FORTRAN to C Workshop

Joe Lambert

Computer Science and Engineering Department

August 1994
CONTENTS

- Simple Programs in FORTRAN and C  1
- Control Structures in FORTRAN and C  16
- Subprograms in FORTRAN and C  35
- Array Structures in FORTRAN and C  58
- Files in FORTRAN and C  65
- Additional Data Types in FORTRAN and C  72
- C calls FORTRAN subroutines  93
CONTENTS

⇒ Simple Programs in FORTRAN and C .
   1
   ■ Hello World
   ■ Average Of Values
   ■ Arithmetic Operations
   ■ Intrinsic functions
   ■ Period of a Pendulum

■ Control Structures in FORTRAN and C
   16

■ Subprograms in FORTRAN and C .. 35

■ Array Structures in FORTRAN and C .
   58

■ Files in FORTRAN and C .......... 65

■ Additional Data Types in FORTRAN
   and C .............................. 72

■ C calls FORTRAN subroutines ..... 93
Simple Programs in FORTRAN and C

Hello World

FORTRAN

*************************************************************************
PROGRAM HELLO
* THIS PROGRAM PRINTS HELLO WORLD
*************************************************************************
PRINT*,’Hello World’
STOP
END

C

Joe Lambert
/* This program prints Hello World */
#include <stdio.h>
void main(void)
{
    printf("Hello World \n");
}

Joe Lambert
Average Of Values
PROGRAM COMPUT
* THIS PROGRAM COMPUTES THE AVERAGE OF
* A SET OF DATA VALUES. NOTE ERROR!

INTEGER COUNT
REAL X, SUM, AVERG

SUM = 0.0
COUNT = 0
READ*, X
1 IF(X.NE.0.0) THEN
   SUM=SUM+X
   COUNT = COUNT + 1
   READ*, X
GO TO 1
ENDIF
AVERG = SUM/REAL(COUNT)
PRINT 5, AVERG

5 FORMAT (1X,'THE AVERAGE IS ',F6.2)
STOP
END
C

/* This program computes the average of
 * a set of data values. Note error. */
#include <stdio.h>
void main(void)
{
    int count;
    float x, sum, average;
        sum = 0;
        count = 0;
        scanf("%f", &x);
    start: if ( x != 0.0 )
        sum = sum + x;
        count = count + 1;
        scanf("%f", &x);
        goto start;
    }
    average = sum/count;
    printf("The average is %6.2f ",average);
}
Better C

/* This program computes the average of
 * a set of data values */
#include <stdio.h>
void main(void)
{
  int count = 0;
  float x, average,
       sum = 0;
  do {
      scanf("%f", &x);
      sum = sum + x;
      count = count + 1;
  }while (x != 0.0);

  average = sum/count;
  printf("The average is %6.2f ",average);
}
Classical C

/* This program computes the average of
 * a set of data values */
#include <stdio.h>
void main(void)
{
    int count =0;
    float x,average,
        sum= 0;
    do {
        scanf("%f", &x);
        sum += x;
        count++ ;
    }while (x != 0.0);

    average = sum/count;
    printf("The average is %6.2f ",average);
}
Arithmetic Operations
PROGRAM COMPUT
* THIS PROGRAM GIVES EXAMPLES OF
* ARITHMETIC OPERATIONS

INTEGER I,J,K
DOUBLE PRECISION X, Y, Z

I=1
J=2
X=1.0
Y=2.0

Z=X*Y
K=I/J
Z=X**J
STOP
END
C

/* This program gives examples of
 * arithmetic operations */
#include <stdio.h>
#include <math.h>
void main(void)
{
    int k,i=1,j=2;
double x=1.0,y=2.0,z;

    z=x+y;
k=i/j;
z = pow(x,j);
    /* double pow( double u, double v) */
}
# Intrinsic functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Fortran</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sqrt{x}$</td>
<td>SQRT(X)</td>
<td>double sqrt(double x)</td>
</tr>
<tr>
<td>$</td>
<td>x</td>
<td>$</td>
</tr>
<tr>
<td>sin($x$)</td>
<td>SIN(X)</td>
<td>double sin(double x)</td>
</tr>
<tr>
<td>cos($x$)</td>
<td>COS(X)</td>
<td>double cos(double x)</td>
</tr>
<tr>
<td>tan($x$)</td>
<td>TAN(X)</td>
<td>double tan(double x)</td>
</tr>
<tr>
<td>$e^x$</td>
<td>EXP(X)</td>
<td>double exp(double x)</td>
</tr>
<tr>
<td>$\log_e(x)$</td>
<td>LOG(X)</td>
<td>double log(double x)</td>
</tr>
<tr>
<td>$\log_{10}(x)$</td>
<td>LOG10(X)</td>
<td>double log10(double x)</td>
</tr>
<tr>
<td>Modulus</td>
<td>MOD(I,J)</td>
<td>i%j</td>
</tr>
</tbody>
</table>
Period of a Pendulum

FORTRAN

Joe Lambert
PROGRAM PERIOD
* THIS PROGRAM CALCULATES THE PERIOD
* OF A PENDULUM

REAL L, ALPHA, P, TEMP
PARAMETER (PI=3.1415, G=980)

PRINT *, 'ENTER LENGTH OF PENDULUM IN CM'
READ *, L
PRINT *, 'ENTER ANGLE OF DISPLACEMENT'
PRINT *, IN RADIANS'
READ *, ALPHA

