


Paper 102: Programming & Problem solving through C

Lecture-17:Unit-II Pointers



More on Pointers

- Pointers store the address of another variable
- Pointers can be declared as follows
 - `int *p;`
 - `char *p;`
 - `float *p;`
 - `double *p;`
- The pointer p can store addresses of integer variable, character variable, float variable or double variable depending on how it was declared.
- Address can be assigned to it as follows
 - `p=&a;`
- Value of a can be access using *p (indirection operator)

void pointers

- Pointers are defined to be a specific data type
- It cannot hold the address of any other type of variable.
- It is incorrect to use

```
int *f;  
float a;  
f=&a;
```
- This restriction can be overcome by using a **void** pointer

Cont...void pointers

- A void pointer is a general purpose pointer.
- It can be declared as
- Pointers declared in this manner can store the address of any type of variables

```
void *ptr;
```

```
ptr=&intvar;
```

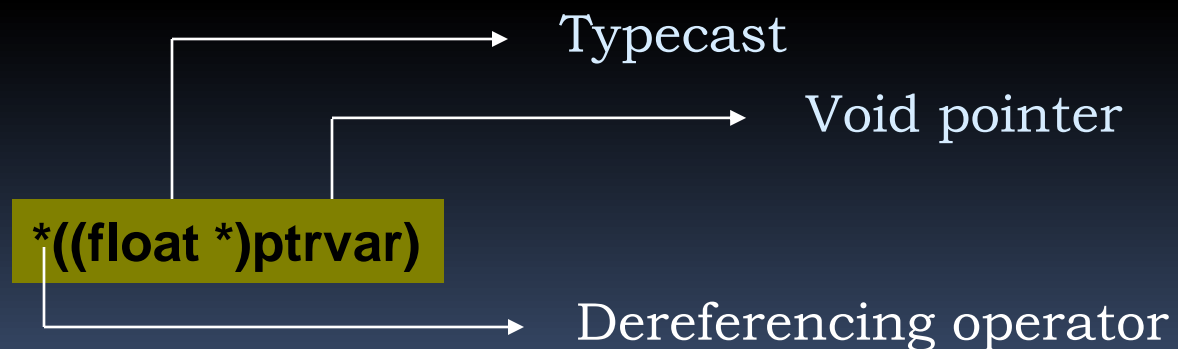
```
ptr=&floatvar;
```

```
ptr=&doublevar;
```

```
ptr=&charvar;
```

Cont...void pointers

- Pointers to void cannot be directly dereference like other pointer variables by using the indirection operator *
- Before dereferencing, a pointer to void must be suitably typecast to the required data type



Example

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

```
void main(void)
```

```
{
```

```
    void *ptr;
```

```
    int a=5;
```

```
    float p=3.14;
```

```
    ptr=&p;
```

```
    printf("Value of Pi is=%f", *((float *)ptr));
```

```
    ptr=&a;
```


```
    printf("Value of a is=%d", *((int *)ptr));
```

```
    getch();
```

```
}
```



Near and far pointers

- Pointers that can access only address within the same segment would store only the offset in that segment
 - Near pointers
 - Pointers that can store both segment and offset can access addresses of a different segment.
 - Far pointers
- 

Pointer Arithmetic

- Addition of a number to a pointer

```
int i=4, *j, *k;
```

```
j=&i;
```

```
j=j+1;
```

```
k=j+3;
```



Pointer Arithmetic

- Subtraction of a number from a pointer

```
int i=4, *j, *k;
```

```
j=&i;
```

```
j=j+2;
```

```
k=j-1;
```



