eq. 27a*/
     else bui = P - (1.-.8*D/(P+.4*D))*(.92+pow(.0114*P,1.7));          /*Eq. 27b*/

     if (bui < = 0.0) bui = 0.0;
}
```

**Table 3. Concluded**

```
/*               FWI calculation         */

void FWIcalc(float R,float U, float& fwi) {
  float Fd,B,S;
     if(U <= 80.) Fd = .626*pow(U,.809)+2.;                                 /*Eq. 28a*/
     else Fd = 1000./(25.+108.64*exp(-.023*U));                             /*Eq. 28b*/
     B = .1*R*Fd;                                                            /*Eq. 29*/
     if(B > 1.) fwi = exp(2.72*pow(.434*log(B),.647));                      /*Eq. 30a*/
     else fwi = B;                                                          /*Eq. 30b*/
}

int main() {
      string   line;
      float    temp, rhum, wind, prcp, x, y;
      float    ffmc, ffmc0, dmc, dmc0, dc, dc0, isi, bui, fwi;
      int      month, day;

/*       Initialize FMC, DMC, and DC              */

      ffmc0 = 85.0;
      dmc0 = 6.0;
      dc0   = 15.0;

/*       Open input and output files              */

      ifstream inputFile("data.txt");
      if(!inputFile.is_open()) {
             cout << "Unable to open input data file";
             return 0;
      }
      ofstream outputFile("fwioutput.txt");
      if(!outputFile.is_open()) {
             cout << "Unable to open output data file";
             return 0;
      }

/*        Main loop for calculating indices         */

      while (getline(inputFile, line)) {
            istringstream ss(line);
            ss >> month >> day >> temp >> rhum >> wind >> prcp;
            FFMCcalc(temp,rhum,wind,prcp,ffmc0,ffmc);
            DMCcalc(temp,rhum,prcp,dmc0,month,dmc);
            DCcalc(temp,prcp,dc0,month,dc);
            ISIcalc(ffmc,wind,isi);
            BUIcalc(dmc,dc,bui);
            FWIcalc(isi,bui,fwi);
            ffmc0 = ffmc;
            dmc0 = dmc;
            dc0 =   dc;
            outputFile<<ffmc<<' '<<dmc<<' '<<dc<<' '<<isi<<' '<<bui<<' '<<fwi<<endl;
      }
      inputFile.close();
}     outputFile.close()
```

**Table 4. Source code for Canadian Forest Fire Weather Index System calculations in the Python programming language**

```
import math
    """ Define Class FWI Class first"""

class FWICLASS:
      def __init__(self,temp,rhum,wind,prcp):
             self.h = rhum
             self.t = temp
             self.w = wind
             self.p = prcp

      def FFMCcalc(self,ffmc0):
             mo = (147.2*(101.0 - ffmc0))/(59.5 + ffmc0)                    #*Eq. 1*#
             if (self.p > 0.5):
                    rf = self.p - 0.5                                       #*Eq. 2*#
                    if(mo > 150.0):
                          mo = (mo+42.5*rf*math.exp(-100.0/(251.0-mo))*
                          (1.0 - math.exp(-6.93/rf))) \
                           + (.0015*(mo - 150.0)**2)*math.sqrt(rf)          #*Eq. 3b*#
             elif mo <= 150.0:
                       mo = mo+42.5*rf*math.exp(-100.0/(251.0-mo))*(1.0 - math.exp(-
                       6.93/rf))                                            #*Eq. 3a*#
             if(mo > 250.0):
                    mo = 250.0
      ed = .942*(self.h**.679) + (11.0*math.exp((self.h-100.0)/10.0))+0.18*(21.1-
self.t) \
                   *(1.0 - 1.0/math.exp(.1150 * self.h))                    #*Eq. 4*#

      if(mo < ed):
            ew = .618*(self.h**.753) + (10.0*math.exp((self.h-100.0)/10.0))
            \+ .18*(21.1-self.t)*(1.0 - 1.0/math.exp(.115 * self.h))        #*Eq. 5*#
            if(mo <= ew):
                   kl = .424*(1.0-((100.0-
self.h)/100.0)**1.7)+(.0694*math.sqrt(self.w)) \
                                 *(1.0 - ((100.0 - self.h)/100.0)**8)       #*Eq. 7a*#
               kw = kl * (.581 * math.exp(.0365 * self.t))                  #*Eq. 7b*#
               m = ew - (ew - mo)/10.0**kw                                  #*Eq. 9*#
            elif mo > ew:
                  m = mo
      elif(mo == ed):
            m = mo
      elif mo > ed:
            kl =.424*(1.0-(self.h/100.0)**1.7)+(.0694*math.sqrt(self.w))*    \
                                (1.0-(self.h/100.0)**8)                    #*Eq. 6a*#
            kw = kl * (.581*math.exp(.0365*self.t))                        #*Eq. 6b*#
            m  = ed + (mo-ed)/10.0 ** kw                                    #*Eq. 8*#

      ffmc = (59.5 * (250.0 -m)) / (147.2 + m)                              #*Eq. 10*#
      if (ffmc  > 101.0):
            ffmc = 101.0
      if (ffmc  <= 0.0):
            ffmc = 0.0
      return ffmc

def DMCcalc(self,dmc0,mth):
      el = [6.5,7.5,9.0,12.8,13.9,13.9,12.4,10.9,9.4,8.0,7.0,6.0]

      t = self.t
      if (t < -1.1):
            t = -1.1
      rk = 1.894*(t+1.1) * (100.0-self.h) * (el[mth-1]*0.0001)    #*Eqs. 16 and 17*#
```

**Table 4. Continued**

