# Day 1 （Go）

package main

import "fmt"

func main() {

fmt.Printf("%v\n", (9.5\*4.5-2.5\*3)/(45.5-3.5))

}

# Day2

## 1.

int pstr\_scan(char\* str, int size)

{

char ch;

int length = 0;

ch=getchar();

while (length<size && ch != ' ' && ch != '\n')

{

str[length++] = ch;

ch=getchar();

}

return length;

}

void pstr\_print(const char\* str, int length)

{

while (length--)

{

putchar(str[0]);

str++;

}

}

## 2.

int pstr\_cmp(const char \*s1,int len1,const char \*s2,int len2)

{

int i=0;

while (i<len1&&i<len2)

{

if (s1[i] != s2[i]) return s1[i]-s2[i];

i++;

}

if (i < len1) return s1[i];

if (i < len2) return -s2[i];

return 0;

}

## 来自CSDN

int pstr\_cpy(char \*s1, int len1, int size, const char \*s2, int len2)

{

int i;

for(i=0;i<size&&i!=len2;i++)

{

s1[i]=s2[i];

}

return i;

}

int pstr\_cat(char \*s1, int len1, int size, const char \*s2, int len2)

{

int i;

for(i=len1;i<size&&i<len1+len2;i++)

{

s1[i]=s2[i-len1];

}

return i;

}

# （Go）

package main

import (

"fmt"

"os"

"unicode"

)

func main() {

var exp string

var numStk []int

var opStk []byte

var num int

\_, err := fmt.Scanln(&exp)

if err != nil {

panic(err)

}

for \_, ch := range exp {

switch unicode.IsDigit(ch) {

case true:

num = num\*10 + int(ch) - '0'

break

case false:

numStk = append(numStk, num)

num = 0

opStk = append(opStk, byte(ch))

}

}

for i, ch := range opStk {

switch ch {

case '+':

numStk[i+1] = numStk[i] + numStk[i+1]

break

case '-':

numStk[i+1] = numStk[i] - numStk[i+1]

break

case '\*':

numStk[i+1] = numStk[i] \* numStk[i+1]

break

case '/':

if numStk[i+1] == 0 {

fmt.Println("ERROR")

os.Exit(0)

}

numStk[i+1] = numStk[i] / numStk[i+1]

break

case '=':

break

default:

fmt.Println("ERROR")

os.Exit(0)

}

}

fmt.Println(numStk[len(numStk)-1])

}

Day 4

#include <iostream>

using namespace std;

static void verify\_failed(const char\* file, int line, const char\* expr){

// TODO: 仿照样例格式，在验证失败时输出位置和表达式。

printf("verify failed at %s:%d: %s\n",file, line, expr);

}

///asdf

#define verify(x) if(x){}else verify\_failed(\_\_FILE\_\_, \_\_LINE\_\_, #x)

int main(){

int m, n;

cin >> m >> n;

for(int i = 0, x; i < n; ++i){

cin >> x;

cout << x << endl;

// TODO: 用#line预处理指令设置行号（整数）和源文件名称（字符串），以便下行报出预期的错误信息;

#line 987654321 "unknown.cpp"

verify(x < m);

}

}

Day5

#include<stdio.h>

#include<math.h>

typedef struct Node

{

int date;

int next;

}Node;

int first[100005]; // 存放第一条链表的地址

int f = 0;;

int isVisited[100005]; // 判断是否有重复，0代表未重复，1代表已存在

int is = 0;

int last[100005]; // 存放第二条链表的地址

int l = 0;

void print(Node arr[],int g[],int n);

int main()

{

Node arr[100005];

int head; // 首地址

int n;

int ads;

scanf("%d %d",&head,&n);

for(int i=0;i<n;i++)

{

scanf("%d",&ads);

scanf("%d %d",&arr[ads].date,&arr[ads].next);

}

int p = head;

while(p != -1)

{

is = abs(arr[p].date); // 取出键值的绝对值

// 判断该键值是否存在

if(!isVisited[is])

{

// 该键值不存在

first[f++] = p; // 将该键值的地址放入第一条链表中

isVisited[is] = 1; // 将该键值位置置为1，表示已存在

}

else

{

// 该键值存在

last[l++] = p; // 将该键值地址放入第二条链表中

}

p = arr[p].next; // 移向下一位置

}

print(arr,first,f);

print(arr,last,l);

return 0;

}

void print(Node arr[],int g[],int n)

{

for(int i=0;i<n;i++)

{

if(i == n-1)

{

printf("%05d %d -1\n",g[i],arr[g[i]].date);

}

else

{

printf("%05d %d %05d\n",g[i],arr[g[i]].date,g[i+1]);

}

}

}