实验报告 01

2020111235 马靖淳

一．实验任务

回顾 C 语言编程

二．上机实验时间

3h

三．知识点

1. 熟悉 C 语言环境
2. 传值调用 传地址调用
3. **C 语言的标准内存分配函数（非常不熟悉）**
4. 预定义常量和类型（不太熟悉）
5. 抽象数据类型的表示和实现
6. 算法的有穷性
7. 记录程序运行时间

四．C 语言源代码

1.

（1）

```
#include <stdio.h>
#include <stdlib.h>
#include "Complex.h"

int main()
{
    printf("hello world!\n");
    return 0;
}
```

（2）

```
#include <stdio.h>
#include <stdlib.h>
#include "Complex.h"

int main()
{
    printf("hello world!\n");
    getchar();
    return 0;
}
```

2.

（1）

```
#include <stdio.h>
```

```
#include <stdlib.h>
#include "Complex.h"

void swap(int x, int y)
{
    int t;
    t = x; x = y; y = t;
}
int main()
{
    int a, b, c;
    a = 5; b = 3; c = 2;
    if (a > b) swap(a, b);
    if (a > c) swap(a, c);
    if (b > c) swap(b, c);
    printf("%d %d %d",a,b,c);
    return 0;
}
```

（2）

```
void swap(int *x, int *y)
{
    int t;
    t = *x; *x = *y; *y = t;
}
int main()
{
    int a, b, c;
    a = 5; b = 3; c = 2;
    if (a > b) swap(&a, &b);
    if (a > c) swap(&a, &c);
    if (b > c) swap(&b, &c);
    printf("%d %d %d", a, b, c);
    return 0;
}
```

3.

（1）

```
#include <stdio.h>
#include <stdlib.h>
#include "Complex.h"

int main()
{
```

```
    int* p, * q;
    p = (int*)malloc(10 * sizeof(int));
    q = p;
    p = (int*)realloc(p, 20 * sizeof(int));
    printf("p=0x%x\n",p);
    printf("q=0x%x\n", q);

    return 0;
}
```

（2）

```
#include <stdio.h>
#include <stdlib.h>
#include "Complex.h"

int main()
{
    int* p, * q;
    p = (int*)malloc(10 * sizeof(int));
    q = p;
    p = (int*)realloc(p, 100 * sizeof(int));
    printf("p=0x%x\n",p);
    printf("q=0x%x\n", q);

    return 0;
}
```

4.

```
#include <stdio.h>
#include <stdlib.h>
#include "Complex.h"

#ifndef STATUS_H
#define STATUS_H

#define TURE 1
#define FALSE 0
#define YES 1
#define NO 0
#define OK 1
#define ERROR 0
#define SUCCESS 1
#define UNSUCCESS 0
#define INFEASIBLE -1
#define OVERFLOW -2
```

```
#define UNDERFLOW -3
typedef int Status;
#endif
```

5.

```
/*Complex.h*/
#pragma once
#include <stdio.h>

typedef struct
{
    float realpart;
    float imagpart;
}Complex;

void Assign(Complex* Z, float Realval, float Imagval);
float GetReal(Complex Z);
float GetImag(Complex Z);
void Add(Complex z1, Complex z2, Complex* sum);
/*Complex.c*/
#include "Complex.h"
void Assign(Complex* Z, float Realval, float Imagval)
{
    Z->imagpart = Imagval;
    Z->realpart = Realval;
}
float GetReal(Complex Z)
{
    return Z.realpart;
}
float GetImag(Complex Z)
{
    return Z.imagpart;
}
void Add(Complex z1, Complex z2, Complex* sum)
{
    sum->realpart = z1.realpart + z2.realpart;
    sum->imagpart = z1.imagpart + z2.imagpart;
}
int main()
{
    Complex com_1, com_2, sum;
    Assign(&com_1, 1.2, 2.3);
    Assign(&com_2, 2.1, 3.2);
```

```
    Add(com_1, com_2, &sum);
    printf("sum.realpart=%f\nsum.imagpart=%f\n", sum.realpart, sum.imagpart);
    system("pause");
    return 0;
}
```

6.

```
#include <stdio.h>
#include "Complex.h"
int HailstoneSeq(int n)
{
    int length = 1;
    while (n > 1)
    {
        printf("%d,", n);
        (n % 2 == 0) ? (n /= 2) : (n = n * 3 + 1);
        ++length;
    }
    if (n <= 1) printf("%d,", 1);
    return length;
}

int main()
{
    int n = 24;
    int x;
    x = HailstoneSeq(n);
    system("pause");
    return 0;
}
```

7.

