《数据结构》第一次作业

# 作业一：编程获取运算频次。

要求：运行给定代码，获取循环次数。

程序代码：

// 1

for (sum=0, i=0; i<N; i++) sum ++;

// 2

for (sum=0, i=0; i<N; i++)   
 for (j=0; j<N\*N; j++) sum ++;

// 3

for (sum=0, i=0; i<N; i++)  
 for (j=0; j<i; j++) sum++;

// 4

for (sum=0, i=0; i<N; i++)  
 for (j=0; j<i\*i; j++)  
 for (k=0; k<j; k++) sum++;

// 5

for (sum=0, i=1; i<=N; i++)  
 for (j=1; j<i\*i; j++) if(j%i==0)  
 for (k=0; k<j; k++) sum++;

## 程序运行结果

1:

2:

3:

4:

5:

# 作业二：编程感受算法时间复杂度

根据绪论部分求连续最大子序列和的示例，编程实现之，并绘制运行时间与数据规模函数图。(给定n个有序整形数据序列，试在其中找出连续k个数据之和最大的子序列)

要求：给出当N分别等103，104，105，106时算法运行需要的时间。数据采用随机生成[-100, 100]之间的整数进行运算。

思考题：当N取值为多大时，运行时间落在1~2分钟之间？

## 算法运算结果表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 算法 | N取不同值时算法运行时间（秒） | | | | 运行时间在1~2分钟之间的N值及运行时间 |
| 103 | 104 | 105 | 106 |
| 算法1 | 0.5 | / | / | / | N:5000  时间：132.628 |
| 算法2 | 0.002 | 0.147 | 20.468 | / | N:2e5  时间：117.786 |
| 算法3 | 0 | 0 | 0.001 | 0.003 | N:溢出  时间：/ |

## 主函数算法代码：

#include<iostream>

using namespace std;

#include<stdio.h>

#include<time.h>

#include<stdlib.h>

#include<math.h>

#define N 5000 //数组规模大于10^6时要设为全局变量

int take\_time(void (\*process)(int \*),int a[]){

    float delay=0;

    clock\_t t0;

    srand(time(0));

    t0=clock();

    process(a);

    delay=clock()-t0;

    return delay;

}

int main()

{

    int a[N];

    int i;

    srand(time(0));

    for(i=0;i<N;i++){//随机生成数据

        a[i]=rand()%201-100;

    }

    cout<<take\_time(N3,a)<<'\n';

    cout<<take\_time(N2,a)<<'\n';

    cout<<take\_time(N1,a);

    return 0;

}

## 算法一：O(N3)算法函数代码

void N3(int a[]){

    int i,j,k;

    int max=0x80000000;

    int sum=0;

    for(i=0;i<N;i++){

        for(j=i;j<N;j++){

            sum=0;

            for(k=i;k<=j;k++){

                sum+=a[k];

            }

            if(sum>max) max=sum;

        }

    }

    cout<<max<<'\n';

}

## 算法二：O(N2)算法函数代码

void N2(int a[]){

    int i,j;

    int max=0x80000000;

    int sum=0;

    for(i=0;i<N;i++){

        sum=0;

        for(j=i;j<N;j++){

            sum+=a[j];

            if(sum>max) max=sum;

        }

    }

    cout<<max<<'\n';

}

## 算法三：O(N)算法函数代码

void N1(int a[]){

    int i;

    int max=0x80000000;//已扫过的最大子序列和

    int sum=0;//当前的序列和

    for(i=0;i<N;i++){

        sum+=a[i];

        if(sum>max) max=sum;

        else if(sum<=0) sum=0;//如果当前序列和<=0，直接更新序列

        //假设最大和子序列的前半部分<=0，直接取后半部分，一定是更大的

    }

    cout<<max<<'\n';

}

# 作业三：有理数的四则运算

从键盘输入四个整数a,b,c,d，分别表示两个有理数x=a/b, y=c/d，计算x+y, x-y, x\*y与x/y的值（注意化简成最简式子）。

要求1：用浮点数输出x与y的值；

要求2：输出用a/b形式表示的x+y, x-y, x\*y与x/y的准确值；

要求3：输出x+y, x-y, x\*y与x/y的浮点数的值，并验证与上一步的误差。

要求4：随机生成1000组值，计算平均误差。

要求5：设计误差传递模型，使得计算误差在10-6以上。

## 测试结果1

1. 输入数据（四个整数）：a=1283,b=9834,c=1230,d=9862
2. 输出两个实数的表示（最简形式）：x=1283/9864,y=615/4931
3. 输出两个实数的float结果：x=0.130466,y=0.124721
4. 两种表示法输出误差：epsx=0.000000000112,epsy=0.000000002674
5. 四则运算整数表示结果：x+y=12374383/48491454,x-y=278563/48491454,x\*y=263015/16163818,x/y=6326473/6047910
6. 四则运算浮点数表示结果：x+y= 0.255187,x-y= 0.005745,x\*y= 0.016272,x/y= 1.046059
7. 四则运算两种表示法的误差：eps(x+y)=0.000000002562, eps(x-y)=0.000000000458, eps(x\*y)=0.000000000785, eps(x/y)= 0.000000012457