```
      if self.p > 1.5:
            ra= self.p
            rw = 0.92*ra - 1.27                                             #*Eq. 11*#
            wmi = 20.0 + 280.0/math.exp(0.023*dmc0)                         #*Eq. 12*#
            if dmc0 <= 33.0:
                  b = 100.0 /(0.5 + 0.3*dmc0)                              #*Eq. 13a*#
            elif dmc0 > 33.0:
                  if dmc0 <= 65.0:
                        b = 14.0 - 1.3*math.log(dmc0)                      #*Eq. 13b*#
                  elif dmc0 > 65.0:
                        b = 6.2 * math.log(dmc0) - 17.2                    #*Eq. 13c*#
            wmr = wmi + (1000*rw) / (48.77+b*rw)                            #*Eq. 14*#
            pr = 43.43 * (5.6348 - math.log(wmr-20.0))                      #*Eq. 15*#
      elif self.p <= 1.5:
            pr = dmc0
      if (pr<0.0):
            pr = 0.0
      dmc = pr + rk
      if(dmc<= 1.0):
            dmc = 1.0
      return dmc

def DCcalc(self,dc0,mth):
      fl = [-1.6, -1.6, -1.6, 0.9, 3.8, 5.8, 6.4, 5.0, 2.4, 0.4, -1.6, -1.6]
      t = self.t

      if(t < -2.8):
            t = -2.8
      pe = (0.36*(t+2.8) + fl[mth-1] )/2                                    #*Eq. 22*#
      if pe < =0.0:
            pe = 0.0

      if (self.p > 2.8):
            ra = self.p
            rw = 0.83*ra - 1.27                                             #*Eq. 18*#
            smi = 800.0 * math.exp(-dc0/400.0)                              #*Eq. 19*#
            dr = dc0 - 400.0*math.log(  1.0+((3.937*rw)/smi) )       #*Eqs. 20 and 21*#
            if (dr > 0.0):
                   dc = dr + pe
      elif self.p <= 2.8:
            dc = dc0 + pe
      return dc

def ISIcalc(self,ffmc):
      mo = 147.2*(101.0-ffmc) / (59.5+ffmc)                                  #*Eq. 1*#
      ff = 19.115*math.exp(mo*-0.1386) * (1.0+(mo**5.31)/49300000.0)        #*Eq. 25*#
      isi = ff * math.exp(0.05039*self.w)                                   #*Eq. 26*#
      return isi

def BUIcalc(self,dmc,dc):
      if dmc <= 0.4*dc:
            bui = (0.8*dc*dmc) / (dmc+0.4*dc)                              #*Eq. 27a*#
      else:
            bui = dmc-(1.0-0.8*dc/(dmc+0.4*dc))*(0.92+(0.0114*dmc)**1.7)   #*Eq. 27b*#
      if bui <0.0:
            bui = 0.0
      return bui
```

**Table 4. Concluded**

```
def FWIcalc(self,isi,bui):
      if bui <= 80.0:
            bb = 0.1 * isi * (0.626*bui**0.809 + 2.0)                      #*Eq. 28a*#
      else:
            bb = 0.1*isi*(1000.0/(25. + 108.64/math.exp(0.023*bui)))       #*Eq. 28b*#
      if(bb <= 1.0):
            fwi = bb                                                       #*Eq. 30b*#
      else:
            fwi = math.exp(2.72 *  (0.434*math.log(bb))**0.647)            #*Eq. 30a*#

   return fwi

    """End of class FWI Class"""

def main():
      ffmc0 = 85.0
      dmc0 = 6.0
      dc0 = 15.0

      infile = open('data.txt','r')
      outfile = open('fwioutput.txt','w')

      try:
            for line in infile:
                   mth,day,temp,rhum,wind,prcp=[float(field) for field in
                                          line.strip().lstrip('[').rstrip(']').split()]
                   if rhum>100.0:
                          rhum = 100.0

                   mth = int(mth)

                   fwisystem = FWICLASS(temp,rhum,wind,prcp)

                   ffmc = fwisystem.FFMCcalc(ffmc0)
                   dmc = fwisystem.DMCcalc(dmc0,mth)
                   dc = fwisystem.DCcalc(dc0,mth)
                   isi = fwisystem.ISIcalc(ffmc)
                   bui = fwisystem.BUIcalc(dmc,dc)
                   fwi = fwisystem.FWIcalc(isi,bui)

                   ffmc0 = ffmc
                   dmc0 = dmc
                   dc0 = dc
                   outfile.write("%s %s %s %s %s
                   %s\n"%(str(ffmc),str(dmc),str(dc),str(isi),str(bui),str(fwi)))
      finally:
                   infile.close()
                   outfile.close()
main()
```

**Table 5. Source code for Canadian Forest Fire Weather Index System calculations in the Java programming language**