TEMP = SIN( ALPHA/2)
P=2*PI*SQRT(L/G)*(1+0.25*TEMP*TEMP)

PRINT *, P, ' IS THE PENDULUM PERIOD'
STOP
END
/* This program Calculates the period
 * of a pendulum*/
#include <stdio.h>
#include <math.h>
void main(void)
{
  double l, alpha, p, temp;
  constant double g = 9.80;
  constant double pi = 3.1415;

  printf("Enter the length of the pendulum");
  printf(" in cm:");
  scanf("%lf", &l);
  printf("\n Enter the angle of ");
  printf("displacement in radians:");
  scanf("%lf", &alpha);

  temp = sin(alpha/2);
  p = 2*pi*sqrt(l/g)*(1+0.25*temp*temp);

  printf("\n %f is the pendulum period", p);
}
CONTENTS

- Simple Programs in FORTRAN and C  1
- Control Structures in FORTRAN and C  16
- Subprograms in FORTRAN and C  35
- Array Structures in FORTRAN and C  58
- Files in FORTRAN and C  65
- Additional Data Types in FORTRAN and C  72
- C calls FORTRAN subroutines  93
 CONTENTS

✓  Simple Programs in FORTRAN and C  1
⇒  Control Structures in FORTRAN and C  16
   ■  Selection
   ■  Repetition
   ■  Relational Operators
   ■  Logical Operators
   ■  Iteration

■  Subprograms in FORTRAN and C  35

■  Array Structures in FORTRAN and C  58

■  Files in FORTRAN and C  65

■  Additional Data Types in FORTRAN and C  72

■  C calls FORTRAN subroutines  93
Control Structures in FORTRAN and C

Selection

FORTRAN

Joe Lambert
PROGRAM SLCT1
* THIS PROGRAM SHOWS IF
* STRUCTURE
REAL FRACT, DEN, NUM

READ *, FRACT, DEN, NUM

IF (DEN .NE. 0.0) THEN
  FRACT = NUM/DEN
  PRINT *, FRACT
ENDIF

PRINT *, 'END OF PROGRAM'
STOP
END
/* This program shows if structure*/
#include <stdio.h>
void main(void)
{
    float fract, den, num;

    scanf(" %f %f %f ", &fract, &den, &num);

    if (den != 0.0)
    {
        fract = num/den;
        printf("%f \n", fract);
    }

    printf("End of Program\n");
}
FORTRAN

************

PROGRAM SLCT2
*
THIS PROGRAM SHOWS IF THEN ELSE
*
STRUCTURE
************

REAL FRACT, DEN, NUM

READ *, FRACT, DEN, NUM

IF ( DEN .NE. 0.0 ) THEN
  FRACT = NUM / DEN
ELSE
  FRACT = -1
ENDIF

PRINT *, FRACT

STOP
END
C

/*/ This program shows if else structure*/
#include <stdio.h>
void main(void)
{
    float fract,den,num;

    scanf(" %f %f %f ",&fract,&den,&num);

    if (den != 0.0)
    {
        fract = num/den;
    }
    else
    {
        fract = -1;
    }
    printf("%f\n",fract);
}
PROGRAM SLCT3
* THIS PROGRAM SHOWS IF ELSEIF ELSE
* STRUCTURE
REAL FRACT, DEN, NUM

READ *, FRACT, DEN, NUM

IF ( DEN .LT. 0.0 ) THEN
   FRACT = NUM / DEN
ELSEIF ( DEN .GT. 0.0 ) THEN
   FRACT = NUM
ELSE
   FRACT = -1
ENDIF
PRINT *, FRACT
STOP
END
/ This program shows if elseif structure*/
#include <stdio.h>
void main(void)
{
    float fract, den, num;

    scanf(" %f %f %f ", &fract, &den, &num);

    if (den < 0.0)
        fract = num / den;
    else if (den > 0.0)
        fract = num;
    else
        fract = -1;
    printf("%f\n", fract);
}
Repetition
PROGRAM FLIGHT
* THIS PROGRAM SIMULATES Rocket FLIGHT

REAL TIME, HEIGHT
LOGICAL DONE

DONE = .FALSE.
TIME = 0.0
HEIGHT = 0

1 IF (.NOT. DONE) THEN
   HEIGHT = HEIGHT + 3.0 * TIME
   IF (HEIGHT .LT. 50) THEN
      TIME = TIME + .05
   ELSE
      TIME = TIME + 2
   ENDIF
   DONE = HEIGHT .GT. 100 .OR.
   + TIME .GT. 100
GO TO 1
ENDIF
END
/ This program simulates
* rocket flight. */
#include <stdio.h>
void main(void)
{
#define TRUE 1
#define FALSE 0

float time, height;
int done = FALSE;

time=0;
height=0;
while( !done)
{
    height += 3*time;

    if (height < 50)
        time+=.05;
    else
        time +=2;

    done = (height > 100) || (time > 100);
}
FORTRAN to C Workshop

Control Structures in FORTRAN and C

}`
## Relational Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Fortran</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>equal</td>
<td>.EQ.</td>
<td>==</td>
</tr>
<tr>
<td>not equal</td>
<td>.NE.</td>
<td>!=</td>
</tr>
<tr>
<td>less than</td>
<td>.LT.</td>
<td>&lt;</td>
</tr>
<tr>
<td>less than or equal</td>
<td>.LE.</td>
<td>&lt;=</td>
</tr>
<tr>
<td>greater than</td>
<td>.GT.</td>
<td>&gt;</td>
</tr>
<tr>
<td>greater than or equal</td>
<td>.GE.</td>
<td>&gt;=</td>
</tr>
</tbody>
</table>
Logical Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Fortran</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>and</td>
<td>.AND.</td>
<td>&amp;&amp;</td>
</tr>
<tr>
<td>or</td>
<td>.OR.</td>
<td></td>
</tr>
<tr>
<td>not</td>
<td>.NOT.</td>
<td>!</td>
</tr>
</tbody>
</table>
Iteration