（1）

```
#include <stdio.h>
#include <time.h>

int main()
{
    time_t start, stop;
    clock_t tic, toc;
    tic = clock();
    time(&start);
    for (int i = 0; i <= 1000000000; i++)
    {
```

```
    }
    time(&stop);
    toc = clock();
    printf("elapsed time is %lds\n", stop - start);
    printf("elapsed time is %fs\n", (double)(toc - tic) / CLOCKS_PER_SEC);
    return 0;
}
```

五．控制台程序操作结果截屏

1.

（1）

```
hello world!

E:\新建文件夹\Project1\Debug\Project1.exe (进程 16900)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```

（2）

```
hello world!
```

2.

（1）

```
5 3 2
E:\新建文件夹\Project1\Debug\Project1.exe (进程 13156)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```

（2）

```
2 3 5
E:\大二上\数据结构\C语言\9.13第一周作业\Debug\Project2.exe (进程 14116)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```

3.

（1）

```
p=0x1534f70
q=0x1534f70

E:\大二上\数据结构\C语言\9.13第一周作业\Debug\Project2.exe (进程 25164)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```

（2）

```
p=0xdfdd30
q=0xdf4f70

E:\大二上\数据结构\C语言\9.13第一周作业\Debug\Project2.exe (进程 19128)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”
按任意键关闭此窗口. . .
```

4.

```

E:\大二上\数据结构\C语言\9.13第一周作业\Debug\Project2.exe (进程 10764)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```

5.

```
sum.realpart=3.300000
sum.imagpart=5.500000
请按任意键继续. . .
```

6.

```
24, 12, 6, 3, 10, 5, 16, 8, 4, 2, 1, 请按任意键继续. . .
```

7.

```
elapsed time is 1s
elapsed time is 0.938000s

E:\大二上\数据结构\C语言\9.13第一周作业\Debug\Project2.exe (进程 8684)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```

六．实验总结

1. 学会 C 程序运行的实验环境，了解头文件的引用，与 C++区别
2. 补充了一下 prinf 的用法

（1）如果在程序中要使用 printf 或者 scanf，那么就必须要包含头文件 stdio.h。因为这两个函数就是包含在该头文件中的。

（2）

表 15.5 printf 格式代码

代码	参数	含义
c	int	参数被裁剪为 unsigned char 类型并作为字符进行打印
d i	int	参数作为一个十进制整数打印。如果给出了精度而且值的位数少于精度位度，前面就用 0 填充
u o x,X	unsigned int	参数作为一个无符号值打印，u 使用十进制，o 使用八进制，x 或 X 使用十六进制，两者的区别是 x 约定使用 abcdef，而 X 约定使用 ABCDEF
e E	double	参数根据指数形式打印。例如，6.023000e23 是使用代码 e，6.023000E23 是使用代码 E。小数点后面的位数由精度字段决定，缺省值是 6
f	double	参数按照常规的浮点格式打印。精度字段决定小数点后面的位数，缺省值是 6
g G	double	参数以%f 或%e（如 G 则%E）的格式打印，取决于它的值。如果指数大于等于-4 但小于精度字段就使用%f 格式，否则使用指数格式
s	char *	打印一个字符串
p	void *	指针值被转换为一串因编译器而异的可打印字符。这个代码主要是和 scanf 中的%p 代码组合使用
n	int *	这个代码是独特的，因为它并不产生任何输出。相反，到目前为止函数所产生的输出字符数目将被保存到对应的参数中
%	(无)	打印一个%字符