```
import java.io.*;
import java.util.*;

public class fwi{

// Subroutine for calculating FFMC  //
      public static double FFMCcalc(double T,double H,double W,double ro,double Fo) {
            double mo,rf,mr,Ed,Ew,ko,kd,kl,kw,m,F;
            mo = 147.2*(101.0-Fo)/(59.5+Fo);                                   /*Eq. 1*/
            if(ro > 0.5) {
                    rf = ro - 0.5;                                             /*Eq. 2*/
                    if(mo <= 150.0) {
                           mr = mo+42.5*rf*(Math.exp(-100.0/(251.0-mo)))*
                                          (1.0-Math.exp(-6.93/rf));           /*Eq. 3a*/
            }
            else {
                  mr = mo+42.5*rf*(Math.exp(-100.0/(251.0-mo)))*(1.0-Math.exp(-
      6.93/rf)) +
                         .0015*Math.pow(mo-150.0,2.0)*Math.pow(rf,.5);       /*Eq. 3b */
            }
            if(mr > 250.0) {
                  mr = 250.0;
            }
            mo = mr;
      }

      Ed = 0.942*Math.pow(H,.679)+11.0*Math.exp((H-100.0)/10.0)+
                    .18*(21.1-T)*(1.0-Math.exp(-.115*H));                      /*Eq. 4*/

      if(mo > Ed) {
            ko = 0.424*(1.0-Math.pow(H/100.0,1.7))+0.0694*Math.pow(W,.5)*
                         (1.0-Math.pow(H/100.0,8));                           /*Eq. 6a*/
            kd = ko*.581*Math.exp(0.0365*T);                                  /*Eq. 6b*/
            m = Ed+(mo-Ed)*Math.pow(10.0,-kd);                                /*Eq. 8 */
      } else {
            Ew = 0.618*Math.pow(H,.753)+ 10.0*Math.exp((H-100.0)/10.0)+
                          .18*(21.1-T)*(1.0-Math.exp(-.115*H));               /*Eq. 5*/
            if(mo < Ew) {
                   kl = 0.424*(1.0-Math.pow((100.0-H)/100.0,1.7)) +
                   0.0694*Math.pow(W,.5)*(1.0-Math.pow((100.0-H)/100.0,8));  /*Eq. 7a*/
                   kw = kl*.581*Math.exp(0.0365*T);                          /*Eq. 7b */
                   m = Ew - (Ew - mo)*Math.pow(10.0,-kw);                     /*Eq. 9*/
            }
            else m = mo;
      }
      F = 59.5*(250.0-m)/(147.2+m);                                           /*Eq. 10*/
      if (F>101.0) F = 101.0;
      return F;
}
```

**Table 5. Continued**

```
// Subrountine to calculate DMC       //
      public static double DMCcalc(double T, double H, double ro, double Po, int I) {
            double re,Mo,Mr,K,b,P,Pr;
            double Le[] = {6.5,7.5,9.,12.8,13.9,13.9,12.4,10.9,9.4,8.,7.,6.};

            if(T >= -1.1) {
                  K = 1.894*(T +1.1)*(100.0-H)*Le[I-1]*0.0001;              /*Eq. 16*/
            }
            else {
                  K = 0.0;                                                   /*Eq. 17*/
            }

            if(ro <= 1.5) {
                  Pr = Po;
            }
            else {
                  re = 0.92*ro-1.27;                                         /*Eq. 11*/
                  Mo = 20.0 + 280.0/Math.exp(0.023*Po);                      /*Eq. 12*/
                  if(Po <= 33.0) {
                         b = 100.0/(.50+.30*Po);                            /*Eq. 13a*/
                  }
                  else {
                         if(Po <= 65.0) {
                                 b = 14.0 - 1.3*Math.log(Po);               /*Eq. 13b*/
                         }
                         else {
                                 b = 6.2*Math.log(Po) - 17.2;
                         }
                  }                                                         /*Eq. 13c*/
                  Mr = Mo +1000.0*re/(48.77+b*re);                           /*Eq. 14*/
                  Pr = 43.43*(5.6348 -Math.log(Mr - 20.0));                  /*Eq. 15*/
            }
            If (Pr < 0.0) {
                   Pr = 0.0;
            }
            P = Pr + K;
            if(P<= 0.0) {
                   P = 0.0;
            }
            return P;
      }
// Subrountine to calculateDC       //
      public static double DCcalc(double T, double ro, double Do, int I) {
      double rd,Qo,Qr,V,D,Dr;
      double Lf[] = {-1.6,-1.6,-1.6,.9,3.8,5.8,6.4,5.,2.4,.4,-1.6,-1.6};

      if(ro > 2.8) {
            rd = 0.83*(ro) - 1.27;                                           /*Eq. 18*/
            Qo = 800.0*Math.exp(-Do/400.0);                                  /*Eq. 19*/
            Qr = Qo +3.937*rd;                                               /*Eq. 20*/
            Dr = 400.0*Math.log(800.0/Qr);                                   /*Eq. 21*/
            if(Dr > 0.0) Do = Dr;
            else Do = 0.0;
      }
      if(T > -2.8) V = 0.36*(T+2.8)+Lf[I-1];                                 /*Eq. 22*/
      else V = Lf[I-1];
      if (V < 0.) V =.0;
      D = Do + 0.5*V;                                                        /*Eq. 23*/
      return D;
      }
```

**Table 5. Continued**

```
// Subrountine to calculate isi       //
      public static double ISIcalc(double F, double W) {
            double fW,m,fF,R;
            fW = Math.exp(0.05039*W);                                     /*Eq. 24*/
            m = 147.2*(101.0-F)/(59.5+F);                                  /*Eq. 1*/
            fF = 91.9*Math.exp(-.1386*m)*(1.+Math.pow(m,5.31)/4.93E7);    /*Eq. 25*/
            R = 0.208*fW*fF;                                              /*Eq. 26*/
            return R;
      }