FORTRAN

******************************************************************************
PROGRAM POLY
* THIS PROGRAM CALCULATES 3*T*T+4.5
******************************************************************************
INTEGER TIME
REAL POLY

*
DO 15 TIME = 1,10
   POLY = 3.0*REAL(TIME)**2+4.5
PRINT 10,TIME,POLY
10 FORMAT(1X,I2,5X,F6.2)
15 CONTINUE
END
/ This program calculates
* 3*t*t+4.5. */
#include <stdio.h>
void main(void)
{
    float poly;
    int time;

    for( time = 1; time <= 10; time++)
    {
        poly = 3.0*time*time +4.5;
        printf("%2d %6.2f\n",time,poly);
    }
}
FORTRAN

*******************************************************************************
PROGRAM NEST
* THIS PROGRAM EXHIBITS NESTED INTERATION
*******************************************************************************
INTEGER I,J
*
   DO 15 I = 1,10
   DO 10 J = 1,12
      PRINT*, I,J
  10 CONTINUE
  15 CONTINUE
END
/* This program exhibits
* nested for loops */
#include <stdio.h>
void main(void)
{
  int i,j;

  for( i = 1; i <= 10;i++)
  for( j = 1; j <= 12; j++)
    printf("%d %d \n",i,j);
}

Joe Lambert
CONTENTS

- Simple Programs in FORTRAN and C  1
- Control Structures in FORTRAN and C  16
- Subprograms in FORTRAN and C  .  35
- Array Structures in FORTRAN and C  .  58
- Files in FORTRAN and C  ............  65
- Additional Data Types in FORTRAN and C  .........................  72
- C calls FORTRAN subroutines .....  93
CONTENTS

√ ▪ Simple Programs in FORTRAN and C  1

√ ▪ Control Structures in FORTRAN and C  16

⇒ ▪ Subprograms in FORTRAN and C . 35
  ▪ Functions
  ▪ Subroutines and Procedures
  ▪ Call by Reference / Value
  ▪ Pointers, Addresses, Values

▪ Array Structures in FORTRAN and C . 58

▪ Files in FORTRAN and C .......... 65

▪ Additional Data Types in FORTRAN and C . 72

▪ C calls FORTRAN subroutines ..... 93
Subprograms in FORTRAN and C

Functions

FORTRAN

Joe Lambert
PROGRAM FUN

* THIS PROGRAM SHOWS FUNCTION STRUCTURE

*-----------------------------------------------------------------------

INTEGER I
REAL CENT, FAHREN, TEMP, NEW, OLD
CENT(TEMP)=(TEMP-32.0)*.5555556
FAHREN(TEMP)= 1.8*TEMP +32.0

OLD = 44.0
NEW = CENT(OLD)

OLD = -15.0
NEW = FAHREN(OLD)

PRINT *,’ END OF PROGRAM’
STOP
END
/* This program shows function structure*/

#include <stdio.h>
void main(void)
{
    int i;
    float cent(float x);
    float fahren(float x);
    float new, old;

    old = 44;
    new = cent(old);

    old = -16;
    new = fahren(old);
}

float cent(float temp)
{
    return (temp-32)*5/9;
}

float fahren(float temp)
{
    return (9*temp/5 + 32);
}
Subroutines and Procedures
PROGRAM ANGLE
* THIS PROGRAM SHOWS SUBROUTINE CALLS

INTEGER DEGS, MINUTS, SECONS, RESP

10 CONTINUE
   PRINT *, ’ENTER DEGREES, MINUTES, SECS’
   READ *, DEGS, MINUTS, SECONS
   CALL PRNDEG(DEGS, MINUTS, SECONS)
   PRINT *, ’MORE (0 = NO, 1 = YES)’
   READ *, RESP
   IF (RESP .NE. 0) GO TO 10
   PRINT *, ’END OF PROGRAM’
   STOP
END

SUBROUTINE PRNDEG(DEG, MIN, SEC)

INTEGER DEG, MIN, SEC
REAL X
X = DEG + REAL(MIN)/60.0 + REAL(SEC)/3600.0
PRINT *, X
END
/* This program shows function calls*/

#include <stdio.h>
void main(void)
{
  int degs, minuts, secons, resp;
  void prndeg( int deg, int min, int sec);

do{
    printf("enter degrees, minutes, secs:");
    scanf("%d %d %d", &degs, &minuts, &secons);
    prndeg(degs, minuts, secons);
    printf("more ? (0 = no, 1 =yes): ");
    scanf(" %d", &resp);
  }while (resp != 0);
  printf("End of Program\n");
}

void prndeg( int deg, int min, int sec)
{
  float x;
  x = deg + min/60.0 + sec/ 3600.0;
  printf("%f \n", x);
}

