**LAB [2], [01/17/2019] MCS 253P**

**Name:Yu Qin, yuq8@uci.edu**

**Partner: Meng ke, jiexu li(jiexunl1@uci.edu)**

**General Problem Description**

1. function **atoi** (stands for “ascii” to “integer”)

to convert a c-string of base-10 digits

to a signed 32 bit decimal number

1. function **itoa**

to convert a signed 32 bit decimal number

into a c-string of digits

main program, **testIntegers**, to read a series of lines as c-strings from standard input (using fgets()) then convert the string (e.g., stored in char s[100];) into a signed integer (e.g., stored in int value;) using your atoi function, then convert the integer back to a string using itoa then print the c-string to the standard output.  If the generated string does not equal the input string (using **strcmp**()==0 for equality), print an appropriate error message to stderr.

**Additional Problem Specifics**

Valid number format in input? Like no char, in decimal…

Valid number range in input? In INT\_32bits: -2147483647..2147483647

Int without point decimals?

Only one number in file? Or do in loop?

Empty line?: since make no sense, suppose valid.

**Sample Input**

12345

-12345

-1

1

0

2111111111

-2111111111

**Proposed Algorithm**

***Description:***

**atoi:**

1. **getline() to get the string, get the size;**
2. **use high = size-1, and low = start = 0 to treat the string;**
3. **judge the first char whether ‘-‘(negative number), flag = -1, low = 1;**
4. **if valid:**

**do ifrom the last digit of the low to high, i--: use the loop of res2int = res2int\*10 + s[i] –‘0’;**

**itoa:**

1. **get the number = input\_int ;**
2. **use low = start = 0 to treat the string;**
3. **judge whether the number is negative for the first ‘-‘ , low = 1;**
4. **//do from the last integer to put the low position of array, and after done, swap the array between low with high;**

**int high = low;**

**while (input\_int!=0) {**

**res2asci10[high] = input\_int %10 – 0 + ‘0’;**

**input\_int /=10;**

**high++;**

**}**

1. **void swap(res, low, high){}, with resize in it just by adding ‘\0’;**

**testIntegers:**

***Correctness:***

**atoi:**

**Right in logic, of transferring the format of array in string to int: consider the negative, and the digits one by one.**

**itoa:**

**Right in logic, of transferring a int to string format: consider the negative, and the digits one by one.**

***Time Complexity:***

**atoi: O(n)**

**itoa: O(n)**

***Space Complexity:***

**atoi: O(1)**

**itoa: O(1)**

**C++ Implementation of Algorithm**

#include <iostream>

#include <fstream>

#include <stdio.h>

#include "string.h"

using namespace std;

int atoi(char\* input\_string){

int high = strlen(input\_string)-1, low = 0;

int flag = 1;

if (input\_string[0] == '-') {

flag = -1;

low = 1;

}

int res2int = 0;

for (int i = low; i <= high; i++){

res2int = res2int\*10 + input\_string[i] - '0';

}

return res2int\*flag;

}

void swap(char\* res2asc, int low, int high){

while(low<=high){

char tmp = res2asc[low];

res2asc[low] = res2asc[high];

res2asc[high] = tmp;

low++; high--;

}

}

void itoa(int input\_int, char\* res2asc){

if (input\_int==0){

strcpy(res2asc,"0");

printf("%s\n",res2asc);

return;

}

int low = 0;

if (input\_int < 0 ){

low = 1;

res2asc[0] = '-';

input\_int = -input\_int;

}

int high = low;

while (input\_int!=0) {

res2asc[high] = input\_int%10 + '0';

input\_int/=10;

high++;

}

high--;

res2asc[high+1] = '\0';

swap(res2asc,low,high);

printf("%s\n",res2asc);

}

void MyTestIntegers(char\* input\_string){

int atoi\_int = atoi(input\_string);

char itoa\_string[20];

itoa(atoi\_int, itoa\_string);

if (strcmp(itoa\_string, input\_string) != 0)

fprintf(stderr,"Wrong cast!input:%s,output\_int:%d,output\_str:%s|\n",input\_string,atoi\_int,itoa\_string);

}

int main(int argc, char\*\* argv){

//open file

if (argc < 2){

cout<<"You choose to input number manually: \n";

char input\_string[20];

cin >> input\_string;

MyTestIntegers(input\_string);

}

else if (argc == 2){

char input\_string[20];

ifstream infile(argv[1]);

if ( infile.is\_open() ){

while ( infile.getline(input\_string,20) ){

MyTestIntegers(input\_string);

}

infile.close();

}

else cout<<"Cannot open the file.\n";

}

else cout<<"Your input file name should not have space. \n";

}

**Advantages/Disadvantages of Your Algorithm and Any Other Comments**

Advantange: intuitively to think and understand

Disadvantage: some process is complex

**Test Cases**

* sample description of a test case
  + output we expect (want)
  + output our algorithm produces

input expect produce result

* 12345 ------------ 12345-------------12345
* -12345------------ -12345------------- -12345
* -1, -1, -1
* 1, 1, 1
* 0,0, 0
* 2111111111,same
* -2111111111,same

**Screenshot of Compilation and Execution of Program Under Valgrind**

**Important Notes (you may erase these notes when pasting your screenshots):**

* **Screenshots should demonstrate your program handling each of your proposed test cases. You may need to create your own custom inputs to demonstrate your test cases.**
* **Your program outputs should be CLEARLY LABELED on the terminal so that you (or any colleague that runs your program) can easily understand each of the following:**
  + **the details of the test case being tested**
  + **what your program’s output is for that test case**
  + **whether your program correctly gave the desired output or not**
    - **one way is to provide the expected output to use as comparison**
    - **or you can use your own method, as long as the person running your program can easily understand from the output whether the program worked as expected or not**

**Since your screenshots capture your program output, this information should also be clearly visible within your screenshots!**

* **To keep things simple, please address all your test cases within a single run of your program.**
  + **one way to do this by using functions to facilitate the testing of the different test cases**
  + **as a final test case, don’t forget to include running your program with the test-case that is provided to you by the instructional staff**
* **Screenshots for each run of your program should start with a screen shot that includes your command to run the program under Valgrind and end with a screenshot that shows the final memory and error report.**
* **For additional notes, please see the Dr. Klefstad’s HW submission guide.**