// Subrountine to calculate bui       //
      public static double BUIcalc(double P, double D) {
            double U;
            if(P <= .4*D) U = 0.8*P*D/(P+.4*D);                          /*Eq. 27a*/
            else U = P - (1.0-.8*D/(P+.4*D))*(.92+Math.pow(.0114*P,1.7));  /*Eq. 27b*/
            if (U < 0.0) U = 0.0;
            return U;
      }

// Subrountine to calculate fwi       //
      public static double FWIcalc(double R, double U) {
            double fD,B,S;
            if(U <= 80.0) fD = .626*Math.pow(U,.809)+2.0;                /*Eq. 28a*/
            else fD = 1000.0/(25.0+108.64*Math.exp(-.023*U));            /*Eq. 28b*/
            B = .1*R*fD;                                                 /*Eq. 29 */
            if(B > 1.0) S = Math.exp(2.72*Math.pow(.434*Math.log(B),.647));  /*Eq. 30a*/
            else S = B;                                                  /*Eq. 30b*/
            return S;
      }

//   Main Program Here                //
      public static void main(String [] args) throws IOException {

            File inFile = new File("data.txt");
            Scanner x = new Scanner (inFile);

            File outFile = new File("fwioutput.txt");
            FileWriter  fWriter = new FileWriter(outFile);
            PrintWriter pWriter = new PrintWriter(fWriter);

            double ffmc0 = 85.0;
            double dmc0 = 6.0;
            double dc0  = 15.0

            while (x.hasNext()) {
                   String a = x.next();
                   String b = x.next();
                   String c = x.next();
                   String d = x.next();
                   String e = x.next();
                   String f = x.next();

                   float  mth  = (Float.valueOf(a)).floatValue();
                   double temp = (Double.valueOf(c)).doubleValue();
                   double rhum = (Double.valueOf(d)).doubleValue();
                   double wind = (Double.valueOf(e)).doubleValue();
                   double prcp = (Double.valueOf(f)).doubleValue();
                   int    mnth = Math.round(mth);
                   if (rhum > 100.0) rhum = 100.0;
```

**Table 5. Concluded**

```
                    double ffmc = FFMCcalc(temp,rhum,wind,prcp,ffmc0);
                    double dmc  = DMCcalc(temp,rhum,prcp,dmc0,mnth);
                    double dc   = DCcalc(temp,prcp,dc0,mnth);
                    double isi  = ISIcalc(ffmc,wind);
                    double bui  = BUIcalc(dmc,dc);
                    double fwi  = FWIcalc(isi,bui);

                    ffmc0 = ffmc;
                    dmc0  = dmc;
                    dc0   = dc;
                    pWriter.printf("%s %s %s %s %s %s\n",ffmc,dmc,dc,isi,bui,fwi);
             }
             pWriter.close();
             x.close();
      }
}
```

**Table 6. Source code for Canadian Forest Fire Weather Index System calculations in the SAS/IML programming language**

```
proc iml;

/******* Subroutine for calculating FFMC ********/
start FFMCcalc(T,H,W,ro,Fo,ffmc);
      mo = (147.2*(101.0 - Fo))/(59.5 + Fo);   /*Equation1 in van Wagner and Pickett 1985*/
      if (ro > 0.5) then do;
             rf = ro - 0.5;
             if(mo <= 150.0) then mr = mo+42.5*rf*exp(-100.0/(251.0-mo))*
(1.0 - exp(-6.93/rf)) ;                                                  /*Eq. 3a*/
             else mr = (mo+42.5*rf*exp(-100.0/(251.0-mo))*(1.0 - exp(-6.93/rf)))
+(.0015*(mo - 150.0)**2)*sqrt(rf);                                       /*Eq. 3b*/

             if (mr > 250.0) then mr = 250.0;
             mo = mr;
      end;

      ed =.942*(H**.679)+(11.0*exp((H-100.0)/10.0))+0.18*(21.1-T)*(1.0-1.0/exp
(.1150 * H));                                                             /*Eq. 4*/

      if(mo > ed) then do;
             ko = .424*(1.0-(H/100.0)**1.7)+(.0694*sqrt(W))*(1.0- (H/100.0)**8); /*Eq. 6a*/
             kd = ko * (.581*exp(.0365*T));                                      /*Eq. 6b*/
             m = ed + (mo-ed)/10.0 ** kd ;                                        /*Eq. 8*/
      end;
      else do;
             ew = .618*(H**.753) + (10.0*exp((H-100.0)/10.0)) + .18*(21.1-T) *
             (1.0 - 1.0/exp(.115 * H));                                           /*Eq. 5*/

      if(mo < ew) then do;
                  kl = .424*(1.0-((100.0 - H)/100.0)**1.7) + (.0694*sqrt(W)) *
(1.0 - ((100.0 - H)/100.0)**8);                                          /*Eq. 7a*/
                   kw = kl * (.581 * exp(.0365 * T));                    /*Eq. 7b*/
                   m = ew - (ew - mo)/10.0**kw;                           /*Eq. 9*/
             end;
             else m = mo;
      end;

      ffmc = (59.5 * (250.0 -m)) / (147.2 + m);                          /*Eq. 10*/
      if (ffmc  > 101.0)  then ffmc = 101.0   ;
      if (ffmc  <=  0.0)  then ffmc =   0.0   ;
finish;
```

**Table 6. Continued**

