Chapter 7

Strings and Pointers

String

- String is an array of characters including the terminating null (**\0**) character

```
char szFamilyName[5];
szFamilyName[0]='J';
szFamilyName[1]='i';
szFamilyName[2]='n';
szFamilyName[3]='\0';
printf("Family Name=%s\n",szFamilyName);
```

char	szFamilyName
0	**' J '**
1	**' i '**
2	**' n '**
3	**'\0'**
4	

- Character array declaration must be large enough to include the **\0**
- A string is enclosed in double quotes, "Jin"
- String initilializing
  - within a declaration, characters enclosed in quotes

String Functions – gets(),puts()

- ***gets()*** get a line from the stdin stream

  *char *__gets__(char *buffer);*

- ***puts()*** write a string to stdout, replacing the string's terminating null character ('\0') with a newline character ('\n') in the output stream.

  int ***puts***(const char *str);

- Required header: <stdio.h>

```
#include "stdafx.h"
//#include <string.h>
int _tmain(int argc, _TCHAR* argv[])
{
char  szName[10];
  gets(szName);
  puts(szName);
  puts(szName);
  return 0;
}
```

String Functions- *sizeof*

- ***sizeof***

  computes the number of bytes of a specified variable or variable type

  When the ***sizeof*** operator is applied to an array, it yields the total number of bytes in that array

```
int _tmain(int argc, _TCHAR* argv[])
{
…
int  Score[10],nLength;
int  nOneCharSize,nOneIntSize,nArraySize1,nArraySize2;
  nOneCharSize=sizeof(char);
  nOneIntSize=sizeof(nLength);
  nArraySize2=sizeof(Score);
  nArraySize1=sizeof(szName);
…
}
```

String Functions

- ***strlen()***
  - Returns the length in bytes of string
- ***strcpy()***
  - Copies a string
- ***strcat()***
  - (Concatenate)Appends the second string to the first string
- ***strcmp()***
  - Compares two strings
- ***strchr()***
  - Finds the specified character
- ***atoi()***
  - Convert a string to integer
- ***isalpha()***
  - Check an integer to see it is an alphabetic character.
- **Required header: <string.h>**

```
#include <string.h>
int _tmain(int argc, _TCHAR* argv[])
{
char   szName[10],szString1[20];
int    Score[10],nLength,nOneCharSize,nOneIntSize;
int    nArraySize1,nArraySize2,nCompareResult;
  nLength=strlen(szName);
  strcpy(szString1,"I am");
  strcpy(szString1,"I am a very very good student");//???
  strcat(szString1," student ");
  strcat(szString1,szName);
  nCompareResult=strcmp(szString1,szName);
  return 0;
}
```

String Functions – sprintf()

- Write formatted data to a string

int *sprintf*(char *buffer, const char *format [,argument] ...);

```
#include <string.h>
int _tmain(int argc, _TCHAR* argv[])
{
char  szString2[100],szItem[20] ="iPhone";
int     nNumber;
float  fPrice;
  nNumber=2;
  fPrice=5266.50;
  printf("%-10s  %+03d×%8f=%12.4f\n","iPhone",2,fPrice,nNumber*fPrice);
  sprintf(szString2,"%10s %3d×%5.2f=%5.1f\n",szItem,2,fPrice,nNumber*fPrice);
  printf("%s\n",szString2);
  return 0;
}
```

File Operation

fputs()-Write a string to a file

```
int fputs(const char *str,FILE *stream);
```

fprintf()-Print formatted data to a stream.

```
int fprintf(FILE *stream,const char *format [, argument ]... );
```

