

CSE1301 Computer Programming: Lecture 21 Character Strings

1


Topics

- Representation
- Declaration
- Index of a char in a string
- String operations
- Common mistakes

2

Representation

- *Recall:* Main memory
 - contiguous array of cells
 - each cell has an address

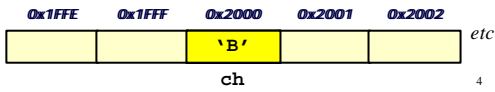


3

Representation (cont)

- *Recall:* Variable declaration
 - sets aside a “box” to contain a value

Example: `char ch;`
`ch = 'B';`

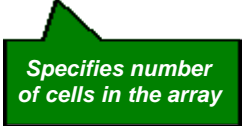


4

Representation (cont)

- String declaration
 - sets aside an array of cells
 - each cell contains a **char**
 - **address** of first cell in the array

Example: `char name[5];`

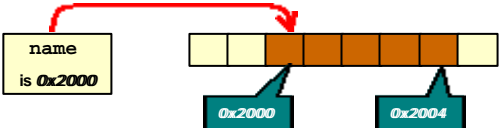


5

Representation (cont)

- String declaration
 - sets aside an array of cells
 - each cell contains a **char**
 - **address** of first cell in the array

Example: `char name[5];`



6

Character Arrays vs Character Strings

- A character string is a char array
- A character string *must* have the terminating character (' \0')
- The terminating character allows scanf() and printf() to handle character strings

7

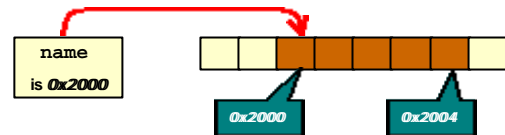
Character Strings

Declaration 1:

```
char name[5];
```

Declaration 2:

```
#define MAXLENGTH 5
char name[MAXLENGTH];
```



8

String Input/Output

```
#include <stdio.h>
#define MAXLENGTH 15
int main()
{
    char string1[MAXLENGTH];
    char string2[MAXLENGTH];
    scanf("%s %s", string1, string2);
    printf("%s %s\n", string1, string2);
    return 0;
}
```

No ampersand (&!)

9

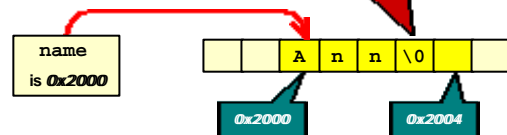
Character String Declaration

Declaration 1:

```
char name[5] = "Ann";
```

Terminating Character:

- Marks the end of string
- Special char: ' \0'
- aka NUL (single L)



10

Character String Declaration

Declaration 1:

```
char name[5] = "Ann";
```

Could have defined this as an array:

```
char name[5] = {'A', 'n', 'n', '\0'};
```

0x2000

0x2004

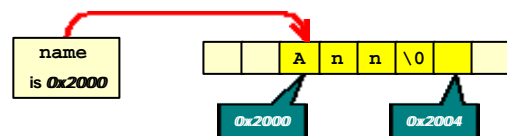
11

Character String Declaration (cont)

Declaration 1:

```
char name[5] = "Ann";
```

Can store at most 4 letters, because of '\0'



12

Character String Declaration (cont)

Declaration 2: ⚠️

```
char name[] = "Ann";
```

Takes up an extra cell for '\0'

13

Character String Declaration (cont)

Declaration 3: ⚠️ ⚠️

```
char *name = "Ann";
```

Result is "undefined" if you try to modify this string

14

Character String Declaration (cont)

Declaration 4: STOP

```
char name[];
```

String with arbitrary length?
No! Will cause an error "array size missing in 'name'"

15

A Char in a String

- The size of a character string is fixed
- Character at position *index*:
 - `string[index]`
 - first character has index 0

16

A Char in a String (cont)

```
char name[8] = "John";
int i = 2;
printf("Char at index %d is %c.\n", i, name[i]);
```

output: Char at index 2 is h.

17

A Char in a String (cont)

```
char name[8] = "John";
name[2] = 'X';
printf("Name: %s\n", name);
```

18

A Char in a String (cont)

```

char name[8] = "John";
name[2] = 'x';
printf("Name: %s\n", name);

```

out put: Name: JoXn

19

String Operations

- `#include <string.h>`
- Operations:
 - Assignment: `strcpy()`
 - Concatenation: `strcat()`
 - Comparison: `strcmp()`
 - Length: `strlen()`
- All rely on and maintain the NUL termination of the strings.