```
/* Subroutine for calculating DMC                          */

start DMCcalc(T, H, ro, Po, I, dmc);
      el = {6.5 7.5 9.0 12.8 13.9 13.9 12.4 10.9 9.4 8.0 7.0 6.0};

      if(ro <= 1.5) then pr = Po;
      else do;
             re = 0.92*ro - 1.27;                                        /*Eq. 11*/
             mo = 20.0 + 280.0/exp(0.023*Po);                            /*Eq. 12*/
             if (Po <= 33.0) then b = 100.0 /(0.5 + 0.3*Po);            /*Eq. 13a*/
             else do;=
                    b =  6.2 * log(Po) - 17.2;                          /*Eq. 13c*/
                    if (Po-65.0 <= 0.0) then b = 14.0 - 1.3*log(Po);    /*Eq. 13b*/
             end;
             mr = mo + (1000.0*re) / (48.77+b*re);                       /*Eq. 14*/
             pr = 43.43 * (5.6348 - log(mr-20.0));                       /*Eq. 15*/
      end;

      if (T >= -1.1) then k = 1.894*(T+1.1) * (100.0-H) * (el[I]*0.0001);        /*Eq. 16*/
      else k = 0.0;                                                      /*Eq. 17*/

      if (pr <  0.0) then pr = 0.0;
      P = pr + k;
      If (P <= 0.0) then P = 0.0;
      dmc = P;
finish;

/* Subroutine for calculating DC                              */
start DCcalc (T, ro, D0, I, dc);

      fl = {-1.6 -1.6 -1.6 0.9 3.8 5.8 6.4 5.0 2.4 0.4 -1.6 -1.6};

      if (ro > 2.8) then do;
             rd = 0.83*ro - 1.27;                                        /*Eq. 18*/
             Qo = 800.0 * exp(-D0/400.0);                                /*Eq. 19*/
             Qr = Qo + 3.937*rd;                                         /*Eq. 20*/
             Dr = 400.0*log(800.0/Qr);                                   /*Eq. 21*/
             If (dr > 0.0) then D0 = Dr;
             else D0 = 0.0;
      end;

      if (T < -2.8) then v = fl[I];
      else v = (0.36*(T+2.8) + fl[I]);                                   /*Eq. 22*/

      if (v <= 0.0) then v = 0.0;

      dc = D0 + 0.5*V;                                                   /*Eq. 23*/
finish;

/* Subroutine for calculating ISI                            */
start ISIcalc(F,W, isi);
      mo =  (147.2*(101.0-f)) / (59.5+f);                                 /*Eq. 1*/
      fw =  exp(0.05039*w);                                              /*Eq. 24*/
      ff =  91.9*exp(-0.1386*mo) * (1.0+(mo**5.31)/4.93e7);              /*Eq. 25*/
      isi =  0.208 * fw * ff;                                            /*Eq. 26*/
finish;

/* Subroutine for calculating BUI                      */
start BUIcalc(P,D,bui);
      if (P <= 0.4*D) then BUI = (0.8*P*D) / (P+0.4*D);                 /*Eq. 27a*/
      else bui = P-(1.0-0.8*D/(P+0.4*D))*(0.92+(0.0114*P)**1.7);        /*Eq. 27b*/
      if (bui<0.0) then bui = 0.0;
finish;
```

**Table 6. Concluded**

```
/* Subroutine for calculating FWI                  */
start FWIcalc(R,U,fwi);
      if (U<= 80.0) then fD = 0.626*U**0.809 + 2.0;                     /*Eq. 28a*/
      else fD = 1000.0/(25.0+108.64*exp(-0.023*U));                      /*Eq. 28b*/
      B = 0.1*R*fD;                                                       /*Eq. 29*/
      if(B>1.0) then fwi = exp(2.72*(0.434*log(B))**0.647);             /*Eq. 30a*/
      else fwi = B;                                                      /*Eq. 30b*/
finish;

/******** Main program starts here ********************************/

filename indata 'P:\data.txt';
filename outdata 'P:\fwioutput.txt';
infile indata;
file outdata;

ffmc0 = 85.0;
dmc0 = 6.0;
dc0 = 15.0;

do data;
      input jm jd temp rhum wind prcp;
      if rhum>100.0 then rhum =100.0;

      call FFMCcalc(temp,rhum,wind,prcp,ffmc0, ffmc);
      call DMCcalc(temp, rhum, prcp, dmc0, jm, dmc);
      call DCcalc (temp, prcp, dc0, jm, dc);
      call ISIcalc(ffmc,wind,isi);
      call BUIcalc(dmc,dc,bui);
      call FWIcalc(isi,bui,fwi);
      ffmc0 = ffmc;
      dmc0 = dmc;
      dc0   = dc;