Joe Lambert
Call by Reference / Value
PROGRAM POLAR
** CALL BY REFERENCE AND VALUE

REAL RCRD, TCRD, XCRD, YCRD
PRINT *, ’ENTER POLAR COORDINATES’
READ *, RCRD, TCRD
CALL CONVER(RCRD, TCRD, XCRD, YCRD)
** WORKS FVCG, NOT WF, F77 ; (TCRD) is VALUE**
PRINT *, XCRD, YCRD, RCRD, TCRD
TCRD = 150
CALL CONVER(RCRD, TCRD, XCRD, YCRD)
PRINT *, XCRD, YCRD, RCRD, TCRD
PRINT *, ’END OF PROGRAM’
STOP
END

SUBROUTINE CONVER(R, THETA, X, Y)

REAL R, THETA, X, Y
X = R* COS(THETA)
Y = R* SIN(THETA)
THETA = 1.0
END
C

Joe Lambert
/* call by reference and call by value */
#include <stdio.h>
#include <math.h>
void main(void)
{
    double rcrd,tcrd,xcrd, ycrd;
    void cnvr(double, double, double* ,double* );
    void cnvr3(double, double *, double* ,double* );
    printf("enter polar coordinates:\n");
    scanf("%lf %lf",&rcrd, &tcrd);
    printf("enter polar coordinates:\n");
    cnvr(rcrd,tcrd,&xcrd,&ycrd);
    printf("%f %f %f %f\n",xcrd,ycrd,rcrd,tcrd);
    tcrd=150;
    cnvr3(rcrd,&tcrd,&xcrd,&ycrd);
    printf("%f %f %f %f\n",xcrd,ycrd,rcrd,tcrd);
    printf("End of Program\n");
}

void cnvr(double r, double t, double *x, double *y)
{
    *x=r*cos(t);
    *y=r*sin(t);
    t=1.0;
}
void cnvr3(double r, double *t, double *x, double *y)
{
    *x = r * cos(*t);
    *y = r * sin(*t);
    *t = 1.0;
}
Pointers, Addresses, Values

C

Joe Lambert
A pointer is scalar type whose value holds an address.

```c
int i;
printf("%x \n", &i);
/* prints the address of i..*/
```

Pointers are separate data types.

Each type int, char, float, etc., has a corresponding pointer type.

```c
int *j;
declares j to be a pointer(an address) that points to an integer.
```

```c
#include<stdio.h>
void main()
{
 int i,*j;
 printf(" *j = %x \n",j);
 j= &i;
 printf(" address of i is %x \n", &i);
 printf(" *j = %x \n",j);
}
```
Cisms

#include<stdio.h>
void main()
{
int i,*j;
char c, *d = &c;
j = &i;
printf(" *j = %x \n",j);
printf(" address of c is %x \n",d);
printf(" address of i is %x \n",j);
}
Passing Parameters

Parameters in C passed by value
i.e. are not changed
BUT you can force
call by reference by using pointers

```c
void main()
{
    int x, y, *ip;
    void changit(int x, int *ip);
    x = 1;
    y = 3;
    ip = &y;
    printf("%d %d %d\n", x, y, *ip);
    changit(x, ip);
    printf("%d %d %d\n", x, y, *ip);
}
void changit(int x, int *ip);
{
    x += 5;
    *ip += 5;
```

Joe Lambert
printf("inside %d %d \n", x, *ip);
yields

1 3 3
inside 6 8
1 8 8

Note x unchanged but y is changed.
Do you really understand?

void swap ( int x , int y)
{
    int temp;
    temp = x;  
    x = y;  
    y = temp; 
}

    a = 3;  
    b = 5;  
    swap(a,b); yields ??

    a = 3;  
    b = 5;
void swap ( int *x, int *y)
{
    int temp;
    temp=*x;
    *x=*y;
    *y= temp;
}

a=3;
b=5;
swap(&a,&b); yields ??

a=5;
b=3;
CONTENTS

- Simple Programs in FORTRAN and C  1
- Control Structures in FORTRAN and C  16
- Subprograms in FORTRAN and C .  35
- Array Structures in FORTRAN and C .  58
- Files in FORTRAN and C ...........  65
- Additional Data Types in FORTRAN and C ..................  72
- C calls FORTRAN subroutines .....  93
CONTENTS

✓ Simple Programs in FORTRAN and C  1
✓ Control Structures in FORTRAN and C  16
✓ Subprograms in FORTRAN and C  .  35
⇒ Array Structures in FORTRAN and C  .  58
  • One Dimensional Arrays
  • Two Dimensional Arrays

- Files in FORTRAN and C  ............  65
- Additional Data Types in FORTRAN and C  .........................  72
- C calls FORTRAN subroutines  .....  93
Array Structures in FORTRAN and C

One Dimensional Arrays
******************
PROGRAM VLIST

* THIS PROGRAM SHOWS ARRAY

* STRUCTURE

******************
INTEGER LIMIT, NUMVEL, I
PARAMETER (LIMIT = 50)
REAL VELOC(LIMIT)

PRINT *, "ENTER NUMBER OF VELOCITIES"
READ *, NUMVEL
PRINT *, "ENTER VELOCITIES, 1 PER LINE"

