实验一 预备实验

**一、实验目的**

（1）了解抽象数据类型（ADT）的基本概念和及描述方法

（2）熟悉C/C++语言语法及程序设计，为以后章节的学习打下基础。

（3）熟悉使用软酷网

二、实验相关知识

（1）C/C++语言程序设计基础

（2）抽象数据类型定义

**三、实验题目**

（1）编写算法，输入整数n(1<=n<=1000)，x(1<=x<=1000)和ai（i=0,1,2,3，…n）的值，求一元多项式的值，并计算法的时间复杂度。(假设多项式每一项的值以及和均不超过int型的范围)

测试用例：

|  |  |  |  |
| --- | --- | --- | --- |
| 输入 | 2 1 2 3 4 | 3 2 -1 -2 0 5 | 4 2 2 1 0 0 0 |
| 输出 | 9 | 35 | 4 |

【实现要求】算法中不能使用求幂函数。

（2）实现以下复数抽象数据类型，并编写程序实现计算两复数的和、差、积。

复数ADT：

ADT complex{

数据对象：D={c1，c2|c1，c2∈float}

基本操作：创建一个复数 creat(a);

输出一个复数 output(a);

求两个复数的和 add(a,b);

求两个复数的差 sub(a,b);

求两个复数的积 mul(a,b);

}ADT complex;

输入a,b,c,d，输出两复数a+bi 和c+ di的和差积。

测试用例：

|  |  |  |
| --- | --- | --- |
| 输入 | 1 2 3 4 | -1 2 1 1 |
| 输出 | 4+6i  -2-2i  -5+10i | 0+3i  -2+1i  -1+3i |

【实现要求】必须以实现复数ADT中的各个操作的函数来完成程序设计。

**三、程序代码及运行结果**

（1）【程序代码】

#include<iostream>

#include<stdio.h>

#pragma warning(disable:4996)

using namespace std;

int main()

{

int n, x, a[1001];

int sum = 0;

int timer = 0;

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

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

{

scanf("%d", &a[i]);

for (int j = 0; i>j; j++)

{

a[i] = a[i] \* x;

timer = timer++;

}

sum = sum + a[i];

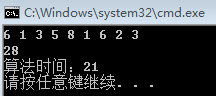
}

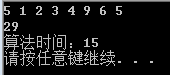
printf("%d \n", sum);

printf("算法时间：%d \n", timer);

}

【运行结果】





【算法时间复杂度分析】

在一个循环里面使用了一个循环，时间复杂度为T（n）=O（n）。

（2）【程序代码】

#include<iostream>

#include<stdio.h>

#pragma warning(disable:4996)

using namespace std;

typedef struct ADT

{

int x, y;

}complex;

complex a, b, a1, b1;

void creat(complex \*comp)

{

int comp1, comp2;

scanf("%d%d", &comp1, &comp2);

(\*comp).x = comp1; comp->y = comp2;

}

void outputc(complex c)

{

printf("\n %d+%di\n ", c.x, c.y);

}

void add(complex one, complex two)

{

complex sum;

sum.x = one.x + two.x;

sum.y = one.y + two.y;

if (sum.y<0)

{

printf("\n %d%di\n ", sum.x, sum.y);

}

else

{

printf("\n %d+%di\n ", sum.x, sum.y);

}

}

void sub(complex one, complex two)

{

complex sum;

sum.x = one.x - two.x;

sum.y = one.y - two.y;

if (sum.y<0)

{

printf("\n %d%di\n ", sum.x, sum.y);

}

else

{

printf("\n %d+%di\n ", sum.x, sum.y);

}

}

void mul(complex one, complex two)

{

complex sum;

sum.x = one.x\*two.x - one.y \* two.y;

sum.y = one.y\*two.x + one.x\*two.y;

if (sum.y<0)

{

printf("\n %d%di\n ", sum.x, sum.y);

}

else

{

printf("\n %d+%di\n ", sum.x, sum.y);

}

}

int main()

{

creat(&a);

creat(&b);

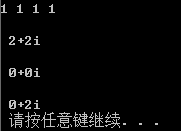
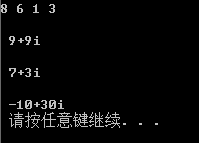
add(a, b);

sub(a, b);

mul(a, b);

}

【运行结果】



**四、实验心得体会**

巩固了上学期的C++知识，并使用了之前未用过的printf减少代码运行时间，提高代码效率。计算机运算也需要时间，所以我们学习算法尽量减少运行所需的时间提高效率。