      put jm 3.0 jd  3.0 ffmc 9.1 dmc 9.1 dc 9.1 isi 9.1 bui 9.1 fwi 9.1;
end;
closefile indata;
closefile outdata;
```

**Table 7. Sample of input data and result output**

| Input data | | | | | | Result output | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Month | Day | Temp. | RH | Wind | Rain | FFMC | DMC | DC | ISI | BUI | FWI |
| 4 | 13 | 17.0 | 42.0 | 25.0 | 0.0 | 87.7 | 8.5 | 19.0 | 10.9 | 8.5 | 10.1 |
| 4 | 14 | 20.0 | 21.0 | 25.0 | 2.4 | 86.2 | 10.4 | 23.6 | 8.8 | 10.4 | 9.3 |
| 4 | 15 | 8.5 | 40.0 | 17.0 | 0.0 | 87.0 | 11.8 | 26.1 | 6.5 | 11.7 | 7.6 |
| 4 | 16 | 6.5 | 25.0 | 6.0 | 0.0 | 88.8 | 13.2 | 28.2 | 4.9 | 13.1 | 6.2 |
| 4 | 17 | 13.0 | 34.0 | 24.0 | 0.0 | 89.1 | 15.4 | 31.5 | 12.6 | 15.3 | 14.8 |
| 4 | 18 | 6.0 | 40.0 | 22.0 | 0.4 | 88.7 | 16.5 | 33.5 | 10.7 | 16.4 | 13.5 |
| 4 | 19 | 5.5 | 52.0 | 6.0 | 0.0 | 87.4 | 17.2 | 35.4 | 4.0 | 17.1 | 5.9 |
| 4 | 20 | 8.5 | 46.0 | 16.0 | 0.0 | 87.4 | 18.5 | 37.9 | 6.6 | 18.4 | 9.7 |
| 4 | 21 | 9.5 | 54.0 | 20.0 | 0.0 | 86.8 | 19.7 | 40.6 | 7.4 | 19.6 | 11.0 |
| 4 | 22 | 7.0 | 93.0 | 14.0 | 9.0 | 29.9 | 10.1 | 29.5 | 0.0 | 10.9 | 0.0 |
| 4 | 23 | 6.5 | 71.0 | 17.0 | 1.0 | 49.4 | 10.7 | 31.6 | 0.4 | 11.6 | 0.2 |
| 4 | 24 | 6.0 | 59.0 | 17.0 | 0.0 | 67.3 | 11.4 | 33.7 | 1.3 | 12.3 | 0.9 |
| 4 | 25 | 13.0 | 52.0 | 4.0 | 0.0 | 77.8 | 13.0 | 37.0 | 1.1 | 13.9 | 0.8 |
| 4 | 26 | 15.5 | 40.0 | 11.0 | 0.0 | 85.5 | 15.4 | 40.7 | 3.9 | 15.9 | 5.5 |
| 4 | 27 | 23.0 | 25.0 | 9.0 | 0.0 | 91.5 | 19.8 | 45.8 | 8.4 | 19.8 | 12.2 |
| 4 | 28 | 19.0 | 46.0 | 16.0 | 0.0 | 89.9 | 22.5 | 50.2 | 9.5 | 22.4 | 14.3 |
| 4 | 29 | 18.0 | 41.0 | 20.0 | 0.0 | 90.0 | 25.2 | 54.4 | 11.7 | 25.1 | 17.7 |
| 4 | 30 | 14.5 | 51.0 | 16.0 | 0.0 | 88.4 | 27.0 | 57.9 | 7.7 | 27.0 | 13.3 |
| 5 | 1 | 14.5 | 69.0 | 11.0 | 0.0 | 85.7 | 28.3 | 63.0 | 4.0 | 28.2 | 8.0 |
| 5 | 2 | 15.5 | 42.0 | 8.0 | 0.0 | 87.4 | 30.8 | 68.2 | 4.4 | 30.8 | 9.1 |
| 5 | 3 | 21.0 | 37.0 | 8.0 | 0.0 | 89.4 | 34.5 | 74.3 | 5.9 | 34.4 | 12.3 |
| 5 | 4 | 23.0 | 32.0 | 16.0 | 0.0 | 91.0 | 38.8 | 80.9 | 11.1 | 38.7 | 21.0 |
| 5 | 5 | 23.0 | 32.0 | 14.0 | 0.0 | 91.2 | 43.1 | 87.4 | 10.3 | 43.0 | 21.1 |
| 5 | 6 | 27.0 | 33.0 | 12.0 | 0.0 | 91.7 | 48.1 | 94.7 | 9.9 | 47.9 | 21.7 |
| 5 | 7 | 28.0 | 17.0 | 27.0 | 0.0 | 95.2 | 54.5 | 102.1 | 34.5 | 54.3 | 52.6 |
| 5 | 8 | 23.5 | 54.0 | 20.0 | 0.0 | 89.7 | 57.4 | 108.8 | 11.3 | 57.2 | 25.9 |
| 5 | 9 | 16.0 | 50.0 | 22.0 | 12.2 | 62.2 | 29.9 | 91.8 | 1.4 | 33.0 | 3.0 |
| 5 | 10 | 11.5 | 58.0 | 20.0 | 0.0 | 76.7 | 31.3 | 96.3 | 2.3 | 34.5 | 5.4 |
| 5 | 11 | 16.0 | 54.0 | 16.0 | 0.0 | 83.5 | 33.4 | 101.6 | 3.8 | 36.7 | 8.9 |
| 5 | 12 | 21.5 | 37.0 | 9.0 | 0.0 | 88.7 | 37.1 | 107.8 | 5.6 | 39.9 | 12.8 |
| 5 | 13 | 14.0 | 61.0 | 22.0 | 0.2 | 86.7 | 38.7 | 112.8 | 8.1 | 41.6 | 17.3 |
| 5 | 14 | 15.0 | 30.0 | 27.0 | 0.0 | 89.6 | 41.7 | 117.9 | 15.9 | 44.2 | 28.8 |
| 5 | 15 | 20.0 | 23.0 | 11.0 | 0.0 | 92.1 | 45.9 | 123.9 | 10.1 | 47.7 | 21.9 |
| 5 | 16 | 14.0 | 95.0 | 3.0 | 16.4 | 21.3 | 20.1 | 97.0 | 0.0 | 26.5 | 0.0 |
| 5 | 17 | 20.0 | 53.0 | 4.0 | 2.8 | 51.0 | 18.3 | 103.0 | 0.2 | 25.3 | 0.2 |
| 5 | 18 | 19.5 | 30.0 | 16.0 | 0.0 | 82.3 | 22.1 | 108.9 | 3.3 | 29.3 | 6.8 |
| 5 | 19 | 25.5 | 51.0 | 20.0 | 6.0 | 75.4 | 16.4 | 106.4 | 2.1 | 23.7 | 3.8 |

**Table 7. Concluded**

| Input data | | | | | | Result output | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Month | Day | Temp. | RH | Wind | Rain | FFMC | DMC | DC | ISI | BUI | FWI |
| 5 | 20 | 10.0 | 38.0 | 24.0 | 0.0 | 84.3 | 18.2 | 110.6 | 6.4 | 25.8 | 11.3 |
| 5 | 21 | 19.0 | 27.0 | 16.0 | 0.0 | 90.3 | 22.1 | 116.4 | 10.0 | 29.9 | 17.2 |
| 5 | 22 | 26.0 | 46.0 | 11.0 | 4.2 | 77.6 | 18.7 | 117.7 | 1.6 | 26.8 | 2.9 |
| 5 | 23 | 30.0 | 38.0 | 22.0 | 0.0 | 90.2 | 23.8 | 125.5 | 13.4 | 32.3 | 22.0 |
| 5 | 24 | 25.5 | 67.0 | 19.0 | 12.6 | 65.3 | 13.1 | 108.5 | 1.4 | 20.2 | 1.9 |
| 5 | 25 | 12.0 | 53.0 | 28.0 | 11.8 | 55.4 | 7.7 | 91.6 | 1.2 | 12.8 | 0.8 |
| 5 | 26 | 21.0 | 38.0 | 8.0 | 0.0 | 80.8 | 11.3 | 97.8 | 1.9 | 17.6 | 2.6 |
