12.010 Computational Methods of Scientific Programming
Fall 2008

Pointers

(not in F77)

1. `int a;` refers to value held at location a.
2. `int *ptr_to_a;` pointer to memory location where value is held.
3. `ptr_to_a=&a;` sets pointer to equal location in memory used by variable a.

Examples

```
main() {
float a;           /* Floating point number              */
  float *ptr_to_a;   /* Pointer to a floating point number */

  a = 7.;            /* Write 7. to memory location associated with a   */
  printf("Value in a == %f\n",a);

  ptr_to_a = &a;     /* Get the address of the memory location where    */
                     /* assignments to a get written.                   */
  printf("Memory address of a (in hexadecimal) == %X\n",ptr_to_a);

  /* Now use pointer to read value stored at an address                 */
  printf("Value stored at address %X == %f\n",ptr_to_a,*ptr_to_a);

  /* Write a new value to an address in memory  */
  *ptr_to_a = 3.;

  /* What value does a have now? */
  printf("Value in a == %f\n",a);

  /* In C arrays and pointers are the same thing! */
  /* [0] is ptr_to_a + offset of 0*4 bytes        */
  /* [1] is ptr_to_a + offset of 1*4 bytes        */
  /* etc......                                    */
  printf("Value stored at address %X == %f\n",ptr_to_a,ptr_to_a[0]);

}
```

Call by value

C

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

F77

```
      PROGRAM MAIN
      INTEGER I1, I2
```

```
void afunc(int, int);
main() {
 int i1, i2;
 i1 = 3; i2 = 4;
 printf("in main()");
 printf(" i1 == %d, i2
== %d\n",i1,i2);
 afunc( i1, i2 );
 printf("in main()");
 printf(" i1 == %d, i2
== %d\n",i1,i2);
}

void afunc(int a, int b) {
 printf("in afunc()");
 printf(" a == %d, b
== %d\n",a,b);
 a = 7; b = 6;
 printf("in afunc()");
 printf(" a == %d, b
== %d\n",a,b);
}
```

```
       I1 = 3
       I2 = 4
       WRITE(6,'(A,I4,I4)') 'in AFUNC()
I1,I2: ',I1,I2
       CALL AFUNC(I1, I2)
       WRITE(6,'(A,I4,I4)') 'in AFUNC()
I1,I2: ',I1,I2
       END

       SUBROUTINE AFUNC( A, B )
       INTEGER A, B
       WRITE(6,'(A,I4,I4)') 'in AFUNC()
A,B: ',A,B
       A = 7
       B = 6
       WRITE(6,'(A,I4,I4)') 'in AFUNC()
A,B: ',A,B
      END
```

Call by reference

C

```
#include <stdio.h>

void afunc(int *, int *);
main() {
 int i1, i2;
 i1 = 3; i2 = 4;
 printf("in main()");
 printf(" i1 == %d, i2
== %d\n",i1,i2);
 afunc( &i1, &i2 );
 printf("in main()");
 printf(" i1 == %d, i2
== %d\n",i1,i2);
}

void afunc(int *a, int *b) {
 printf("in afunc()");
 printf(" a == %d, b
== %d\n",*a,*b);
 *a = 7; *b = 6;
 printf("in afunc()");
 printf(" a == %d, b
== %d\n",*a,*b);
}
```

F77

```
       PROGRAM MAIN
       INTEGER I1, I2
       I1 = 3
       I2 = 4
       WRITE(6,'(A,I4,I4)') 'in AFUNC()
I1,I2: ',I1,I2
       CALL AFUNC(I1, I2)
       WRITE(6,'(A,I4,I4)') 'in AFUNC()
I1,I2: ',I1,I2
       END

       SUBROUTINE AFUNC( A, B )
       INTEGER A, B
       WRITE(6,'(A,I4,I4)') 'in AFUNC()
A,B: ',A,B
       A = 7
       B = 6
       WRITE(6,'(A,I4,I4)') 'in AFUNC()
A,B: ',A,B
      END
```

structures and defined types

C

```
#include <stdio.h>

typedef struct { float cx;
                 float cy;
                 float cz;
                 int   color; }
t_point;
void point_print( t_point );

main() {
 t_point point;
 point.cx=3.; point.cy=3.; point.cz=2.;
point.color=10;
 point_print(point);
}

void point_print(t_point point) {
 printf(" cx == %f, cy
== %f\n",point.cx,point.cy);
 printf(" cz == %f, color
== %d\n",point.cz,point.color);
}
```

F77

```
       PROGRAM MAIN
       REAL POINT_COORD(3)
       INTEGER POINT_COLOR
       POINT_COORD(1) = 3.
       POINT_COORD(2) = 3.
       POINT_COORD(3) = 2.
       POINT_COLOR    = 10
       CALL
POINT_PRINT( POINT_COORD,
POINT_COLOR )
       END

       SUBROUTINE
POINT_PRINT( POINT_COORD,
POINT_COLOR )
       REAL    POINT_COORD(3)
       INTEGER POINT_COLOR
       WRITE(6,'(A,3F12.4,I4)')
'COORD(1,2,3): ',
     &  POINT_COORD(1),
     &  POINT_COORD(2),
     &  POINT_COORD(3),
     &  POINT_COLOR
      END
```

malloc()

```
#include < stdio.h >

 typedef struct { float cx;
                  float cy;
                  float cz;
                  int   color; } t_point;
 void point_print( t_point );

 main() {
  int nel;
  t_point *points;

  nel=10;

  points = (t_point *) calloc(nel, sizeof(*points) );

  for(i=0;i < nel;++i) {
   point_print(points[i]);
  }
 }
```

```
void point_print(t_point point) {
 printf(" cx == %f, cy == %f\n",point.cx,point.cy);
 printf(" cz == %f, color == %d\n",point.cz,point.color);
}
```