DO 10 I = 1, NUMVEL
   READ *, VELOC(I)
10 CONTINUE

PRINT *, 'END OF PROGRAM'
STOP
END
C

/* This program shows array structure*/
#include <stdio.h>
void main(void)
{
#define LIMIT 50
int numvel,i
float veloc[LIMIT];

printf("Enter number of velocities:");
scanf(" %d",&numvel);
printf("Enter the velocities ,1 per line\n");
for (i=0;i<numvel;i++) /* i=0?? < numvel*/
    scanf("%f",&veloc[i]);

printf("End of Program\n");
}
Two Dimensional Arrays
PROGRAM MATMUL
* THIS PROGRAM SHOWS MATRIX MULTIPLICATION

INTEGER I, J, K, M, N, P, Q
PARAMETER (M=5, N=6, P=6, Q=7)
REAL M1(M,N), M2(P,Q), M3(M,Q)

DO 30 I = 1, M
   DO 20 J = 1, Q
      SUM = 0
      DO 10 K = 1, N
         SUM = SUM + M1(I,K) * M2(K,J)
      10 CONTINUE
      M3(I,J) = SUM
   20 CONTINUE
30 CONTINUE

PRINT *, ’ END OF PROGRAM’
STOP
END
/* This program shows matrix multiplication*/

#include <stdio.h>
void main(void)
{
    int i,j,k;
    int m=5,n=6,p=6,q=7;
    float m1[m][n],m2[p][q],m3[m][q];

    for (i=0;i<m;i++)
    {
        for (j=0;j<q;j++)
        {
            m3[i][j]=0;
            for (k=1;k<n;k++)
                m3[i][j] += m1[i][k]*m2[k][j];
        }
    }
    /* again off by one */

    printf("End of Program\n");
}
Simple Programs in FORTRAN and C  1

Control Structures in FORTRAN and C  16

Subprograms in FORTRAN and C  35

Array Structures in FORTRAN and C  58

Files in FORTRAN and C  65

Additional Data Types in FORTRAN and C  72

C calls FORTRAN subroutines  93
CONTENTS

✓ Simple Programs in FORTRAN and C  1
✓ Control Structures in FORTRAN and C  16
✓ Subprograms in FORTRAN and C  .  35
✓ Array Structures in FORTRAN and C  .  58

⇒ Files in FORTRAN and C  ..........  65
  ■ I/O Files
  ■ FORTRAN FILE Statements
  ■ C FILE Mode Statements
  ■ FORTRAN File Positioning
  ■ C FILE Positioning

■ Additional Data Types in FORTRAN and C  .........................  72

■ C calls FORTRAN subroutines  .....  93
Files in FORTRAN and C

I/O Files

FORTRAN

Joe Lambert
**Program FUN**

*This Program File I/O*

```
PROGRAM FUN

* This Program File I/O

REAL R1, R2, R3, RC

OPEN (UNIT=10, FILE='RES3', STATUS='OLD')
OPEN (UNIT=11, FILE='RESC', STATUS='NEW')
READ (10, *) R1, R2, R3

RC = 1.0 / (1.0 / R1 + 1.0 / R2 + 1.0 / R3)
WRITE (11, *) RC

CLOSE (10)
CLOSE (11)

PRINT *, 'END OF PROGRAM'
STOP
END
```
C

/* This program shows file i/o*/

#include <stdio.h>
void main(void)
{
    FILE *fip,*fop;
    float r1,r2,r3,rc;

    fip= fopen("res3","r");
    fop= fopen("resc","w");

    fscanf(fip,"%f %f %f",&r1,&r2,&r3);
    rc=1/(1/r1+1/r2+1/r3);
    fprintf(fop,"%f\n",rc);

    fclose(fip);
    fclose(fop);
}
### FORTRAN FILE Statements

<table>
<thead>
<tr>
<th>Statement</th>
<th>Parameter</th>
<th>Values</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>FILE</td>
<td></td>
<td>File name</td>
</tr>
<tr>
<td>STATUS</td>
<td></td>
<td>OLD</td>
<td>File exists</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NEW</td>
<td>File created</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SCRATCH</td>
<td>No FILE specified</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UNKNOWN</td>
<td>None of above</td>
</tr>
<tr>
<td>IOSTAT</td>
<td></td>
<td>0</td>
<td>Success</td>
</tr>
<tr>
<td></td>
<td></td>
<td>positive</td>
<td>No success</td>
</tr>
<tr>
<td>ERR</td>
<td></td>
<td>number</td>
<td>GOTO</td>
</tr>
<tr>
<td>ACCESS</td>
<td></td>
<td>SEQUENTIAL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DIRECT</td>
<td></td>
</tr>
<tr>
<td>FORM</td>
<td></td>
<td>FORMATTED</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UNFORMATTED</td>
<td></td>
</tr>
<tr>
<td>RECL</td>
<td></td>
<td>number</td>
<td>rec length</td>
</tr>
<tr>
<td>CLOSE</td>
<td>IOSTAT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATUS</td>
<td></td>
<td>KEEP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DELETE</td>
<td></td>
</tr>
</tbody>
</table>
**C FILE Mode Statements**

FILE *fopen(const char *filename, const char *mode)

fopen(”myfile”, ”r”);

