VG101 — Introduction to Computer and Programming

Assignment 6 (22/11/2016)

Manuel — UM-JI (Fall 2016)

- Write each exercise in a different file
- Include simple comments in the code
- If applicable, split the implementation over several functions
- Write a single README file per assignment
- Archive all the files in a `zip` file and upload it onto Sakai

Ex. 1 — *Structure, pointers, and functions*

A point P in the complex plan can be defined using either Cartesian or Polar coordinates.

1. Write two structures to represent a point in the complex plan. In the README file explain and argue on your choices, including data types.
2. Write two functions to convert from Cartesian to Polar and from Polar to Cartesian. The functions should use pointers such that more than one set of coordinates can be converted at a time. That is the pointer could contain more than one complex number such that the function would returns the conversion of all the complex numbers at once.
3. In the main function define the following complex numbers and convert them between Cartesian and Polar coordinates: $3+\frac{4}{5}i$, $\log(4)i$, $45.245+0.235i$, $3e^{\frac{i\pi}{17}}$, $4(\cos\frac{\pi}{9}+i\sin\frac{\pi}{9})$, $e^{\frac{i\pi}{12}}$.

Specifications.

- Organise the complex numbers into two arrays: Cartesian and Polar
- Prints the numbers by pair, one pair per line, the original number first and the converted one second

Ex. 2 — *Pointers, loops, and conditional statements*

Write a C program that reads through an array of integers using pointers, and scan through it to find all the values larger than a randomly generated number r. The element r must be smaller than the max of the array.

Specifications.

- Start by reading the number of integers n
- Then read the n space separated integers
- On a new line display the output as space separated elements

Ex. 3 — *Strings and file I/O*

Write a program to find the number of times a given string occurs in a sentence. The program should read the sentence from a file called `sentence.txt` and the word from a file called `word.txt`. The output should be printed in a file `count.txt`.

Specifications.

- The binary, input, and output files are expected to be in the same directory
- Do not use absolute paths
- Do not prompt the user

Ex. 4 — *File I/O, arrays, and loops*

Read two matrices A and B from a text file called `matrices.txt`. Compute $A + B$, $A \cdot B$ and $A^T \cdot B^T$. Output the result into a text file named `result.txt`. Each row of a matrix is represented as a list of integers separated by a space. When the end of a row is reached a new one starts on the next line. Two matrices a separated by a blank line.

Specifications.

- The binary, input, and output files are expected to be in the same directory
- Do not use absolute paths
- Do not prompt the user
- Output the resulting matrices in the order "addition, multiplication, transpose"

Partial sample output (ex. 4)

```
$ ./h6 -ex4
1 2 3
2 3 4
5 6 7

9 8 7
4 3 2
5 7 2
```

Ex. 5 — *Basic object oriented programming in C*

A mathematical set is a collection of distinct objects, such as $\{1, 3, 9\}$, $\{r, g, b\}$, and $\{5, 11, 11\} = \{5, 11\}$. Based on the multi-set header file below write in a corresponding `set.c` file the necessary functions to handle a set (creation, deletion as well as adding and removing elements). Assume a set can contain elements of type either `char`, `int` or `double`.

Hint: to resize a memory block use the function `realloc`.

Multi-set header file (ex. 5)

```c
#define INITSETSIZE 64 // Initial memory allocated for the set
#define CHAR 1
#define INT sizeof(int)
#define DOUBLE sizeof(double)
/* elem: list of elments; card: cardinal of teh set; type: data type (CHAR INT or DOUBLE) */
typedef struct universalSet { void *elem; int card; int type; } uset;
/* Initialize an empty set of given type and allocate the initial memory: INITSETSIZE*type */
void newSet(uset *set, int type);
void deletSet(uset *set); // Free the memory allocated by newSet
/* add elem to the set: check whether it is already in the set;
   resize memory if card = allocated memory; new memory = previous+64
   e.g. before: mem=128, card=128, after: men=192, card=129 */
void addElem(void *elem, uset *set);
/* remove elem from the set; do nothing if the set does not contain this elem;
   resize memory if "too much memory" is used; new = previous-64
  e.g. before: mem=192, card=129, after: card=128, mem=128 */
void remElem(void *elem, uset *set);
```

Group Exercise

The goal of this exercise is to get a better understanding of pointers, while helping each others. For a better result please apply the following suggestions.

- Start thinking of the problem as early as possible;
- Relate linked lists to arrays and think of the differences;
- Think of the difference between inserting/deleting the first element and the last element;
- For a total of nine functions, each student is expected to write three. The workload distribution should be detailed in the README file;

Remark: linked lists are a very common data structure, and many implementations are available online. Do not reuse any code from others; Honor Code will be strictly applied.