| 5 | 27 | 13.0 | 70.0 | 20.0 | 3.8 | 61.7 | 8.4 | 97.9 | 1.2 | 13.8 | 0.9 |
| 5 | 28 | 9.0 | 78.0 | 24.0 | 1.4 | 64.5 | 9.0 | 101.9 | 1.7 | 14.7 | 2.0 |
| 5 | 29 | 11.0 | 54.0 | 16.0 | 0.0 | 77.6 | 10.5 | 106.3 | 2.0 | 16.8 | 2.8 |
| 5 | 30 | 15.5 | 39.0 | 9.0 | 0.0 | 85.4 | 13.1 | 111.5 | 3.5 | 20.3 | 5.8 |
| 5 | 31 | 18.0 | 36.0 | 5.0 | 0.0 | 88.5 | 16.3 | 117.1 | 4.4 | 24.2 | 7.9 |

BUI = Buildup Index.
FWI = Fire Weather Index.
Temp. = temperature.
RH = Relative Humidity.
FFMC = Fine Fuel Moisture Code.
DMC = Duff Moisture Code.
DC = Drought Code.
ISI = Initial Spread Index.

## DISCUSSION AND CONCLUSIONS

The FWI System has proven successful as a forest fire danger rating system in Canada and has been adopted by many forest fire research and management communities around the world since its release in the 1980s. The purpose of this report was to improve the original source code by updating it to the standards set by commonly used programming languages and thus to make the coding more useful for forest fire research communities across the world. The digital files of all source codes are available from the authors upon request. In addition to the updates described here, the FWI System has also been coded in the R language (CFFDRS, available online at https://r-forge.r-project.org/projects/cffdrs/), and Wildland Fire Information Systems Group of the Canadian Forest Service, in conjunction with other provincial and territorial agencies, has been developing a web-based operational library of FWI and fire behavior prediction functions called REDapp in Java (see www.REDapp.org).

This report should prove useful to researchers in the scientific community who have been using the original source code. The new codes have been written in a modular programming style, and each module acts as an independent program unit. As such, the FWI calculation module can be easily included in larger programs of which the FWI calculation is only one element. We created these FWI calculation modules in several programming languages that are widely used in science and engineering computations, to provide more choice for users with various programming backgrounds. Over the years, many requests have been received by the Canadian Forest Service for such source code, and this report addresses these needs.

## ACKNOWLEDGMENTS

We are grateful to C. E. Van Wagner and T. L. Pickett of the Canadian Forest Service for their FORTRAN 77 source code; this current publication builds upon their original source code, without which this work could not have been completed. We also appreciate the time and effort invested by reviewers Mike Wotton of the Great Lakes Forestry Centre, Canadian Forest Service, Steve Taylor of the Pacific Forestry Centre, Canadian Forest Service, and Dan Thompson and Bruce Macnab of the Northern Forestry Centre, Canadian Forest Service. Finally, we thank Brenda Laishley of the Northern Forestry Center, Canadian Forest Service, for editing this report.

## LITERATURE CITED

Canadian Forestry Service. 1984. Tables for the Canadian Forest Fire Weather Index System. (4th ed.) Environ. Can., Can. For. Serv., Ottawa, ON. For. Tech. Rep. 25. Also available at http://cfs.nrcan.gc.ca/pubwarehouse/pdfs/31168.pdf.

Chapman, S. J. 2004. FORTRAN 90/95 for scientists and engineers. McGraw Hill Higher Education, Toronto, ON. Also available at http://www.mhhe.com/engcs/general/chapman/index.mhtml.

De Groot, W.J.; Field, R.D.; Brady, M.A.; Roswintiarti, O.; Mohamad, M. 2006. Development of the Indonesian and Malaysian fire danger rating system. Mitig. Adapt. Strateg. Glob. Chang. 12:165–180. doi: 10.1007/s11027-006-9043-8.

Engisch, R.L.; Walker, J.D. 1971. PDP-8L version of Simard's Fire Weather Index program. Can. For. Serv., Petawawa For. Exp. Stn., Chalk River, ON. Intern. Rep. PS-23. Also available at http://cfs.nrcan.gc.ca/publications?id=36197.