<table>
<thead>
<tr>
<th>Mode Values</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>open file for read</td>
</tr>
<tr>
<td>w</td>
<td>create file, if exists delete current</td>
</tr>
<tr>
<td>a</td>
<td>append, open or create file, write at end</td>
</tr>
<tr>
<td>r+</td>
<td>open for update, read-write</td>
</tr>
<tr>
<td>w+</td>
<td>create for update, if exists delete current</td>
</tr>
<tr>
<td></td>
<td>read write</td>
</tr>
<tr>
<td>a+</td>
<td>append read-write</td>
</tr>
</tbody>
</table>
## FORTRAN File Positioning

<table>
<thead>
<tr>
<th>Statement</th>
<th>Parameter</th>
<th>Values</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>REWIND</td>
<td>IOSTAT</td>
<td>0 positive</td>
<td>Start of file</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Success</td>
</tr>
<tr>
<td></td>
<td>ERR</td>
<td>number</td>
<td>No success</td>
</tr>
<tr>
<td>BACKSPACE</td>
<td>IOSTAT</td>
<td>0 positive</td>
<td>to previous record start</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Success</td>
</tr>
<tr>
<td></td>
<td>ERR</td>
<td>number</td>
<td>No success</td>
</tr>
<tr>
<td>ENDFILE</td>
<td></td>
<td></td>
<td>writes EOF</td>
</tr>
</tbody>
</table>
C FILE Positioning

void rewind(FILE *stream);

rewind(”myfile”);

int fseek(FILE *stream, long int offset, int whence);

fseek(”myfile”,1000, SEEK_SET);

<table>
<thead>
<tr>
<th>Statement</th>
<th>Parameter</th>
<th>Values</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>fseek</td>
<td>stream</td>
<td>FILE *</td>
<td>file name</td>
</tr>
<tr>
<td></td>
<td>offset</td>
<td>long int</td>
<td>num of chars to move</td>
</tr>
<tr>
<td></td>
<td>whence</td>
<td>SEEK_SET, SEEK_CUR, SEEK_END</td>
<td>from begin position, from current position, from end position</td>
</tr>
</tbody>
</table>
CONTENTS

- Simple Programs in FORTRAN and C  1
- Control Structures in FORTRAN and C  16
- Subprograms in FORTRAN and C  35
- Array Structures in FORTRAN and C  58
- Files in FORTRAN and C  65
- Additional Data Types in FORTRAN and C  72
- C calls FORTRAN subroutines  93
<table>
<thead>
<tr>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ ▪ Simple Programs in FORTRAN and C  1</td>
</tr>
<tr>
<td>✓ ▪ Control Structures in FORTRAN and C  16</td>
</tr>
<tr>
<td>✓ ▪ Subprograms in FORTRAN and C  35</td>
</tr>
<tr>
<td>✓ ▪ Array Structures in FORTRAN and C  58</td>
</tr>
<tr>
<td>✓ ▪ Files in FORTRAN and C  65</td>
</tr>
<tr>
<td>⇒ ▪ Additional Data Types in FORTRAN and C  72</td>
</tr>
<tr>
<td>▪ Characters</td>
</tr>
<tr>
<td>▪ Strings</td>
</tr>
<tr>
<td>▪ String Comparison</td>
</tr>
<tr>
<td>▪ Complex Data Type</td>
</tr>
<tr>
<td>▪ C calls FORTRAN subroutines  93</td>
</tr>
</tbody>
</table>
Additional Data Types in FORTRAN and C

Characters

FORTRAN

Joe Lambert
PROGRAM CHAR

CHARACTER*20 ITEM
CHARACTER*4 NAME(50)
PRINT*, 'ENTER ITEM ENCLOSED IN '''
READ *, ITEM
PRINT 5, ITEM
5 FORMAT(1X, 'ITEM IS ', A)

NAME(1) = 'MARY'
NAME(2) = 'JOHN'
PRINT *, 'END OF PROGRAM'
STOP
END
C

/* This program shows char data type */

#include<stdio.h>
void main()
{
char c, item[20];
char *name[50];
printf("Enter item \n",item);
scanf("%s",item);
c=’9’;
name[0]="mary";
name[1]="Johnny";
printf("%s %c %s %s\n",item, c, name[0],name[1]);
}
PROGRAM STRNG

ITEM = 'FORTRAN'
NEXT = 'C'
LAST = ITEM//NEXT
PRINT 5,ITEM
PRINT 5,ITEM(2:4)
PRINT 5,LAST
LAST = NEXT//LAST
PRINT 5,LAST
5 FORMAT(1X,'ITEM IS ',A)

PRINT *, 'END OF PROGRAM'
STOP
END

yields
ITEM IS FORTRAN
ITEM IS ORT
ITEM IS FORTRAN C
ITEM IS C FORTRAN
END OF PROGRAM
C

Joe Lambert
#include<stdio.h>
#include<string.h>
void main(void)
{
    char *strcpy(char *s1,const char *s2);
    char *strcat(char *s1,const char *s2);

    char item[10],next[10],last[20];
    strcpy(item,"FORTRAN");
    strcpy(next,"C");
    printf("item is %s\n",item);
    strcpy(last,item);
    strcat(last,next);
    printf("item is %s\n",last);
    strncpy(last,item+1,3);
    last[3]=\0;
    printf("item is %s\n",last);
}

yields
item is FORTRAN
item is FORTRANC
item is ORT
## String Comparison

### FORTRAN

<table>
<thead>
<tr>
<th>Function</th>
<th>Meaning</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLT(S1,S2)</td>
<td>less than</td>
<td>.TRUE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.FALSE.</td>
</tr>
<tr>
<td>LLE(S1,S2)</td>
<td>less than or equal</td>
<td>.TRUE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.FALSE.</td>
</tr>
<tr>
<td>LGT(S1,S2)</td>
<td>greater than</td>
<td>.TRUE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.FALSE.</td>
</tr>
<tr>
<td>LGE(S1,S2)</td>
<td>greater than or equal</td>
<td>.TRUE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>.FALSE.</td>
</tr>
</tbody>
</table>
C