Ex. 6 — *Linked lists*

1. Online research questions.
   a) Explain what a linked list is?
   b) List some applications of linked lists.
   c) Search what kinds of linked list exist.
2. Programming questions.
   a) The code provided below has been "compacted" in order to fit over a single page. Reorganise it writing no more than one instruction per line while respecting indentation.
   b) Based on the header file below complete the implementation of the linked list.

Linked list header file (ex. 6)

```c
#ifndef LIST_H
#define LIST_H
typedef struct node{ char ch; struct node *next; } node_t;
typedef enum{false, true} bool;
node_t *Initialize(char ch);
void PrintList(node_t *head);
void FreeList(node_t **head);
bool IsEmptyList(node_t *head); // Return true if the list is empty, false otherwise
void InsertFirstList(node_t **head,char insert_char); // Prepend a node
void InsertLastList(node_t **head,char insert_char); // Append a node
void DeleteFirstList(node_t **head); // Delete the first element in the list
void DeleteLastList(node_t **head); // Delete the last element in the list
int SizeList(node_t *head); // Return the size of the list
int SearchList(node_t **head, char target); // Count how many times target appears
void SplitList(node_t **head, node_t **tail, int pos); // Split into [0;pos-1] and [pos,end]
void MergeList(node_t **head1, node_t **head2); // Merge two lists
#endif
```

Linked list implementation (ex. 6)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "list.h"
int ex6() {
    node_t *a=Initialize('1'); node_t *b=NULL; PrintList(a);
    InsertFirstList(&a, 'V'); InsertFirstList(&a, 'M');
    PrintList(a); InsertLastList(&a, 'C'); PrintList(a);
    SplitList(&a, &b, 2); PrintList(a); PrintList(b);
    DeleteFirstList(&a); PrintList(a); InsertLastList(&a, 'G');
    DeleteLastList(&b); PrintList(b); InsertLastList(&b,'0');
    PrintList(b); InsertLastList(&b, '1'); PrintList(b);
    MergeList(&a,&b); PrintList(a);
    char target='G';
    printf("Count '%c': %d\n",target, SearchList(&a,target));
    target='1';
    printf("Count '%c': %d\n",target, SearchList(&a,target));
    FreeList(&a);
    return 0;
}
node_t *Initialize(char ch) {
    node_t *head;
    head=(node_t*)calloc(1,sizeof(node_t));
    if(head==NULL){ fprintf(stderr,"Failed to assign memory!\n"); exit(-1); }
    head->next=NULL; head->ch=ch;
    return head;
}
void PrintList(node_t *head) {
    node_t *temp=head;
    printf("***Print Linked List***\n");
    while(temp!=NULL) { printf("%c ",temp->ch); temp=temp->next; }
    printf("\n****Print Finished****\n\n");
}
void FreeList(node_t **head) {
    node_t *tmp=NULL; node_t *pHead=*head;
    while(pHead->next!=NULL) { tmp=pHead; pHead=pHead->next; free(tmp); }
    free(pHead);
}
```

Expected output (ex. 6)

