Pointers – Basics-II

Atul Gupta

Pointer Arithmetic

```
main( )
    {
        int i = 3, *x ;
        float j = 1.5, *y ;
        char   k = 'c', *z ;

        printf ( "\nValue of i = %d", i ) ;
        printf ( "\nValue of j = %f", j ) ;
        printf ( "\nValue of k = %c", k ) ;
        x = &i ;
        y = &j ;
        z = &k ;
        printf ( "\nOriginal address in x = %u", x ) ;
        printf ( "\nOriginal address in y = %u", y ) ;
        printf ( "\nOriginal address in z = %u", z ) ;
        x++ ;
        y++ ;
        z++ ;
        printf ( "\nNew address in x = %u", x ) ;
        printf ( "\nNew address in y = %u", y ) ;
        printf ( "\nNew address in z = %u", z ) ;
    }
```

Output?

Pointer Arithmetic

1. Addition of a number to a pointer. For example,

```
int  i = 4, *j, *k ;
     j = &i ;
     j = j + 1 ;
     j = j + 9 ;
     k = j + 3 ;
```

2. Subtraction of a number from a pointer. For example,

```
int  i = 4, *j, *k ;
      j = &i ;
      j = j - 2 ;
      j = j - 5 ;
      k = j - 6 ;
```

Pointer Arithmetic

3. Subtraction of one pointer from another

```
main( )
    {
       int arr[ ] = { 10, 20, 30, 45, 67, 56, 74 } ;
       int *i, *j ;

       i = &arr[1] ;
       j = &arr[5] ;
       printf ( "%d %d", j - i, *j - *i ) ;
    }
```

Pointer Arithmetic

4. Comparison of two pointer variables

```
main( )
    {
       int array[ ] = { 10, 20, 36, 72, 45, 36 } ;
       int *j, *k ;

       j = &array [ 4 ] ;
       k = ( array + 4 ) ;

       if ( j == k )
          printf ( "The two pointers point to the same location" ) ;
       else
          printf ( "The two pointers do not point to the same location" ) ;
    }
```

Pointer Arithmetic

- *Do not attempt following operations on pointers !*
  - Addition of two pointers
  - Multiplication of a pointer with a constant
  - Division of a pointer with a constant

Pointers are Good!

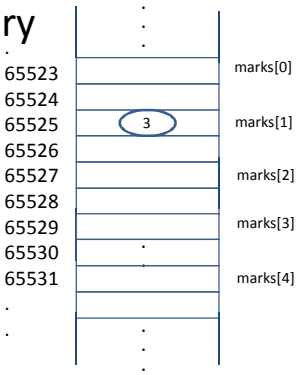
- What do the following declarations stand for?

```
Int  ***i;
float  **j;
char ****k;

void  funct1 (int *, char *);
Float *funct2 (float *, float **);
Int  **funct3 (float *, char **);
```

Arrays

- A collection of similar elements stored in consecutive memory locations
- An array is known as a 'subscripted variable'

```
int  marks[5];
```

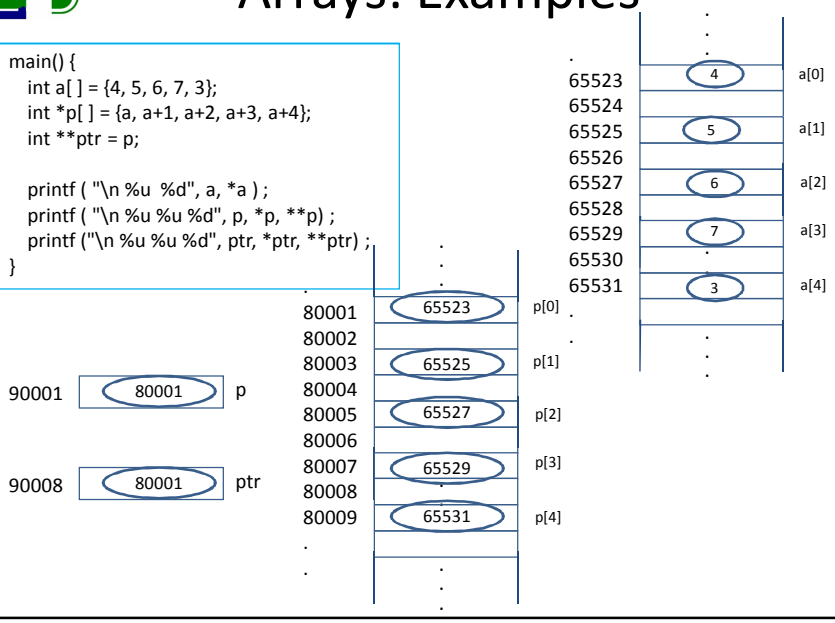


Arrays

- Functions manipulating arrays
  - An array element can be passed as an argument to a function (as a primitive data type)
  - Array can be passed as an argument to a function
  - An array element can be returned as a primitive data type
  - An array can also be returned from a function
- An array of pointers