#include <string.h>
int strcmp(const char *s1, const char *s2);

<table>
<thead>
<tr>
<th>Function</th>
<th>Meaning</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>strcmp(S1,S2)</td>
<td>S1 less than S2</td>
<td>negative</td>
</tr>
<tr>
<td>strcmp(S1,S2)</td>
<td>S1 equal S2</td>
<td>zero</td>
</tr>
<tr>
<td>strcmp(S1,S2)</td>
<td>S1 greater than S2</td>
<td>positive</td>
</tr>
</tbody>
</table>
Complex Data Type
PROGRAM CMPLX

REAL R,C,W,MAG,PH
COMPLEX I,HW

I=(0.0,1.0)
R=3.5
C=2.1
W=1.7

HW=(W*R*C*I)/(1.0 + W*R*C*I)
MAG=CAABS(HW)
PH= ATAN(AIMAG(HW)/REAL(HW))

PRINT *, HW, MAG, PH
PRINT *,’ END OF PROGRAM’
STOP
END

yields
( 0.993636, 7.95227E-02) 0.996813 7.98618E-02
END OF PROGRAM
```c
#include<stdio.h>
#include<stdlib.h>

typedef struct {
    float real;
    float imag;
}complex;

void casgn(float , float , complex * );
complex *cmul (complex *,complex *);
complex *cadd(complex *,complex *);
complex *cdiv(complex *,complex *);
complex *coerce(float);
void main(void)
{
    float r=3.5,c=2.1,w=1.7;

    complex i, hw;
    complex *tp,*tn,*tr;

    casgn(0,1,&i);
    tp=cmul(cmul(cmul(coerce(r),coerce(c)),coerce(w)),&i);
    tn=cadd(coerce(1),tp);
    tr=cdiv(tp,tn);
```
printf(" %f %f \n",tr->real,tr->imag);
}
void casgn(float r, float j, complex *c)
{
    c->real=r;
    c->imag=j;
}

complex *cmul (complex *u ,complex *v)
{
    complex *temp;
    temp=malloc(sizeof(complex));
    temp->real=u->real*v->real-u->imag*v->imag;
    temp->imag=u->imag*v->real+u->real*v->imag;
    return(temp);
}

complex *cadd(complex *u ,complex *v)
{
    complex *temp;
    temp=malloc(sizeof(complex));
    temp->real=u->real+v->real;
    temp->imag=u->imag+v->imag;
return(temp);
}

complex *cdiv(complex *u, complex *v) 
{
    complex *temp;
    float mag;
    mag = v->real * v->real + v->imag * v->imag;
    temp = malloc(sizeof(complex));
    temp->real = (u->real * v->real + u->imag * v->imag) / mag;
    temp->imag = (u->imag * v->real - u->real * v->imag) / mag;
    return(temp);
}
complex *coerce(float x)
{
    complex *temp;
    temp = malloc(sizeof(complex));
    temp->real = x;
    temp->imag = 0;
    return(temp);
}
  yields
| 0.993636 | 0.079523 |
CONTENTS

- Simple Programs in FORTRAN and C  1
- Control Structures in FORTRAN and C  16
- Subprograms in FORTRAN and C  35
- Array Structures in FORTRAN and C  58
- Files in FORTRAN and C  65
- Additional Data Types in FORTRAN and C  72
- C calls FORTRAN subroutines  93
CONTENTS

- Simple Programs in FORTRAN and C 1
- Control Structures in FORTRAN and C 16
- Subprograms in FORTRAN and C 35
- Array Structures in FORTRAN and C 58
- Files in FORTRAN and C 65
- Additional Data Types in FORTRAN and C 72

⇒ C calls FORTRAN subroutines 93
  - Common Question with Answers
  - Simple Example
  - For More Information
C calls FORTRAN subroutines

Common Question with Answers
FORTRAN to C Workshop  C calls FORTRAN subroutines

: .......
:   * How to call NAG Fortran Library with C (under Unix) ? *
:
: We have a NAG Fortran Library on our machine but I do not
: know how to call them in my C program. Can anybody give
: me some advice about how to do this?
:
Hello,
we have some programs written in C calling the NAG library (which
is written in Fortran). There are a number of
important things you must be aware of:

1) Fortran uses a column wise storage of matrices while C stores them
row wise. This means that when you want to parse a matrix from your
C-program to the NAG (-fortran-) routine you must transpose the matrix
in your program before entering the routine. Of course, any output
from such a routine must be transposed again.

If you omit this step, then probably your program will run (because
it has data to compute on) but it will generate wrong answers.

B.T.W. if you have the Fortran source code (of any routine) then
on some platforms you may use compiler directives specifying
that the Fortran compiler must use row wise storage. Some platforms
support these directives. However watch out with this if you call
the same routine from another Fortran routine/program.

