目录

Week 3 Exercises

sum3str.c

Assume you have the following string declarations:

切换行号显示

```c
char *r = "1";
char *s = "23";
char *t = "456";
```

Write a short program that uses the *sscanf()* function to read these 3 strings, sums them as if they were integers, and then prints the total on *stdout*. (Remember a *scanf()* returns an *int*, which is the number of arguments it has read.)

sum3argA.c

Write a short program that uses *sscanf()* to read 3 strings from the command line. If the arguments are all numerical, the program should sum them, and print the total.

- *What happens if one or more of the arguments is not an integer?* For example, try **./sum3argA 1 z 3**

sum3argB.c

Modify **sum3argA.c** to now use the *stdlib* function *atoi()* instead of *sscanf()*. The program should also print the total of the 3 arguments.

- *What happens if one or more of the arguments is not a number?* For example, try **./sum3argB 1 z 3**

postfixStack.txt

Consider postfix expressions consisting of integer operands and the operators + and *.

We saw in lectures that evaluating such a postfix expression using a stack involves *pushing* operands onto the (top of the) stack, and whenever an operator is encountered, *popping* the top 2 elements off the stack, performing the operation, and *pushing* the result back onto the stack. This process is repeated until there is a single operand left on the stack (and there is no more input). This operand is the result of the postfix expression. So for example, given the expression "1 2 +", the contents of the stack after each *push* action is

```
<<1>>
<<2,1>>
<<3>>
```

where left-to-right is considered top-to-bottom.

Show the contents of the stack after each *push* for the postfix expression "1 2 3 + 4 5 * + 6 * +".

postfixQueue.txt

If a queue is used instead of a stack, then instead of pushing onto the stack, operands and results are *qushed* onto the queue (the bottom of the data structure). Evaluate the postfix expression from the previous exercise using a queue, showing the contents after each *qush* action.

qush.c

- Implement the *qush* operation in the *quack* ADT described in Week 3's lecture.
- Test it by compiling it with the client *Josephus.c*.

matcher.c

A stack can be used to check that opening and closing brackets: '(' and ')', '[' and ']', '{' and '}', are balanced in given text. For example, `(a[i]+b[j])*c[k]` is balanced, and so is

```
void f(char a[], int n) {int i; for(i=0;i<n;i++) a[i] =
(a[i]*a[i]*a[i])*(i+1); return;}
```

Mismatches can occur for for 3 reasons:

A. mixed kinds, e.g. (a+1}
B. missing opening, e.g. a[1])
C. missing closing, e.g. ((a+1)

A basic algorithm to match brackets (*(), [] and {}*) can use a stack, in the following way:

- if you read an opening bracket, push it onto the stack
- if you read a closing bracket, pop the stack, and ...
    - ... compare that character with the closing bracket
        - if not the same, then brackets are mismatched // case A above
    - if the pop generates underflow then brackets are mismatched // case B

- after all characters have been read:
  - if the stack is not empty, then mismatched // case C

In this exercise,

1. Using the *quack* ADT, write a program that reads from *stdin*, and reports a mismatch if an opening bracket '(','[' or '{' is not correctly matched with a closing bracket ')', ']' or '}' (resp.).

   If all the brackets match, then the program generates no output, otherwise it prints the string *mismatch detected*. So for example if the file *data.inp* contains the 'bad' text `({[ })]` , then

   ```
   prompt$ ./matcher < data.inp
   mismatch detected
   ```

   or you can test using a pipe for example

   ```
   prompt$ echo "(())" | ./matcher
   prompt$
   ```

   Also try the following test cases as well as any you create yourself:

   ```
   {[{{}{}(([]([](((()([])({}))))))()()())}{}]}
   ```

   and

   ```
   {[{{}{}(([]([](((()([])({})))))))()()())}{}]}
   ```

2. Provide a fragment of C code that contains no syntax errors, but would fail your *matcher* program.

Week3Exercises (2019-06-19 15:55:23由AlbertNymeyer编辑)