## 测试结果2

1. 输入数据（四个整数）：a=3,b=6,c=60,d=90
2. 输出两个实数的表示（最简形式）：x=1/2,y=2/3
3. 输出两个实数的float结果：x=0.500000,y=0.666667
4. 两种表示法输出误差：epsx=0.000000000000,epsy=0.000000019868
5. 四则运算整数表示结果：x+y=7/6, x-y=-1/6, x\*y=1/3, x/y=3/4
6. 四则运算浮点数表示结果：x+y=1.166667, x-y=-0.166667, x\*y=0.333333, x/y= 0.750000
7. 四则运算两种表示法的误差：eps(x+y)= 0.000000039736, eps(x-y)= 0.000000004967, eps(x\*y)= 0.000000009934, eps(x/y)= 0.000000000000

## 测试结果3

1. 输入数据（四个整数）：a=6613,b=20353,c=12403,d=12563
2. 输出两个实数的表示（最简形式）：x= 6613/20353,y= 12403/12563
3. 输出两个实数的float结果：x= 0.324915,y= 0.987264
4. 两种表示法输出误差：epsx= 0.000000014167,epsy=0.000000027456
5. 四则运算整数表示结果：x+y=335517378/255694739, x-y=-169359140/255694739, x\*y=4824767/15040867, x/y=83079119/252438259
6. 四则运算浮点数表示结果：x+y=1.312179, x-y=-0.662349, x\*y=0.320777, x/y= 0.329107
7. 四则运算两种表示法的误差：eps(x+y)= 0.000000011821, eps(x-y)= 0.000000016513, eps(x\*y)= 0.000000008770, eps(x/y)= 0.000000001862

## 程序代码

#include<iostream>

using namespace std;

#include<stdio.h>

#include<time.h>

#define N 2

struct frac{

    int num;

    int den;

};

int GCD(int a, int b){

    a=abs(a);

    b=abs(b);

    if(a<b){

        a+=b;

        b=a-b;

        a-=b;

    }

    int t;

    while(a%b!=0){

        t=b;

        b=a%b;

        a=t;

    }

    return b;

}

frac reduction(frac \*a){

    frac ans;

    int p=GCD(a->num,a->den);

    ans.num=a->num/p;

    ans.den=a->den/p;

    if(ans.den<0){

        ans.num=-ans.num;

        ans.den=-ans.den;

    }

    return ans;

}

frac fracadd(frac \*a, frac \*b){

    frac ans;

    ans.den=a->den\*b->den;

    ans.num=a->num\*b->den+a->den\*b->num;

    return ans;

}

frac fracminus(frac \*a, frac \*b){

    frac ans;

    ans.den=a->den\*b->den;

    ans.num=a->num\*b->den-a->den\*b->num;

    return ans;

}

frac fracmultiple(frac \*a, frac \*b){

    frac ans;

    ans.den=a->den\*b->den;

    ans.num=a->num\*b->num;

    return ans;

}

frac fracdivide(frac \*a, frac \*b){

    frac ans;

    ans.den=a->den\*b->num;

    ans.num=a->num\*b->den;

    return ans;

}

float frac\_to\_float(frac \*a){

    return(float(a->num)/float(a->den));

}

double frac\_to\_double(frac \*a){

    return(double(a->num)/double(a->den));

}

int main()

{

    int a,b=0,c,d=0;

    frac x,y;

    float fx,fy;

    double epsx,epsy;

    frac ans;

    float fans;

    double epssum=0;

    double epsave;

    for(int i=0;i<1000;i++){

        srand(time(0));

        while(b==0||d==0){

            a=rand();

            b=rand();

            c=rand();

            d=rand();

        }

        x.num=a;

        x.den=b;

        y.num=c;

        y.den=d;

        x=reduction(&x);

        y=reduction(&y);

        fx=frac\_to\_float(&x);

        fy=frac\_to\_float(&y);

        epsx=frac\_to\_double(&x)-fx;

        epsy=frac\_to\_double(&y)-fy;

        if(epsx<0) epsx=-epsx;

        if(epsy<0) epsy=-epsy;

        epssum+=epsx+epsy;

    }

    epsave=epssum/1000/2;

    printf("平均误差：%.12lf",epsave);

    return 0;

}

# 附录1：C语言获取系统时间示例代码

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

void take\_time();

**int** main()

{

**clock\_t** t0; // 用于记录当下时钟时间

**float** delay = 0;

// 初始化随机数，可以保证每次运行时数据都不一致

// 注意该函数只需要用一次，用多了反而出问题

**srand(time(0));**

t0 = clock(); // 记录时间

take\_time();

delay = clock() - t0; // 记录并计算时差

printf("运行时间：%f 秒\n", delay);

}

// 耗时函数，或需要测试运行时间的函数

**void** take\_time()

{

**for** (**int** k = 0; k<20; k++) {

**double** x, y;

**for** (**int** i=0; i<10000000; i++) {

y = (**rand()**%10000)\*0.01;

x += y;

}

}

}