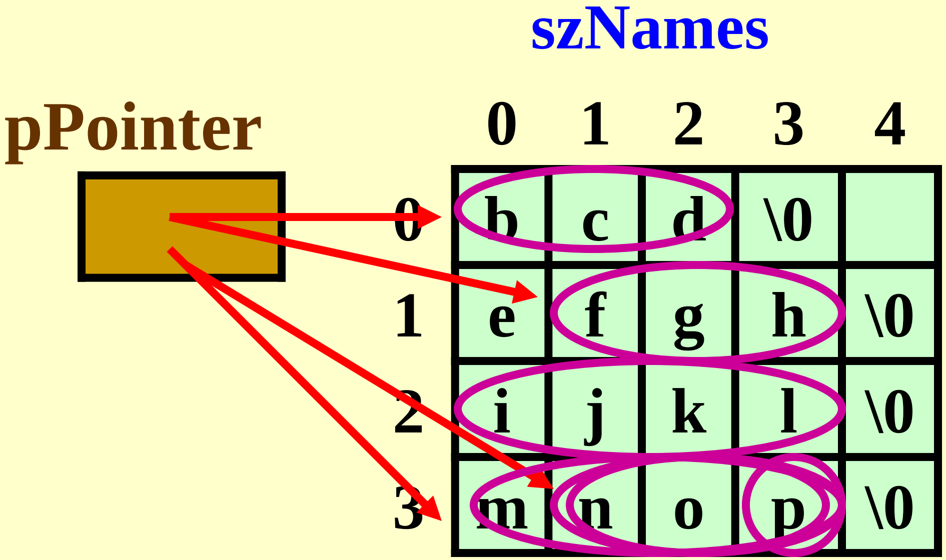
fwrite()-Writes data to a stream

```
size_t fwrite(const void *buffer,
                     size_t size,
                     size_t count,
                     FILE *stream );
```

```
int _tmain(int argc, _TCHAR* argv[])
{
…
FILE  *fpWrite;
  if((fpWrite=fopen("D:\\Data3.txt","w"))==NULL)
     puts("Error");
  else
  {
     fputs(szString2,fpWrite);
     fprintf(fpWrite,"%10s %3d×%5.2f=%5.1f\n",szItem,2,fPrice,nNumber*fPrice);
     fwrite(szString2,1,strlen(szString2),fpWrite);
     fclose(fpWrite);
  }
  return 0;
}
```

Pointer Arithmetic

```
#include <string.h>
int _tmain(int argc, _TCHAR* argv[])
{
char  szNames[4][5]={{'b','c','d','\0'},"efgh","ijkl","mnop"};
char  *pPointer;
  pPointer=&szNames[0][0];
  puts(pPointer);
  pPointer=&szNames[1][1];
  puts(pPointer);
  puts(szNames[2]);
  pPointer=szNames[3];
  puts(pPointer);
  puts(pPointer+1);
  pPointer++;
  puts(pPointer);
  puts(pPointer+2);
}
```



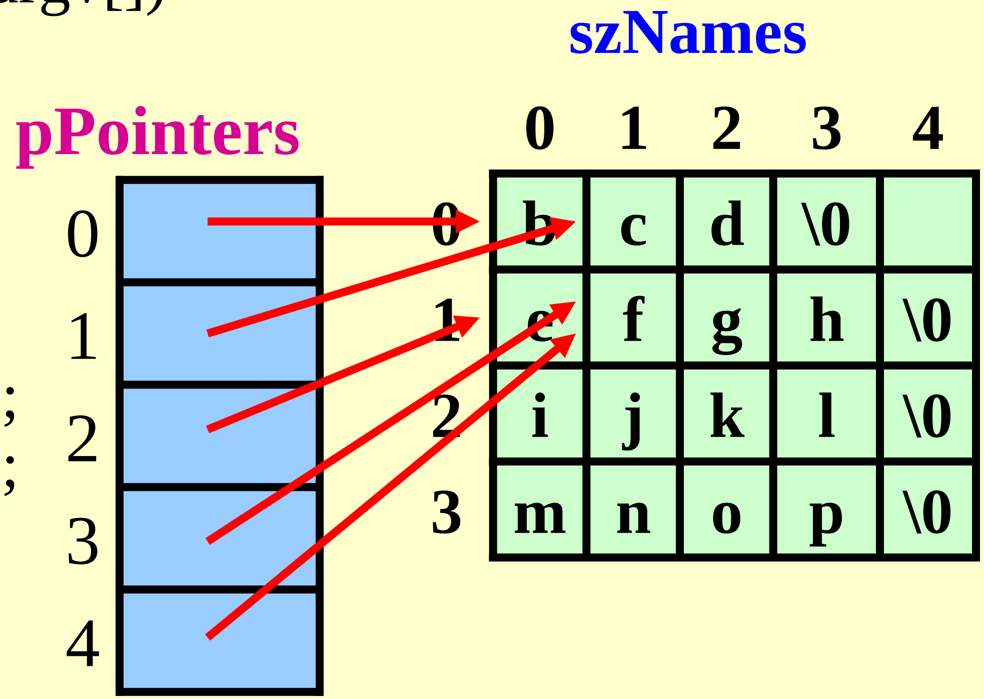
szNames

pPointer

	0	1	2	3	4
0	b	c	d	\0	
1	e	f	g	h	\0
2	i	j	k	l	\0
3	m	n	o	p	\0

Array of Pointers

```
#include <string.h>
int _tmain(int argc, _TCHAR* argv[])
{
char  cCharacter;
char  szNames[4][5];
…
char  *pPointers[5];
  pPointers[0]=&szNames[0][0];
  pPointers[1]=&szNames[0][1];
  pPointers[2]=szNames[1];
  pPointers[3]=szNames[1]+1;
  pPointers[4]=pPointers[2]+1;
  for(int i=0;i<5;i++)
     puts(pPointers[i]);
}
```