20

String Operation: Assignment

```

#include <stdio.h>
#include <string.h>

#define MAXLENGTH 100

int main()
{
    char string1[MAXLENGTH];
    char string2[MAXLENGTH];

    strcpy(string1, "Hello World!");
    strcpy(string2, string1);

    return 0;
}

```

string1: <garbage>
string2: <garbage>

21

String Operation: Assignment (cont)

```

#include <stdio.h>
#include <string.h>

#define MAXLENGTH 100

int main()
{
    char string1[MAXLENGTH];
    char string2[MAXLENGTH];

    strcpy(string1, "Hello World!");
    strcpy(string2, string1);

    return 0;
}

```

string1: "Hello World!"
string2: <garbage>

22

String Operation: Assignment (cont)

```

#include <stdio.h>
#include <string.h>

#define MAXLENGTH 100

int main()
{
    char string1[MAXLENGTH];
    char string2[MAXLENGTH];

    strcpy(string1, "Hello World!");
    strcpy(string2, string1);

    return 0;
}

```

string1: "Hello World!"
string2: "Hello World!"

23

Common Mistake 1:

Incompatible types

Example:

```

char name1[5] = "Ann";
char name2[5] = "Dave";
name2 = name1;

```

Error: "lvalue required ..."

24

Common Mistake 2:
* *Not enough space*

```
char name[] = "Ann";
strcpy(name, "David");
```

name is 0x2000

0x2000 0x2003

25

Common Mistake 2:
* *Not enough space*

```
char name[] = "Ann";
strcpy(name, "David");
```

name is 0x2000

0x2000 0x2003

26

Caution 1:
Pointer Assignment

Example:

```
char *name1 = "Ann";
char *name2 = "Dave";
name2 = name1;
```

27

Caution 1:
Pointer Assignment

```
char *name1 = "Ann";
char *name2 = "Dave";
```

0x2000 name1

0x3990 name2

28

Caution 1:
Pointer Assignment

```
name2 = name1;
```

0x2000 name1

0x2000 name2

29

Example: strassign.c

```
#include <stdio.h>
#include <string.h>
#define MAXLENGTH 5
int main()
{
    char name1[MAXLENGTH] = "Ann";
    char name2[] = "Ann";
    char *name3 = "John";
    char name4[MAXLENGTH];

    printf("\nBEFORE\nname1=%s, name2=%s, name3=%s",
           name1, name2, name3);
    strcpy(name1, "Fred");
    strcpy(name2, "Fred");
    strcpy(name4, name1);
    name3 = name2;
    printf("\nAFTER\nname1=%s, name2=%s, name3=%s, name4=%s",
           name1, name2, name3, name4);

    strcpy(name1, "Jack");
    strcpy(name2, "Jim");
    printf("\nLAST\nname1=%s, name2=%s, name3=%s, name4=%s",
           name1, name2, name3, name4);
    return 0;
}
```

30

String Operation: Concatenation

```
char string1[MAXLENGTH];
char string2[MAXLENGTH];

strcpy(string1, "Goodbye");
strcpy(string2, ", Cruel ");

strcat(string1, string2);
strcat(string1, string2);
strcat(string1, "World!");
```

string1: "Goodbye"
string2: ", Cruel "

31

String Operation: Concatenation (cont)

```
char string1[MAXLENGTH];
char string2[MAXLENGTH];

strcpy(string1, "Goodbye");
strcpy(string2, ", Cruel ");

strcat(string1, string2);
strcat(string1, string2);
strcat(string1, "World!");
```

string1: "Goodbye, Cruel "
string2: ", Cruel "

32

String Operation: Concatenation (cont)

```
char string1[MAXLENGTH];
char string2[MAXLENGTH];

strcpy(string1, "Goodbye");
strcpy(string2, ", Cruel ");

strcat(string1, string2);
strcat(string1, string2);
strcat(string1, "World!");
```

string1: "Goodbye, Cruel , Cruel "
string2: ", Cruel "

33

String Operation: Concatenation (cont)

```
char string1[MAXLENGTH];
char string2[MAXLENGTH];

strcpy(string1, "Goodbye");
strcpy(string2, ", Cruel ");

strcat(string1, string2);
strcat(string1, string2);
strcat(string1, "World!");
```

string1: "Goodbye, Cruel , Cruel World!"
string2: ", Cruel "

34

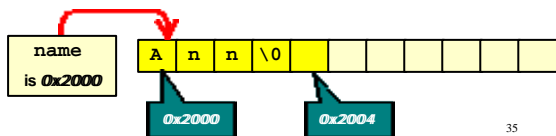


Common Mistake:

Not enough space

```
char name[5];

strcpy(name, "Ann");
strcat(name, " Smith");
```