Pointer Arithmetic

- Subtraction of a pointer from another
 - Possible only if both pointers point to elements of the same array
 - The result is the number of elements separating the corresponding array elements

```
int a[]={10,20,30,40,50,60,70};
```

```
int *j,*k;
```

```
j=&a[1];
```

```
k=&a[5];
```

```
printf("%d %d",j-k,*j-*k);
```

a[0]	j	k
10	2012	2020
2010	3010	4010

Pointer Arithmetic

- Comparison of two pointer variables
 - Possible only if both pointers point to objects of the same data type
 - Usually done testing of equality or inequality

```
int a[]={10,20,30,40,50,60,70};
```

```
int *j,*k;
```

```
j=&a[1];
```


```
k=(a+1);
```

```
if (j==k) printf("pointing to same location");
```

```
else printf("not pointing to same location");
```



Pointer Arithmetic

- Invalid operations
 - Addition of two pointers
 - Multiplication of a pointer with a constant
 - Division of a pointer with a constant
- 

Pointers and Arrays

//dynamic 1-d array

```
#include<stdio.h>
void main()
{
    int *n,size,i;
    clrscr();
    printf("\nHow many elements:");
    scanf("%d",&size);
    n=(int *)malloc(size * sizeof(int));
    for(i=0;i<size;i++)
    {printf("\n enter a number:");
    scanf("%d",(n+i));
    }
    printf("\n the output is:\n");
    for(i=0;i<size;i++)
    {
        printf("%d\t",*(n+i));
    }
    getch();
}
```

Pointers and Arrays -cont..

//an array of pointers

```
#include<stdio.h>
#define MAXR 10
void inputdata(int *n[MAXR],int r, int c);
void main()
{
    int *b[MAXR];
    int r,c,i,j;
    clrscr();
    printf("\n how many rows?");
    scanf("%d",&r);
    printf("\n how many cols?");
    scanf("%d",&c);
    for(i=0;i<r;i++)
        b[i]=(int *) malloc(c * sizeof(int));
    printf("\n input data");
    inputdata(b,r,c);

    printf("\n b array\n");
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            printf("%d\t",*(b+i) + j));
        }
        printf("\n");
    }
    getch();
}
```

An array of pointers

```
void inputdata(int *n[MAXR],int
r,int c)
{
    int i,j;
    for(i=0;i<r;i++)
    {
        for(j=0;j<c;j++)
        {
            printf("\nenter number:");
            scanf("%d",&*(n+i)+j));
        }
    }
}
```

Creating a 2-d array Dynamically

```
#include<stdio.h>
void main()
{
    int **array1;
    int nrows,ncols,i,j;
    clrscr();
    printf("\n enter the rows:");
    scanf("%d",&nrows);
    printf("\n enter the cols:");
    scanf("%d",&ncols);
    // Allocate an array of pointers.
    // Then initialize each pointer to a dynamically
    // allocated row.
    array1 = (int **) malloc( nrows * sizeof(int*));
    if( array1 == NULL){
        printf("Out of memory");
    }
    for(i = 0; i < nrows; i++){
        array1[i] = (int *)malloc(ncols * sizeof(int));
    }
}
```


Creating a 2-d array Dynamically

```
for (i=0; i<nrows; i++){
    for (j=0; j<ncols; j++)
    {
        printf("\n enter a number:");
        scanf("%d",&array1[i][j]);
    }
}
for (i=0; i<nrows; i++){
    for (j=0; j<ncols; j++)
        printf("%8d",array1[i][j]);
    printf("\n");
}
}
```

//using command line arguments

```
#include<stdio.h>
#include<string.h>
void main(int argc,char *argv[])
{
    int i,j;
    if(argc==1)
        exit();
    printf("number of arguments=%d",argc);
    for(i=0;i<argc;i++)
    {
        puts(argv[i]);
        printf("%d",strlen(argv[i]));
    }
}
```

1. **argc** is the total number of arguments
2. ***argv[]** is an array of strings, storing the list of arguments including the program name
3. To run create an exe file and execute from the dos prompt:

D:\>cmdline *argument1 argument2*

Class Assignment

- Create an array to store the marks of a list of students in three subjects. Write a program to find the average of marks of each student, and display the Top Five students based on their average marks. Use Dynamic arrays.
- Write a program to display the Calendar of any Month in a particular Year.