Field, R.D.; Wang, Y.; Roswintiarti, O.; Guswanto. 2004. A drought-based predictor of recent haze events in western Indonesia. Atmos. Environ. 38: 1869–1878. doi: 10.1016/j.atmosenv.2004.01.011.

Fogarty, L.G.; Pearce, H.G.; Catchpole, W.R.; Alexander, M.E. 1998. Adoption vs. adaptation: lessons from applying the Canadian Forest Fire Danger Rating System in New Zealand. Pages 1011–1028 in D.X. Viegas, ed.) Proceedings 3rd International Conference on Forest Fire Research and 14th Conference on Fire and Forest Meteorology, 16–20 November 1998. Luso, Portugal.. Associação para o Densenvolvimento de Aerodinâmica Industrial, Coimbra, Portugal, Vol. 1. Also available at https://cfs.nrcan.gc.ca/publications?id=18753.

Forestry Canada. 1992. Development and structure of the Canadian Forest Fire Behavior Prediction System. For. Can., Fire Danger Group, Ottawa, ON. Inf. Rep. ST-X-3. Also available at https:// http://cfs.nrcan.gc.ca/pubwarehouse/pdfs/10068.pdf .

Horel, J.D.; Ziel, R.; Galli, C.; Pechmann, J.J.; Dong, X. 2014. An evaluation of fire danger and behaviour indices in the Great Lakes Region calculated from station and gridded weather information. Int. J. Wildland Fire 23: 202–214. doi:10.1071/WF12186.

Jandt, R.; Allen, J.; Horschel, E. 2005. Forest floor moisture content and fire danger indices in Alaska. US Dep. Inter., Bur. Land Manag., Washington, DC. Alaska Tech. Rep. 54. BLM/AK/ST-05/009+9218+313.Also available at http://fire.ak.blm.gov/content/effects/techreport54.pdf

Kean, W.A. 1975. A PDP-8L program for calculating the Fire Weather Index. Can. For. Serv., Petwawa For. Exp. Stn., Chalk River, ON. Inf. Rep. PS-X-57. Also available at http://cfs.nrcan.gc.ca/publications?id=36176.

Simard, A.J. 1970. Computer program to calculate the Canadian Forest Fire Weather Index System. Can. For. Serv., For. Fire Res. Inst., Ottawa, ON. Intern. Rep. FF-12. Also available at http://cfs.nrcan.gc.ca/publications?id=36178.

Simpson, C.C.; Pearce, H.G.; Sturman, A.P; Zawar-Reza, P. 2014. Verification of WRF modelled fire weather in the 2009–10 New Zealand fire season. Int. J. Wildland Fire 23: 34–45. doi: 10.1071/WF12152.

Šturm, T.; Fernandes, P.M.; Sumrada, R. 2012. The Canadian fire weather index system and wildfire activity in the Karst forest management area, Slovenia. Eur. J/ For. Res. 131: 829–834. doi: 10.1007/s10342-011-0556-7.

Taylor, S.W.; Alexander, M.E. 2006. Science, technology, and human factors in fire danger rating: the Canadian experience. Int. J. Wildland Fire15: 121–135. doi: 10.1071/WF05021.

Tian, X.R.; Zhao, F.J.; Shu, L.F.; Wang, M.Y. 2014. Changes in forest fire danger for south-western China in the 21st century. Int. J. Wildland Fire 23: 185–195. doi:10.1071/WF13014.

Van Wagner, C.E. 1987. Development and structure of the Canadian Forest Fire Weather Index System. Can. For. Serv., Ottawa, ON. For. Tech. Rep. 35. Also available at http://cfs.nrcan.gc.ca/publications/download-pdf/19927.

Van Wagner, C.E.; Pickett, T.L. 1975. Equations and FORTRAN IV program for the 1976 metric version of the forest fire weather index. Can. For. Serv., Petawawa Natl. For. Inst., Chalk River, ON. Inf. Rep. PS-X-58. Also available at https://cfs.nrcan.gc.ca/publications?id=23602.

Van Wagner, C.E.; Pickett, T.L. 1985. Equations and FORTRAN program for the Canadian Forest Fire Weather Index System. Can. For. Serv., Petawawa Natl. For. Inst., Chalk River, ON. For. Tech. Rep. 33. Also available at https://cfs.nrcan.gc.ca/publications?id=19973.

Wright, J.G. 1933. Forest-fire hazard tables for mixed red and white pine forest eastern Ontario and western Quebec regions. Can. Dep. Inter., For. Serv., Ottawa, ON. For. Fire Hazard Pap. 3. Also available at https://cfs.nrcan.gc.ca/publications?id=36177.

Wright, J.G. 1937. Preliminary improved fire hazard index tables for pine forests at Petawawa Forest Experiment Station. Can. Dep. Mines Resour., Dom. For. Serv., Ottawa, ON. Also available at http://cfs.nrcan.gc.ca/publications?id=36217.

Canada