

# Arrays and Strings (2H)

---

Copyright (c) 2014 - 2018 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to [youngwlim@hotmail.com](mailto:youngwlim@hotmail.com).

This document was produced by using LibreOffice.

---

Based on Embedded Software in C for an ARM Cortex M  
<http://users.ece.utexas.edu/~valvano/Volume1/>

# Array Declarations

```
short    data[5];      /* define data, allocate space for 5 16-bit integers */
char     string[20];   /* define string, allocate space for 20 8-bit characters */
int time, width[6];    /* define time, width, allocate space for 16-bit characters */
short    xx[10][5];    /* define xx, allocate space for 50 16-bit integers */
short    pts[5][5][5]; /* define pts, allocate space for 125 16-bit integers */
extern char buffer[]; /* declare buffer as an external character array */
```

# Array Reference

```
short x, *pt, data[5];          /* a variable, a pointer, and an array */

void Set(void){
    x  = data[0];               /* set x equal to the first element of data */
    x  = *data;                 /* set x equal to the first element of data */
    pt = data;                  /* set pt to the address of data */
    pt = &data[0];             /* set pt to the address of data */
    x  = data[3];               /* set x equal to the fourth element of data */
    x  = *(data+3);             /* set x equal to the fourth element of data */
    pt = data+3;                /* set pt to the address of the fourth element */
    pt = &data[3];             /* set pt to the address of the fourth element */
}
```

# Pointers and Array Names

```
short *pt, data[5];          /* a pointer, and an array */

void Set(void){
    pt      = data;          /* set pt to the address of data */
    data[2] = 5;             /* set the third element of data to 5 */
    pt[2]   = 5;             /* set the third element of data to 5 */
    *(pt+2) = 5;             /* set the third element of data to 5 */
}
```

```
short buffer[5], data[5];   /* two arrays */

void Set(void){
    data    = buffer;        /* illegal assignment */
}
```

# String Functions

```
typedef unsigned int size_t;
```

```
void *memchr (void *, int, size_t);
int memcmp (void *, void *, size_t);
void *memcpy (void *, void *, size_t);
void *memmove (void *, void *, size_t);
void *memset (void *, int, size_t);
char *strcat (char *, const char *);
char *strchr (const char *, int);
int strcmp (const char *, const char *);
int strcoll (const char *, const char *);
char *strcpy (char *, const char *);
size_t strcspn (const char *, const char *);
size_t strlen (const char *);
char *strncat (char *, const char *, size_t);
int strncmp (const char *, const char *, size_t);
char *strncpy (char *, const char *, size_t);
char *strpbrk (const char *, const char *);
char *strrchr (const char *, int);
size_t strspn (const char *, const char *);
char *strstr (const char *, const char *);
```

# FIFO Queue (1)

```
/* Index,counter implementation of the FIFO */
#define FifoSize 10          /* Number of 8 bit data in the Fifo */

unsigned char Putl;          /* Index of where to put next */
unsigned char Getl;         /* Index of where to get next */
unsigned char Size;        /* Number currently in the FIFO */
                           /* FIFO is empty if Size=0 */
                           /* FIFO is full if Size=FifoSize */
char Fifo[FifoSize];      /* The statically allocated data */

void Fifo_Init(void) {
    Putl = Getl = Size = 0; /* Empty when Size==0 */
}

int Fifo_Put (char data) {
    if (Size == FifoSize )
        return(0);          /* Failed, fifo was full */
    else{
        Size++;
        Fifo[Putl++] = data; /* put data into fifo */
        if (Putl == FifoSize) Putl = 0; /* Wrap */
        return(-1);         /* Successful */
    }
}
```



# FIFO Queue (2)

```
int Fifo_Get (char *datapt) {  
    if (Size == 0)  
        return(0);                /* Empty if Size=0 */  
    else {  
        *datapt = Fifo[Getl++];  
        Size--;  
        if (Getl == FifoSize) Getl = 0;  
        return(-1);  
    }  
}
```

## References

- [1] Essential C, Nick Parlante
- [2] Efficient C Programming, Mark A. Weiss
- [3] C A Reference Manual, Samuel P. Harbison & Guy L. Steele Jr.
- [4] C Language Express, I. K. Chun
- [5] “A Whirlwind Tutorial on Creating Really Teensy ELF Executables for Linux”  
<http://cseweb.ucsd.edu/~ricko/CSE131/teensyELF.htm>
- [6] <http://en.wikipedia.org>
- [7] <http://www.muppetlabs.com/~breadbox/software/tiny/teensy.html>
- [8] <http://csapp.cs.cmu.edu/public/ch7-preview.pdf>