```
$  ./h6 -ex6
**Print Linked List***
1
***Print Finished****

***Print Linked List***
M V 1
***Print Finished****

***Print Linked List***
M V 1 C
***Print Finished****

***Print Linked List***
M V
***Print Finished****

***Print Linked List***
1 C
***Print Finished****

***Print Linked List***
V
***Print Finished****

***Print Linked List***
1
***Print Finished****

***Print Linked List***
1 0
***Print Finished****

***Print Linked List***
1 0 1
***Print Finished****

***Print Linked List***
V G 1 0 1
***Print Finished****

Count 'G': 1
Count '1':  2
```

Groups

耿若馨 (Geng Ruoxin) 516370910077
马至皓 (Ma Zhihao) 516370910039
陆思轶 (Lu Siyi) 5133709149

项之渊 (Xiang Zhiyuan) 516370910126
秦亦蕉 (Qin Yijiao) 516021910124
翁可歆 (Weng Kexin) 516370910163

洪颖慧 (Hong Yinghui) 516370910160
董政元 (Dong Zhengyuan) 516370910113
金浩 (Jin) 516370910116

杜轲 (Du Ke) 516370910114
刘诗雨 (Liu Shiyu) 516370910027
SARIT KITTIRATTANAPAIBOON 516370990012

刘知正 (Liu Zhizheng) 516370910063
沈琮凯 (Shen Congkai) 516370910066
时广泽 (Shi Guangze) 516021910199

汪庭康 (Wang Tingkang) 516370910124
李睿 (Li Rui) 516370910251
OSAMA MALIK AWAN 516370990003

王师溟 (Wang Shi) 516021910693
成隽奕 (Cheng Junyi) 516370910032
李金儒 (Li Jinru) 516370910036

孙琦岳 (Sun Qiyue) 516021910259
郑智心 (Zheng Zhixin) 516370910083
王天予 (Wang Tianyu) 516370910256

刘倪逸秋 (Liu Niyiqiu) 516370910118
朱文琪 (Wenqi Zhu) 516370910217
娄渊清 (Lou Yuanqing) 516370910120

廖昕皓 (Liao) 516370910037
张沈宇 (Zhang Shenyu) 516370910129
陈东箭 (Chen Dongjian) 516370910166

朱文耀 (Zhu Wenyao) 5143709246
朴柱薰 (Pu Zhu) 516370990013
周家成 (Zhou Jiacheng) 516370910075

葛天逸 (Ge Tianyi) 516370910168
朱波颖 (Zhu Boying) 516370910165
郭成彰 (Guo Chengzhang) 516021910639

TAE HOON KWEON 516370990005
任欣阳 (Ren Xinyang) 516370910080
王嫣然 (Yanran Wang) 516370910082

胡哲榜 (Hu Zhebang) 516370910115
唐星烨 (Tang Xing) 516370910042
周耘平 (Chou YunPing) 516370910156

熊家正 (Xiong Jiazheng) 516021910564
关书文 (Guan Shuwen) 516021910603
陶奕彤 (Tao Yitong) 516370910081

聂鞠荷 (Nie Juhe) 516021910662
姜乐钧 (Jiang Lejun) 516370910250
刘惟中 (Liu Weizhong) 515370910160

孙旭华 (Sun Xuhua) 516370910255
李雨姗 (Li Yushan) 516370910026
JESCO ALEXANDER BECK 714370990038

王菁滢 (Wang) 516370910029
张佳宁 (Zhang Jianing) 516370910073
盛禹国 (Sheng Yuguo) 516370910041

孙天行 (Sun Tianxing) 516021910289
朱柯阅 (Zhu Keyue) 516370910158
陈曦雯 (Chen Xiwen) 516021910271

王涵之 (Wang Hanzhi) 516021910768
丁恺雯 (Ding) 516370910024
鲁瑞明 (Lu Ruiming) 516021910167

苏镜瑜 (Su Jing) 516370910123
王瑞琦 (Wang Ruiqi) 516370910203
黄子瀛 (Huang Ziying) 516370910169

潘崇聃 (Pan Chongdan) 516370910121
仰佳欣 (Yang Jiaxin) 516370910205
闫李豪 (Yan Lihao) 516021910715

江南佳 (Nanjia Jiang) 516370910212
孙艺 (Sun Yi) 516370910214
陈培煜 (Chen Peiyu) 516021910369

杨瑞敏 (Yang Ruimin) 516370910127
陈嘉懿 (Chen Jiayi) 516370910211
蒋沁诚 (Jiang Qincheng) 516370910034

谢舒翔 (Xie Shuxiang) 516370910070
朱昊杰 (Zhu Haojie) 515010910035
孟天宇 (Meng Tianyu) 516370910065

瞿昱易 (Qu Yuyi) 516370910238
周家西 (Zhou Jiaxi) 516370910110
陈智博 (Chen Zhibo) 515020910276

刘嘉晨 (Liu Jiachen) 516370910079
沈阳 (Shen) 516370910040
周宇航 (Zhou Yuhang) 516021910296

陈天博 (Chen Tianbo) 5133709262
林牧星 (Lin Muxing) 516370910078
彭以绘 (Peng Yihui) 516370910213

沈子浩 (Shen Zihao) 516370910200
周瑞星 (Zhou Ruixing) 516370910130
何航吉 (He Hangji) 516370910247

宣思源 (Xuan Siyuan) 516370910215
潘智辉 (Pan Zhihui) 516370910174
陈佳茜 (Chen Jiaxi) 516370910159

Delmwin BAEKA 516370990007
范哲良 (Fan Zheliang) 516021910518
吴明远 (Wu Mingyuan) 516370910125

陶元杰 (Tao Yuanjie) 516370910068
朱皓轩 (Zhu Haoxuan) 516370910157
郭凌云 (Guo Lingyun) 516021910303

钱星 (Qian XingYue) 516370910028
吕步 (Lv) 516370910173
芮意进 (Rui Yijin) 516370910254

沈缘 (Shen Yuan) 516370910122
王奕纬 (Wang Yiwei) 516370910043
沈叶沁 (Shen Yeqin) 516370910162

苏星宇 (Su Xingyu) 516021910560
屠立芸 (Tu Liyun) 516370910240
何屹滔 (He Yitao) 516370910248

石一茗 (Shi Yiming) 516370910108
乐一炜 (Le Yiwei) 516370910117
鲍家君 (Bao Jiajun) 516370910112

叶柔霜 (Ye Roushuang) 516370910241
秦宇隽 (Qin Yujun) 516370910199
戚伏波 (Qi Fubo) 516370910175

王依勤 (Wang Yiqin) 516370910109
苏浩雄 (Haoxiong Su) 516021910707
刘紫薇 (Ziwei Liu) 516370910161

胡炳城 (Hu Bingcheng) 516021910219
於宛灏 (Yu Wanhao) 516370910258