2) Your Fortran compiler may add an underscore "_" to the routine name
in the symbol table e.g. subroutine example(...) becomes
example_ in the table. Hence in the calling C-program/routine
you must add a trailing underscore ! Otherwise the loader will complain
about an undefined symbol "example" while "example_" is loaded.
However, check your compiler for this. For example the Fortran compiler
on VAX-VMS systems does NOT add a trailing underscore (there watch out
with the fact that the VAX-Fortran compiler translates everything
in uppercase).

3) Fortran passes its variables by reference. This means that you MUST
give addresses in your calling C-program (i know, this is a stupid
remark but it is too often forgotten (my experience ....)).

Joe Lambert
FORTRAN to C Workshop

4) Watch out especially with float's and double's. Make sure that the size of the variable in the calling program is identical to the size in the Fortran routine e.g. double ----> real*8, float ----> real.

5) Remember that the array index in Fortran starts at 1 while in C this is at index 0; hence a parsed array fortran_array[1:100] must be used in the C-routine/program as c_array[0:99].

--
Rudi Vankemmel | These are my opinions, not those of
IMEC vzw. - ASP division | my employer, so don't take them away
Process and Device Modelling group
Kapeldreef 75 phone: (32)-(0)16/28 13 37
3001 Leuven fax: (32)-(0)16/28 12 14
Belgium email: vankemme@imec.be

another post:
--From: beardsl@mepsi.mobil.com (Reginald Beardsley)
--Subject: A: [F77-C] Fortran to c and c to fortran
--Message-ID: <2081se$26j@d1sn31.dal.mobil.com>

I do a lot of this. The following is true on the following machines. I cannot say about others.

Sun 3 & 4
IBM RS/6000
SGI
DECstation
Intergraph Clipper (Apogee & Green Hills compilers)
H-P 7xx

1) If possible, do not pass strings to FORTRAN from C or vice versa.

2) Do not mix I/O on the same file descriptors.

3) Do all your math in FORTRAN, and all the rest in C if at all possible.

4) NEVER ever attempt to write the equivalent of a FORTRAN function that returns a character variable in C. Life is too short for the suffering it causes.

5) If you do ANY I/O in FORTRAN, you MUST use a FORTRAN main program.

Joe Lambert
FORTRAN to C Workshop

6) FORTRAN always passes pointers.
7) **FORTRAN** passes string lengths **BY VALUE** in the order the strings appear in
the argument list. These do **NOT** appear in the **FORTRAN** argument list,
but will appear in the **C** argument list.

8) You will need to take care of nulls and blanks spaces explicitly if you
ignore #1 above.

9) The Sun **FORTRAN** compiler used **lex** and **yacc** to do the conversion of a run
time format from a character variable. If you use **lex** and **yacc** either
rename the variables and functions or partially link before you link
to the **FORTRAN** libraries.

10) **FORTRAN** symbols have trailing underscores appended. Some compilers require
a compiler flag to get this. Use it! It makes the code more portable.

11) Don’t pass structures. If you must access a structure element, pass a
pointer through to a routine which passes back the element pointer.

12) Don’t forget that the storage orders for arrays is opposite and
transposition is expensive.

---

Reginald H. Beardsley  Contract Consultant/Programmer
Mobil Exploration and Production Technology

Office: (214)-851-8547  beardsl@dal.mobil.com
Home: (214)-306-3907
Simple Example
Subroutine CmplxRef(w)
complex w
w = (6,7)
return
end

main()
{
struct complex(float r, i);
struct complex d1;
struct complex *w = &d1;
extern cmplxref_( );
complxref_(w);
}

f77 -c CmplexRef
cc -c main.c
f77 CmplxRef.o main.o
a.out
For More Information

From: khb@chiba.Eng.Sun.COM (Keith Bierman-khb@chiba.eng.sun.com::SunPro)
Netnews: comp.lang.fortran
Subject: Fortran FAQ
Date: 05 Jun 1994 19:33:43 GMT

Q14) How do I call f77 from C (and visa versa)
     This is quite platform dependent. For Suns see the FORTRAN User's

     There is a package available from usenet which attempts to make
     this "quick and easy" for a wide range of platforms:

Host ftp.germayy.eu.net

    Location: /newsarchive/comp.sources.misc/volume20
        DIRECTORY drwxr-xr-x 512 Jul 7 1993 cfortran

Host ftp.sunet.se

    Location: /pub/usenet/comp.sources.misc/volume20
        DIRECTORY drwxrwxr-x 512 May 28 1993 cfortran

Host ftp.wustl.edu

    Location: /usenet/comp.sources.misc/volume20
        DIRECTORY drwxr-xr-x 8192 Oct 30 15:09 cfortran

Host halcyon.com

    Location: /dec/.0/usenet/comp.sources.misc/volume20
        DIRECTORY dr-xr-xr-x 512 Jul 8 1993 cfortran

Joe Lambert
Host lth.se

Location: /pub/netnews/sources.misc/volume20
DIRECTORY drwxr-xr-x 512 Jun 7 1993 cfortran

Host math.mps.ohio-state.edu

Location: /pub/archives/comp.sources.misc/vol20
DIRECTORY drwxrwxr-x 512 Jun 2 1993 cfortran

It is on many other sites (around the world) too. See archie if you need other pointers.

For some systems, you have to initialize a runtime system explicitly if you call a different language, and stopping execution from the other program may not work.

The most recent version of cfortran.h is available via anon. ftp at zebra.desy.de.