Chapter Fourteen

Strings Revisited

1

Strings

- A string is an array of characters
- A string is a pointer to a sequence of characters
- A string is a complete entity

2

Strings as Arrays

- A string is an array of characters terminated by the null character '\0'

3

An Example

```
int findFirstVowel(char word[])
{
   int i;

   for (i = 0; word[i] != '\0'; i++) {
      if (isVowel(word[i])) return i;
   }
   return -1;
}
```

4

An Example

```
int findFirstVowel(char *word)
{
   char *cp;

   for (cp = word; *cp != '\0'; cp++) {
      if (isVowel(*cp)) return cp - word;
   }
   return -1;
}
```

5

An Example

```
int findFirstVowel(string word)
{
   int i;

   for (i = 0; i < stringLength(word); i++) {
      if (isVowel(ithChar(word, i))) return i;
   }
   return -1;
}
```

6

Common Pitfalls

```
main()
{
   char carray[6];
   char *cptr;

   carray = "hello";  /* error: array name has no lvalue */
   cptr = "hello";
   strcpy(carray, "hello");
   strcpy(cptr, "hello");  /* error: no memory spaces */
}
```

7

Common Pitfalls

```
char *charToString(char ch)
{
   char *cptr;

   cptr = (char *) malloc(2);
   cptr[0] = ch;
   cptr[1] = '\0';
   return cptr;
}
```

```
char *charToString(char ch)
{
   char carray[2];

   carray[0] = ch;
   carray[1] = '\0';
   return carray; /* error */
}
```

8

ANSI String Library

- strcpy(dst, src): copy characters from src to dst
- strncpy(dst, src, n): copy at most n characters from src to dst
- strcat(dst, src): append characters from src to the end of dst
- strncat(dst, src, n): append at most n characters from src to the end of dst
- strlen(s): return the length of s

9

ANSI String Library

- strcmp(s1, s2): return an integer indicating the result of the comparison
- strncmp(s1, s2, n): like strcmp but compare at most n characters
- strchr(s, ch): return a pointer to the first instance of ch in s (or NULL)
- strrchr(s, ch): return a pointer to the last instance of ch in s (or NULL)
- strstr(s1, s2): return a pointer to the first instance of s2 in s1 (or NULL)

10

strcpy

```
void strcpy(char dst[], char src[])
{
   int i;

   for (i = 0; src[i] != '\0'; i++) {
      dst[i] = src[i];
   }
   dst[i] = '\0';
}
```

```
void strcpy(char *dst, char *src)
{
   while (*dst++ = *src++) ;
}
```

11

strncpy

```
void strncpy(char dst[], char src[], int n)
{
   int i;

   for (i = 0; src[i] != '\0' && i < n; i++) {
      dst[i] = src[i];
   }
   for (; i < n; i++) {
      dst[i] = '\0';
   }
}
```

12

strcat

```
void strcat(char dst[], char src[])
{
   int i, j;

   for (i = 0; dst[i] != '\0'; i++) ;
   for (j = 0; src[j] != '\0'; j++, i++) {
      dst[i] = src[j];
   }
   dst[i] = '\0';
}
```

13

strncat

```
void strncat(char dst[], char src[], int n)
{
   int i, j;

   for (i = 0; dst[i] != '\0'; i++) ;
   for (j = 0; src[j] != '\0' && j < n; j++, i++) {
      dst[i] = src[j];
   }
   for (; j < n; j++, i++) {
      dst[i] = '\0';
   }
}
```

14

strlen

```
int strlen(char str[])
{
   int i;

   for (i = 0; str[i] != '\0'; i++) ;
   return i;
}
```

15

strcmp

```
int strcmp(char s1[], char s2[])
{
   int i;

   for (i = 0; s1[i] != '\0' && s2[i] != 0; i++)  {
      if (s1[i] < s2[i]) return –1;
      if (s1[i] > s2[i]) return 1;
   }
   if (s1[i] < s2[i]) return –1;
   if (s1[i] > s2[i]) return 1;
   return 0;
}
```

16

strncmp

```
int strncmp(char s1[], char s2[], int n)
{
   int i;

   for (i = 0; i < n && s1[i] != '\0' && s2[i] != 0; i++)  {
      if (s1[i] < s2[i]) return –1;
      if (s1[i] > s2[i]) return 1;
   }
   if (i == n || s1[i] == s2[i]) return 0;
   if (s1[i] < s2[i]) return –1;
   return 1;
}
```

17

strchr

```
char *strchr(char s[], char ch)
{
   int i;

   for (i = 0; s[i] != '\0'; i++)  {
      if (ch == s[i]) return &s[i];
   }
   return NULL;
}
```

18

strrchr

```
char *strrchr(char s[], char ch)
{
   int i;
   char *ptr;

   ptr = NULL;
   for (i = 0; s[i] != '\0'; i++)  {
      if (ch == s[i]) ptr = &s[i];
   }
   return ptr;
}
```

19

strstr

```
char *strstr(char s1[], char s2[])
{
   int i, len1, len2;

   len1 = strlen(s1);
   len2 = strlen(s2);
   for (i = 0; i < len1-len2; i++)  {
      if (!strncmp(&s[i], s2, len2)) return &s[i];
   }
   return NULL;
}
```

20

An Example

First Middle Last

Last, First Middle

21

An Example

```
static void invertName(char result[], char name[]) {
   int len;   char *sptr;
   len = strlen(name);
   sptr = strrchr(name, ' ');
   if (sptr != NULL) len++;
   if (len > MaxName) Error("Name too long");
   if (sptr == NULL) {
      strcpy(result, name);
   } else {
      strcpy(result, sptr + 1);
      strcat(result, ", ");
      strncat(result, name, sptr-name);
      result[len] = '\0';
   }
}
```

22

Implementing strlib.h

- Using string.h, the user uses the type char *, and needs to worry about the memory allocation for strings
- Using strlib.h, the user uses the type string, and needs not worry about the memory allocation for strings. The memory for new strings are automatically allocated from heap

23

Pass-Through Functions

```
int stringLength(string s)
{
   return strlen(s);
}

int stringCompare(string s1, string s2)
{
   return strcmp(s1, s2);
}
```

24

String Allocation Functions

```
static string createString(int len)
{
   return (string) malloc(len+1);
}
```

25

String Allocation Functions

```
string charToString(char ch)
{
   string  result;
   result = createString(1);
   result[0] = ch;
   result[1] = '\0';
   return result;
}
```

26

String Allocation Functions

```
string strcat(string s1, string s1)
{
   string s;
   int i, len1, len2;
   len1 = strlen(s1);  len2 = strlen(s2);
   s = createString(len1 + len2);
   for (i = 0; i < len1; i++)  s[i] = s1[i];
   for (i = 0; I < len2; i++)  s[i+len1] = s2[i];
   s[len1+len2] = '\0';
   return s;
}
```

27