35

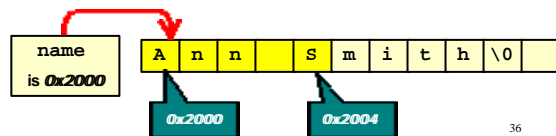


Common Mistake:

Not enough space

```
char name[5];

strcpy(name, "Ann");
strcat(name, " Smith");
```



36

String Operation: Comparison

```
strcpy(string1, "Apple");
strcpy(string2, "Wax");

if (strcmp(string1, string2) < 0)
{
    printf("%s %s\n", string1, string2);
}
else
{
    printf("%s %s\n", string2, string1);
}
```

Returns:
 negative if string1 < string2
 zero if string1 == string2
 positive if string1 > string2

37

String Operation: Comparison (cont)

```
strcpy(string1, "Apple");
strcpy(string2, "Wax");

if (strcmp(string1, string2) < 0)
{
    printf("%s %s\n", string1, string2);
}
else
{
    printf("%s %s\n", string2, string1);
}
```

output: Apple Wax

38



Common Mistake: Wrong Comparison



```
strcpy(string1, "Apple");
strcpy(string2, "Wax");

if (string1 < string2)
{
    printf("%s %s\n", string1, string2);
}
else
{
    printf("%s %s\n", string2, string1);
}
```

39



Caution 1: Not a Boolean

```
strcpy(string1, "Hi Mum");
strcpy(string2, "Hi Mum");

if ( strcmp(string1, string2) )
{
    printf("%s and %s are the same\n",
        string1, string2);
}
```

Returns zero if the strings are the same.

40

String Operation: Length

```
char string1[100];
strcpy(string1, "Apple");
printf("%d\n", strlen(string1));
```

output: 5

Number of char-s before the '\0'

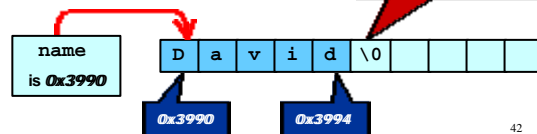
41



Common Mistake: Not enough space

```
char name[5];
strcpy(name, "David");
```

Don't forget the '\0'



42

Character Strings as Parameters

- Strings as formal parameters are declared as `char*` or `char[]`
 - Examples:


```
void Greet ( char* name )
void Greet ( char name[] )
```
- They point to the first element of the string (array of chars)
- Changes to the string inside the function affect the actual string

43

Example: hello3.c

<pre>#include <stdio.h> #include <string.h> #define NAMELEN 50 /* Print a simple greeting to the user */ void Greet (char * name) { strcat(name, "! How are ya?"); }</pre>	<pre>int main() { char user[NAMELEN]; printf("Who are you? "); scanf("%s", user); Greet(user); printf("%s\n", user); return 0; }</pre>
--	--



44

Example: hello3.c (cont)

<pre>#include <stdio.h> #include <string.h> #define NAMELEN 50 /* Print a simple greeting to the user */ void Greet (char * name) { strcat(name, "! How are ya?"); }</pre>	<pre>int main() { char user[NAMELEN]; printf("Who are you? "); scanf("%s", user); Greet(user); printf("%s\n", user); return 0; }</pre>
--	--



45

Example: hello3.c (cont)

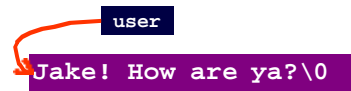
<pre>#include <stdio.h> #include <string.h> #define NAMELEN 50 /* Print a simple greeting to the user */ void Greet (char * name) { strcat(name, "! How are ya?"); }</pre>	<pre>int main() { char user[NAMELEN]; printf("Who are you? "); scanf("%s", user); Greet(user); printf("%s\n", user); return 0; }</pre>
--	--



46

Example: hello3.c (cont)

<pre>#include <stdio.h> #include <string.h> #define NAMELEN 50 /* Print a simple greeting to the user */ void Greet (char * name) { strcat(name, "! How are ya?"); }</pre>	<pre>int main() { char user[NAMELEN]; printf("Who are you? "); scanf("%s", user); Greet(user); printf("%s\n", user); return 0; }</pre>
--	--



47

More of `scanf` demystified

No ampersand (&) in `scanf` with strings!



```
int main()
{
    char user[NAMELEN];
    printf("Who are you? ");
    scanf("%s", user);
    Greet(user);
    printf("%s\n", user);
    return 0;
}
```

48

Summary

- A string is a contiguous array of chars
- The string identifier is the address of the first char in the string
- Individual chars are accessed using the `str[index]` notation
- There are C library functions for copying, concatenating and comparing strings

49

Reading

- King
 - Chapter 13 (except Section 13.6)
- Deitel and Deitel
 - Chapter 8 (8.1 – 8.7)

50