目录

Week 4 Exercises

pointers.txt

Given the definition

切换行号显示

```
1 int nums[12] = {5,3,6,2,7,4,9,1,8};
```

and assuming that *&nums[0]* is *0xbabeface*, what are the values of the following expressions?

a. *nums + 4*
b. **nums + 4*
c. **(nums + 4)*
d. *nums[4]*
e. **(nums + *(nums + 3))*
f. *nums[nums[2]]*

structs.txt

Consider the following fragment of code:

切换行号显示

```
 1 typedef struct {
 2    int   studentID;
 3    int   age;
 4    char  gender;
 5    float WAM;
 6 } Person;
 7
 8 Person per1;
 9 Person per2;
10 Person *ptr;
11
12 ptr = &per1;
13 per1.studentID = 3141592;
14 ptr->gender = 'M';
15 ptr = &per2;
16 ptr->studentID = 2718281;
17 ptr->gender = 'F';
18 per1.age = 25;
19 per2.age = 24;
```

```
20 ptr = &per1;
21 per2.WAM = 86.0;
22 ptr->WAM = 72.625;
```

What are the values of the fields in *per1* and *per2* after execution of the above statements?

qush.c

- Implement the *qush* operation in the *quackLL* ADT described in Week 4's lecture.
- Test it by compiling it with the client *Josephus.c*.

base2.c

A stack can be used to convert a positive number *n* base 10 (i.e. a decimal number) to a number base *m* using the following algorithm:

```
  while n>0 do
      push n%m onto the stack
      n = n / m
  end while
  and then popping the numbers off the stack until it is empty
```

The resulting series of digits is the representation of the number *n* in base *m*.

In this exercise you should assume *m=2*, so you are converting a decimal number into a binary number only.

For example, applying the algorithm to *n=13*, we find:

```
start of loop
  push 13%2 ==> 1 onto the stack
  n =  13/2
  push 6%2  ==> 0 onto the stack
  n =  6/2
  push 3%2  ==> 1 onto the stack
  n =  3/2
  push 1%2  ==> 1 onto the stack
  n =  1/2
loop terminates
Popping yields 1101
```

and of course 13 is 1101 in binary.

Implement this algorithm using a quack ADT. Your program should read the number *n* from the command line, and print the error message

```
Usage: ./base2 number
```

if there is not exactly 1 numerical argument (≥0) on the command line.

For example:

```
prompt$ dcc -o base2 quack.c base2.c
prompt$ ./base2 0
0

prompt$ ./base2 1
1
```

```
prompt$ ./base2 127
1111111

prompt$ ./base2 2730
101010101010

prompt$ ./base2 x
Usage: ./base2 number

prompt$ ./base2 -1
Usage: ./base2 number

prompt$ ./base2 4 2
Usage: ./base2 number
```

base.c

Extend the previous program to handle any base *m* between 2 and 16. So, for example:

```
2730 in base 2 is 101010101010
2730 in base 3 is 10202010
2730 in base 8 is 5252
2730 in base 12 is 16b6
2730 in base 14 is dd0
2730 in base 15 is c20
2730 in base 16 is aaa
```

where in the output 'a' represents *10*, 'b' represents *11*, 'c' represents *12* etc. Checking some of the results, we note that *2730* equals $1*12^3 + 6*12^2 + 11*12 + 6$ confirming the base 12 result *16b6*, and equals $10*16^2 + 10*16 + 10$ confirming the base 16 result *aaa*.

Your program should read the 2 numbers *n* and *m* from the command line, and print the error message

```
Usage: ./base number base
where 2<=base<=16
```

if there are not exactly 2 numerical arguments ($n \geq 0$ and $m \geq 2$) on the command line.

For example:

```
prompt$ ./base 0 2
0

prompt$ ./base 2730 2
101010101010

prompt$ ./base 2730 12
16b6

prompt$ ./base 2730 16
aaa

prompt$ ./base 1234
Usage: ./base number base
where 2<=base<=16

prompt$ ./base -1 2
Usage: ./base number base
where 2<=base<=16
```

Test your program with all the test cases above, and any more you think useful. As well, what is:

- 51966 expressed as a hexadecimal number?
- 19006 expressed as a duodecimal number?

prefixes.c

Write a C-program that takes 1 command line argument and prints all its prefixes in decreasing order of length.

- You are not permitted to use any library functions other than *printf()*
- You are not permitted to use any array other than *argv[]*

An example of the program executing is

```
./prefixes Programming
Programming
Programmin
Programmi
Programm
Program
Progra
Progr
Prog
Pro
Pr
P
```

Week4Exercises (2019-06-26 09:48:41由AlbertNymeyer编辑)