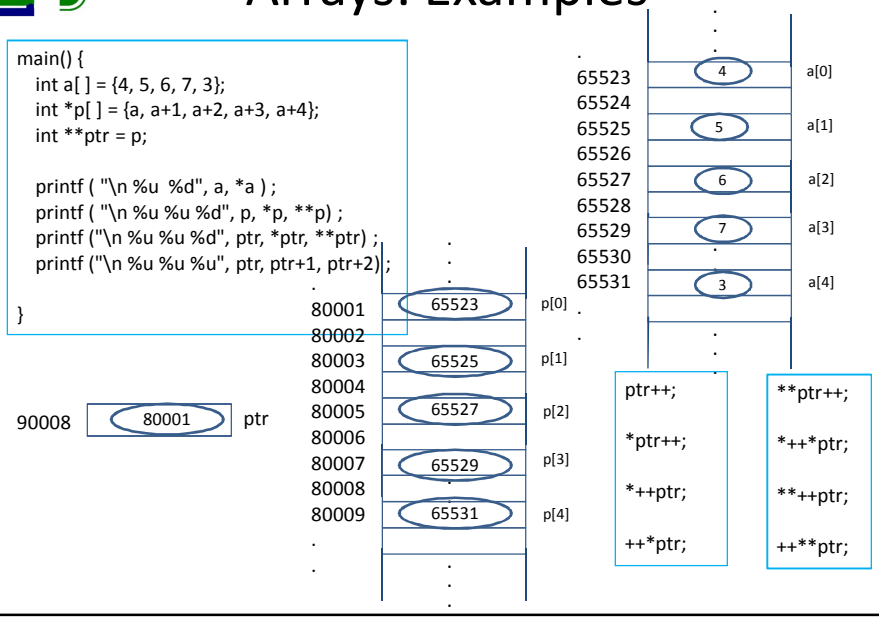
Arrays: Examples

```
main() {
   int a[ ] = {4, 5, 6, 7, 3};
   int *p[ ] = {a, a+1, a+2, a+3, a+4};
   int **ptr = p;

   printf ( "\n %u  %d", a, *a ) ;
   printf ( "\n %u %u %d", p, *p, **p) ;
   printf ("\n %u %u %d", ptr, *ptr, **ptr) ;
}
```



Arrays: Examples

```
main() {
   int a[ ] = {4, 5, 6, 7, 3};
   int *p[ ] = {a, a+1, a+2, a+3, a+4};
   int **ptr = p;

   printf ( "\n %u  %d", a, *a ) ;
   printf ( "\n %u %u %d", p, *p, **p) ;
   printf ("\n %u %u %d", ptr, *ptr, **ptr) ;
   printf ("\n %u %u %u", ptr, ptr+1, ptr+2) ;
}
```



Structures

- A group of meaningful, related data to identify an entity

```
main() {
        struct book{
          char accession_no[15];
          char *title;
          char[] *authors;
          float price;
          …
        }
        //structure declarations
        struct book book1,book2,book3;
        struct book textbooks[10000];
        struct book *book_ptr;
}
```

```
main() {
    printf("Book1 price is %f", book1.price);
    printf("Book1 price is %f", book_ptr→price);
}
```

‘struct’ in C: An Example

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

struct Employee{              /* the structure type */
   char lname[20];            /* last name */
   char fname[20];            /* first name */
   int age;                   /* age */
   float wages;               /* e.g. 12.75 per hour */
};

struct Employee anEmployee;            /* declare an structure */
void showName(struct Employee *p);  /* function prototype */

void showName(struct Employee *e)
{
   printf("\n%s ", e->fname);  /* p points to a structure */
   printf("%s ", e->lname);
   printf("%d\n", e->age);
}
```

```
int main(void)
{
   struct Employee *st_ptr;       /* a pointer to a structure */
   st_ptr = &anEmployee;       /* point the pointer to my_struct */
   strcpy(anEmployee.lname,"Jensen");
   strcpy(anEmployee.fname,"Ted");
   printf("\n%s ",anEmployee.fname);
   printf("%s\n",anEmployee.lname);
   getch();
   anEmployee.age = 63;
   showName(st_ptr);        /* pass the pointer */
   getch();
   return 0;
}
```