Xi'an Jiaotong-Liverpool University
西交利物浦大学

CPT104 - Operating Systems Concepts

Lab 3

While Loops, Character Strings

While Loop

- In this example, we keep asking input until the user enter the value 6

```
#include <stdio.h>

int main() {

    int diceValue, notSix;
    scanf("%d", &diceValue);
    notSix = diceValue != 6;
    while (notSix) {
        scanf("%d", &diceValue);
        notSix = diceValue != 6;
    }

    return 0;
}
```

While format:

```
while (expression)
    statement
```

while will keep looping until notSix is false

While Loop

- You could also use the expression in while loop condition testing directly

```
#include <stdio.h>

int main() {

    int diceValue;
    scanf("%d", &diceValue);

    while (diceValue != 6) {
        scanf("%d", &diceValue);
    }

    return 0;
}
```

use while loop when you don't know exactly how many times to loop

Example: Guess Number

- A program that print number of guesses needed to guess a secret number:

```
#include <stdio.h>
int main() {
    int secretNum, guess, numGuess = 1;
    scanf("%d", &secretNum);
    scanf("%d", &guess);
    while (guess != secretNum) {
        if(guess > secretNum) {
            printf("guess less\n");
        } else {
            printf("guess more\n");
        }
        numGuess++;
        scanf("%d", &guess);
    }
    printf("number of guesses: %d\n", numGuess);
    return 0;
}
```

try in Codecast

Example: Collecting Signatures

- A program that count how many days needed to get 1000 signatures:

```c
#include <stdio.h>
int main() {
    int signaturesNeeded = 1000;
    int day = 0;
    int newSignatures = 3;
    int totalSignatures = 3;
    while (totalSignatures < 1000) {
        day++;
        newSignatures = 2 * newSignatures;
        printf("Day %d: %d new signatures! ", day, newSignatures);
        totalSignatures = totalSignatures + newSignatures;
        printf("Total: %d\n", totalSignatures);
    }
    return 0;
}
```

try in Codecast

say you start with 3 persons

each new person will ask another 2 persons' sign

repeat until total sign >= 1000

Variants of While and for loop

- Infinite loop

```
for (;;) {
    ...
}
```

- do-while loop

```
do
    statement
while (expression);
```

- exit loop

```
int i = 1;//initializing a local variable
//starting a loop from 1 to 10
for (i = 1;i <= 10;i++) {
  printf("%d \n", i);
  if (i == 5) {//if value of i is equal to 5, it will break the loop
    break;
  }
}//end of for loop
```

It is sometimes convenient to be able to exit from a loop other than by testing at the top or bottom. break statement provides an early exit from for, while, and do

❖ **How to convert for loop to while loop?**

```
for (expr1; expr2; expr3)
    statement
```

Store, Read, Print Array of Characters (String)

- Write a program that read a 3-letter word from user and then print it

```c
#include <stdio.h>

int main() {

    char word[4];
    printf("Enter a word with 3 letters: ");
    scanf("%s", word);
    printf("The word is: %s.\n", word);

    return 0;
}
```

unlike Java, C does not have a string data structure, but we use array of char

to store 3 chars, we set it to 4 because the last char is the null terminator '\0' to tell C where the array of char ends

scanf with format character %s will automatically add the null terminator at the end

Store, Read, Print Array of Characters (String)

- Write a program that read 3-letter word from user and then print it

```c
#include <stdio.h>

int main() {

    char word[4];
    printf("Enter a word with 3 letters: ");
    scanf("%s", word);
    printf("The word is: %s.\n", word);

    return 0;
}
```

notice that we do **not** use & in front of the variable word to store the %s value
we will explain in future lab

to print, we use with printf also with format character %s
printf will stop printing at '\0'

now run this on Codecast and enter a 3-letter word

Visualize Array of Characters

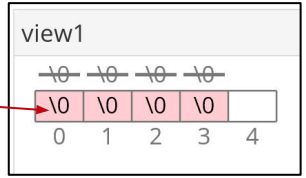
- Write a program that read 3-letter word from user and then print it

```c
#include <stdio.h>

int main() {

    //! showArray(word, cursors[i])
    char word[4];
    printf("Enter a word with 3 letters: ");
    scanf("%s", word);
    printf("The word is: %s.\n", word);

    return 0;
}
```

add this visualization command in your code, compile in Codecast, run step by step, by clicking step into

you will see how your input is stored in the memory

note that only in Codecast, the array has been initialized to '\0'
it is **not** so in general C compiler, when word is only declared