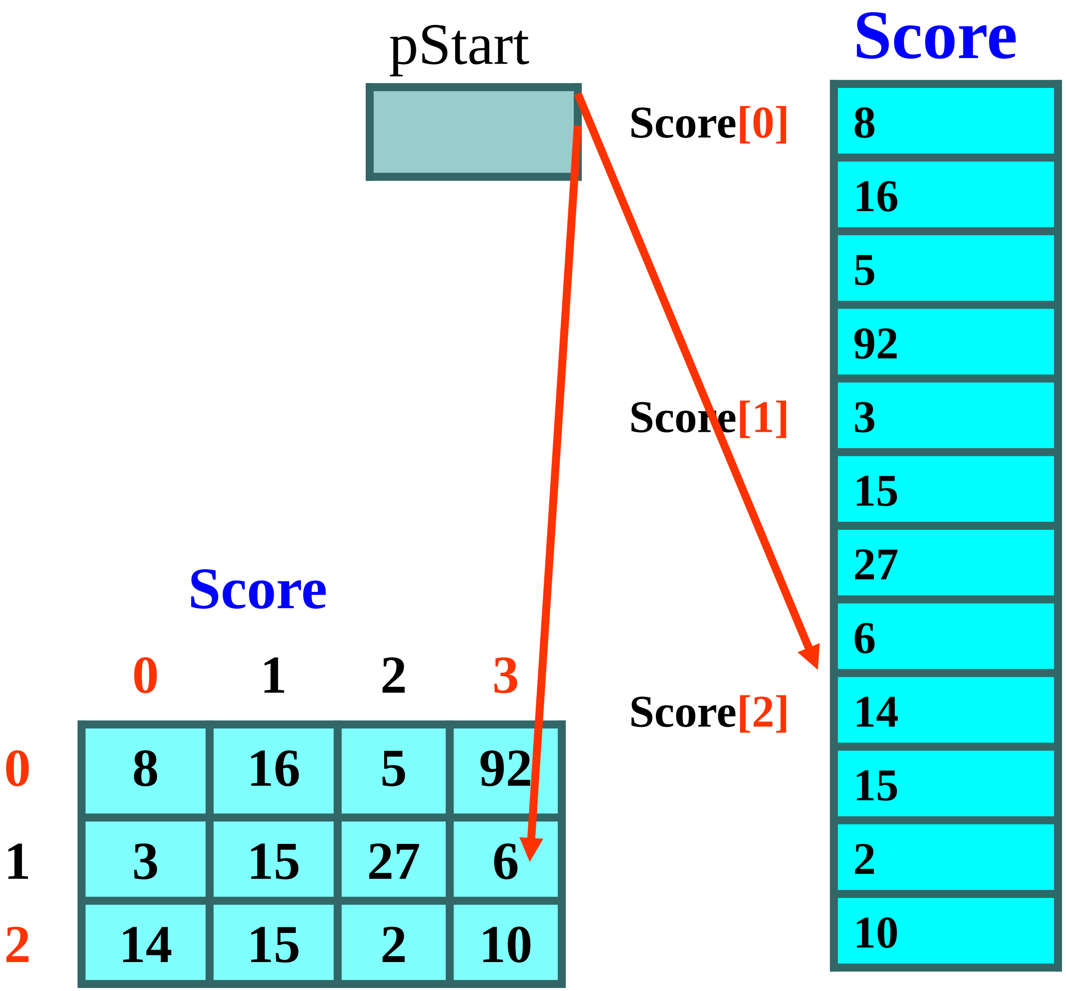


	0	1	2	3	4
0	b	c	d	\0	
1	e	f	g	h	\0
2	i	j	k	l	\0
3	m	n	o	p	\0

2-D Array Name as Pointer

- Array element can be expressed by pointer

```
int   Score[3][4]={{8,16,5,92},{3,15,27,6},{14,15,2,10}};
int   *pStart,x[10];
    pStart=&Score[1] [3];
     x[0]=*pStart;
     pStart++;
     x[1]=*pStart;
     x[2]=*pStart++;
     x[3]=*pStart;
     x[4]=*++pStart;
     x[5]=*(pStart+2);
```



Score

	0	1	2	3
0	8	16	5	92
1	3	15	27	6
2	14	15	2	10

Score

Score[0]	8
	16
	5
	92
Score[1]	3
	15
	27
	6
Score[2]	14
	15
	2
	10