Strings

- Strings are multiple characters in a certain order
  - "my car"
  - "Please enter three integers"
- C allows variable types for single characters, but not for strings
- char type arrays are used to represent strings
- Strings in C are terminated by the null character, "\0"
- So the string "my car"
  - Notice that space is counted as a character
  - Need to add the null character to the end of the string
  - Minimum length of array needed is 7 (including the null character)
- What is the minimum length of an array to store "Please enter three integers"?

Declaring and Initializing Strings

- Strings are declared as arrays of characters
  ```c
  char FirstName[15];
  char LastName[25];
  ```
- Strings can also be defined as constants
  ```c
  #define ERR1 "Date is wrong"
  #define ERR2 "Cannot Leave Name Field Blank"
  ```
- Strings can be initialized when they are declared
  ```c
  char FirstName[15] = {S, a, d, i, k};
  char LastName[25] = {E, s, m, e, l, o, g, i, u};
  ```
Strings in Memory

- **char** \(x[15] = \text{"John Smith"}\)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>J</td>
<td>o</td>
<td>h</td>
<td>n</td>
<td>S</td>
<td>m</td>
<td>i</td>
<td>t</td>
<td>h</td>
<td>0</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>15</td>
</tr>
</tbody>
</table>

- **char** \(x[15] = \text{"John Adam Smith"}\)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>J</td>
<td>o</td>
<td>h</td>
<td>n</td>
<td>A</td>
<td>d</td>
<td>a</td>
<td>m</td>
<td>S</td>
<td>m</td>
<td>i</td>
<td>t</td>
<td>h</td>
<td>0</td>
<td>-1</td>
<td>23</td>
</tr>
</tbody>
</table>

- Length of the string should be one less than the array size

Arrays of Strings

- Strings are arrays of characters → single dimensional arrays
- Lists are arrays of strings → two dimensional arrays
- Players
  - Ahmet
  - Hasan
  - Naciye
  - Ayse
  - Nuri
- This list can be represented as a two dimensional array (5 rows, ? columns)

\[
\text{char Players}[5][7] = \{\text{"Ahmet"}, \text{"Hasan"}, \text{"Naciye"}, \text{"Ayse"}, \text{"Nuri"}\}
\]
Storing in Memory

All have a value of 7000

Printing Strings

- Use printf and scanf
- %s is the placeholder for strings
- example
  ```c
  printf("%s", "This is a book");
  printf("%8s*%5s", "12345", "123");
  printf("%8s*%5s", "12345", "123");
  ```
Printing Strings

• Normally strings are right justified, but a minus sign after the % sign makes the string left justified.

```c
printf("%s*%s", "123456", "123");
prints >123456*123

printf("%s\n%5s\n%5s\n%5s\n", "Ana", "Baba", "Cocuk");
prints > Ana
> Baba
> Cocuk

printf("-%5s\n-%5s\n-%5s\n", "Ana", "Baba", "Cocuk");
prints >Ana
>Baba
>Cocuk
```

Printing Arrays of Strings

```c
#include <stdio.h>
int main(void)
{
    int i;

    printf("%s\n", Players);
    printf("%s\n", Players[0]);
    printf("%c\n", Players[0][0]);

    for(i=0;i<=4;i++) printf("%s\n", Players[i]);

    return(0);
}
```
Reading Strings

- Reading strings with `scanf` works the same way as reading numbers:
  - scanning for numbers begin with a numeric character (or decimal point) and ends when a non-numeric character is encountered.
  - scanning for strings starts after the first white space character and ends when a white space character is reached (white space characters are spaces, tabs, newlines, ...)
  - a null character (`\0`) is placed after the last character is read.

```c
char x[8];
scanf("%s", x);
```

The entry is stored in memory as shown:

```
J o h n \0
```

or user enters

> John
>...
Reading Strings

```c
char c[5];
int n;
char d[5];
int t;
```

```c
scanf("%s %d %s %d", c, &n, d, &t);
```

user enters

> CENG 114 Thu 240

or user enters

> CENG
> 114 Thu
> 240

The entries are stored in memory as shown:

```
<table>
<thead>
<tr>
<th>Var</th>
<th>Addr</th>
<th>Val</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>c[0]</td>
<td>7000</td>
<td>C</td>
<td>01000011</td>
</tr>
<tr>
<td>c[1]</td>
<td>7001</td>
<td>E</td>
<td>01000101</td>
</tr>
<tr>
<td>c[2]</td>
<td>7002</td>
<td>N</td>
<td>01001110</td>
</tr>
<tr>
<td>c[3]</td>
<td>7003</td>
<td>G</td>
<td>01000111</td>
</tr>
<tr>
<td>c[4]</td>
<td>7004</td>
<td>?0</td>
<td>00000000</td>
</tr>
<tr>
<td>n</td>
<td>7005</td>
<td>114</td>
<td>01110010</td>
</tr>
<tr>
<td>d[0]</td>
<td>7006</td>
<td>T</td>
<td>01010100</td>
</tr>
<tr>
<td>d[1]</td>
<td>7007</td>
<td>h</td>
<td>01101000</td>
</tr>
<tr>
<td>d[2]</td>
<td>7008</td>
<td>u</td>
<td>01110101</td>
</tr>
<tr>
<td>d[3]</td>
<td>7009</td>
<td>?0</td>
<td>00000000</td>
</tr>
<tr>
<td>t</td>
<td>7011</td>
<td>240</td>
<td>11110000</td>
</tr>
</tbody>
</table>
```

Example

```c
#include <stdio.h>

#define NUM_STD100 /* Maximum number of students */
#define NM_LEN15 /* Maximum length of each name */

int main(void)
{
    char name[NUM_STD][NM_LEN]; /* Array of student names */
    int Grade[NUM_STD]; /* Array of student grades */
    int i = 0; /* Loop index */
    int num_std = 0; /* Actual num of students read */
    char ans = 'y'; /* if equals 'y' continue reading */
    char junk; /* reading newlines */
```
Example

```c
while(ans == 'y') {
    printf("Please enter Last Name and Grade: ");
    scanf("%s %d", name[i], &Grade[i]);
    i++;

    printf("Are there more students? [y/n]: ");
    scanf("%c%c%c", &junk, &ans, &junk);
}
num_std = i;
for(i = 0; i < num_std; i++)
    printf("%20s %4d\n", name[i], Grade[i]);
return(0);
```

String Library Functions

- `#include <string.h>`
- `strlen` return the length of string not counting \0
- `strcpy` copies string from source to dest
- `strncpy` copies n chars from source to dest
- `strcat` appends string from source to end of dest
- `strncat` appends n chars from source to end of dest
- `strcmp` compares two strings alphabetically
- `strncmp` compares the first n chars of two strings
- `strtok` breaks string into tokens using delimiters