Initialize Char Array and Printing Null

- Let us try to print the null terminator, what do you see?

```c
#include <stdio.h>

int main() {

    char word[4] = "fun";
    printf("The word is: %s.\n", word);
    printf("The characters are: <%c> <%c> <%c> <%c>\n",
            word[0], word[1], word[2], word[3]);

    return 0;
}
```

initialize to the word fun

just an arbitrary delimiter to separate the printed chars

you can print the letter like the usual element of an array

now run this on Codecast
what gets printed in console ?

Change A Letter

- We can change a letter just like array assignment

```c
#include <stdio.h>

int main() {

    char word[4] = "fun";
    printf("The word is: %s.\n", word);
    word[1] = 'a';
    printf("The word now is: %s.\n", word);

    return 0;
}
```

"fun" is an example of a *string literal* in C
it is terminated with null terminator '\0'

assign a new character

Characters vs String Literals

- String literals may contain as few as one or even zero characters
- Do not confuse a single-character string literal, e.g. "a" with a single character, 'a':
  - "a" is actually two characters, because of the null terminator stored at the end
  - 'a' is just one character

- An empty string "" consists of only the null terminator,
  and is considered to have a string length of zero,
  because the null terminator does not count when determining string lengths

Two Strings and Null Terminator (1)

- Printf will stop printing when it encounter the null terminator

```
#include <stdio.h>

int main() {
    //! word1 = showArray(word1, cursors=[i], width=0.5)
    //! word2 = showArray(word2, cursors=[i], width=0.5)
    char word1[5];
    char word2[8];
    scanf("%s %s", word1, word2);
    printf("%s %s\n", word1, word2);
    return 0;
}
```

run in Codecast with Step Into
enter words *good morning*

you will see two arrays of characters
with null terminator automatically added

Two Strings and Null Terminator (2)

- Printf will stop printing when it encounter the null terminator

```c
#include <stdio.h>

int main() {
    //! word1 = showArray(word1, cursors=[i], width=0.5)
    //! word2 = showArray(word2, cursors=[i], width=0.5)
    char word1[5];
    char word2[8];
    scanf("%s %s", word1, word2);
    word1[3] = '\0';
    word2[2] = '\0';
    printf("%s %s\n", word1, word2);
    return 0;
}
```

now add these two lines of codes

compile, step into, and
enter *good morning* again in terminal
what gets printed this time?

Find the Length of a String

- How do we find the length of a string, read from a user?

```c
#include <stdio.h>
int main() {
    char word[30];
    int i = 0;
    printf("Please enter a word: ");
    scanf("%s", word);
    while(word[i] != '\0') {
        i++;
    }
    printf("%s has length %d.\n", word, i);
    return 0;
}
```

by using while and null terminator!

keep incrementing counter, until we find a null terminator

run in Codecast with your own input

Find the Length of a String - Visualized

- How do we find the length of a string, read from a user?

```
#include <stdio.h>
int main() {
    //! showArray(word, cursors=[i])
    char word[30];
    int i = 0;
    printf("Please enter a word: ");
    scanf("%s", word);
    while(word[i] != '\0') {
        i++;
    }
    printf("%s has length %d.\n", word, i);
    return 0;
}
```

run in Codecast using Step Into

Thank you for your attention !

- In this lab, you have learned:
  - While loops
  - String in C which is an array of characters
    - terminated with null terminator '\0'
  - Format Specifier %s
  - String literals
  - Finding length of strings

- **For more information:**

✓ **while** : refer to chapter 3, 3.5, 3.6, (for advanced application 3.7, 3.8)

✓ **characters**: refer to 1.9, 2.3, 7.2,7.4