表 15.6 printf 格式标志

标志	含义
-	值在字段中左对齐，缺省情况下是右对齐
0	当数值为右对齐时，缺省情况下是使用空格填充值左边未使用的列。这个标志表示用零来填充，它可用于 d,i,u,o,x,X,e,E,f,g 和 G 代码。使用 d,i,u,o,x 和 X 代码时，如果给出了精度字段，零标志就被忽略。如果格式代码中出现了负号标志，零标志也没有效果
+	当用于一个格式化某个有符号值的代码时，如果值非负，正号标志就会给它加上一个正号。如果该值为负，就像往常一样显示一个负号。在缺省情况下，正号并不会显示
空格	只用于转换有符号值的代码。当值非负时，这个标志把一个空格添加到它的开始位置。注意这个标志和正号标志是相互排斥的，如果两个同时给出，空格标志便被忽略
#	选择某些代码的另一种转换形式。它们在表 15.8 中描述

（3）可以在"%“和字母之间插进数字表示最大场宽。

例如:

%3d 表示输出 3 位整型数，不够 3 位右对齐。

%9.2f 表示输出场宽为 9 的浮点数，其中小数位为 2，整数位为 6,小数点占一位，不够 9 位右对齐。

%8s 表示输出 8 个字符的字符串，不够 8 个字符右对齐。

如果字符串的长度、或整型数位数超过说明的场宽，将按其实际长度输出。

但对浮点数，若整数部分位数超过了说明的整数位宽度，将按实际整数位输出;

若小数部分位数超过了说明的小数位宽度，则按说明的宽度以四舍五入输出。

（4）另外，若想在输出值前加一些 0，就应在场宽项前加个 0。

例如:

%04d 表示在输出一个小于 4 位的数值时，将在前面补 0 使其总宽度为 4 位。

如果用浮点数表示字符或整型量的输出格式，小数点后的数字代表最大宽度,小数点前的数字代表最小宽度。

例如:

%6.9s 表示显示一个长度不小于 6 且不大于 9 的字符串。若大于 9，则第 9 个字符以后的内容将被删除。

（5）可以在”%“和字母之间加小写字母 l，表示输出的是长型数。

例如:

%ld 表示输出 long 整数

%lf 表示输出 double 浮点数

（6）可以控制输出左对齐或右对齐，即在”%“和字母之间加入一个”-" 号可说明输出为左对齐，否则为右对齐。

例如:

%-7d 表示输出 7 位整数左对齐

%-10s 表示输出 10 个字符左对齐

3. 补充内存分配函数用法

（1）malloc 调用形式为（类型*）malloc（size）

在内存的动态存储区中分配一块长度为 size 字节的连续区域，返回该区域的首地址

（2）calloc 调用形式为（类型*）calloc（n,size）

在内存的动态存储取中分配 n 块长度为 size 字节的连续区域，返回首地址

（3）realloc 调用形式为（类型*）realloc（*ptr,size）

将 ptr 内存大小增大到 size

（4）free 的调用形式为 free（*ptr）

释放 ptr 所指向的一块内存空间

4. 补充类型定义（typedef）用法

（1）数据结构的表示（存储结构）用类型定义（typedef）描述。数据类型约定为 ElemType,由用户在使用该数据类型时自行定义

（2）

使用关键字 typedef 可以为类型起一个新的别名。typedef 的用法一般为：

```
typedef   oldName   newName;
```

oldName 是类型原来的名字，newName 是类型新的名字。
例如：

```
typedef int INTEGER;

INTEGER a, b;

a = 1;

b = 2;
```

INTEGER a, b;等效于 int a, b;。
（3）typedef 在表现上有时候类似于 #define，但它和宏替换之间存在一个关键性的区别。正确思考这个问题的方法就是把 typedef 看成一种彻底的“封装”类型，声明之后不能再往里面增加别的东西。
（4）
#ifndef x //先测试 x 是否被宏定义过
#define x
程序段 1 //如果 x 没有被宏定义过，定义 x，并编译程序段 1
#endif
程序段 2 //如果 x 已经定义过了则编译程序段 2 的语句，“忽视”程序段 1。