C/C++205_LAB1&LAB2

Learn from each other

```
#include <stdio.h>
#include <stdio.h>
#include <string.h>
#define INPUT_LEN 80
#define NAME_LEN 20
#define ID_LEN 8
int main()
{
        char input[INPUT_LEN];
        char name[NAME_LEN];
        char ID[ID_LEN];
        fprintf(stderr,"Please enter your name: ");
        fgets(input, INPUT_LEN, stdin);
        strncpy(name, input, NAME_LEN);
        int num = strlen(name);
        name[num - 1] = '\0’;
        fprintf(stderr,"Please enter your ID: ");
        fgets(input, INPUT_LEN, stdin);
        strncpy(ID, input, ID_LEN);
        printf("Welcome %s to the C and C++ world! %s is your student ID.\n",name, ID);
        return 0;
}
```

```
#include<stdio.h>

int main(){
       char n [20];
       long i;
       printf("Please enter your name:");
       fgets(n,20,stdin);

       printf("Please enter your ID:");

       if (scanf("%ld",&i) == 1)
              printf("Welcome %s %ld to the C and C++ World!",n,i);
       else
              printf("ERROR!Interger expected");

       return 0;
}
```

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

int main(){
	char studentID[20],name[30];
	char myname[30],mystudentID[20];

	//为了防止name和id长度超过设定长度，用fgets
	printf("Please enter your student number:");
	fgets(studentID,20,stdin);
	sscanf(studentID,"%s",mystudentID);

	printf("Please enter your name:");
	fgets(name,30,stdin);
	sscanf(name,"%s",&myname);

	printf("welcome  %s  %s to the C and C++ world!",mystudentID,myname);
}
```

- char user_name_id[20] = “0123456789_0123456789";

- {
  - fp=fopen("text.txt","w+");
  - fscanf(fp,"%s",tex);
  - fclose(fp);
- }

```
int main()
{
        FILE *fp;  char a[60]; int i ; char key[9]={'V','E','G','I','N','E','R','E','\0'};
        fp = fopen("file.txt" , "r");
        if(fp == NULL)    {       perror("Error opening file");       return(-1);    }
        fgets (a, 60, fp);    fclose(fp);
        for(i = 0; i < sizeof(a); i++)        a[i] = toupper(a[i]);
        char b[sizeof(a)];    i = 0;   int j = 0;
        while(i < sizeof(a)-1){
                if(isalpha(a[i])){       b[i] = key[j];     }else {       b[i] = a[i];     }
                i++;
                if(j==7){      j-=7;    }    else      j++;
        }
        char vigenere[26][26];intp,q;
        for(p=0;p<=25;p++)
        {     for(q=0;q<=25;q++){    vigenere[p][q] = 'A'+(p+q)%26;  }    }
        i = 0;   int x,y;   char c[sizeof(a)];
        while(i < sizeof(a)-1){
                if(isalpha(a[i])){     x = b[i] - 65;     y = a[i] - 65;     c[i] = vigenere[x][y];   }
                else{     c[i] = a[i];   }
        i++;   }
        for(i=0;i<sizeof(c);i++){     printf("%c",c[i]);   }
        return 0;
}
```

```
char *encode(char key[], char plainText[]);

int main(int argc, char *argv[]) {
    char k[1024];                              // store the inputed key
    char key[1024];                            // store the key which only contains alphabets
    char plainText[1024];
    if (argc == 4) {
        char *k = argv[1];
        int kIndex = 0;
        for (int i = 0; i < strlen(k); i++) {
            if (isalpha(k[i])) {
                key[kIndex] = k[i];
                kIndex++;
            }
        }
        key[kIndex] = '\0';
        fgets(stdin, 1024, plainText);
        char *cipherText = encode(key, plainText);
        printf("%s\n", cipherText); } else {
        printf("Please quote your password\n");
        exit(-1);
    }
}
```

```
int main(int argc, char *argv[])
{
   int i,j,k;
   char a='A';
              char c;
              char alpha[26][26];
              char password[PASSWORD_LEN];
              char *p;
              p  =argv[1];
              i = 0;
              while ((i < PASSWORD_LEN) && (*p != '\0')){
              if (isalpha(*p)) {
                             password[i] = toupper(*p);
                             i++;
                }
                p++;
               }
              password[i] = '\0';
              printf("%s\n", password);
              int len1=strlen(password);

              ......
```

```
#include <stdio.h>
#include <string.h>
int main(int argc,char *argv[])
{
       int i,j=0,L;
       char temp;
       char ori[100];       char res[100];       char key[100];
       scanf("%[^\n]",&ori);
       L=strlen(ori); printf("L : %d\n",L);
       for(i=0;i<=L;i++)   {
              if(ori[i]>=97&&ori[i]<=122)
              {            ori[i]=ori[i]-32;        }
       }
       for(i=0;i<=L;i++)    {
              if(ori[i]>=65&&ori[i]<=90)
              {             res[j]=(ori[i]-'A'+*argv[i%(argc-1)]-'A')%26+'A';        }
              else
              {            res[j]=ori[i];        }
              j++;
        }
       printf("%s",res);
       return 0;
}
```

```
int main(int argc, char **argv) {
        //Check parameter count
        if(argc != 3) {
                fputs("Usage:\n\tvigenere -e/-d PASSWORD < [input file] > [output file]\n", stderr);
                fputs("\t-e ---- Encrypt\n", stderr);
                fputs("\t-d ---- Decrypt\n", stderr);                    return 1;  }
        //Check parameter -e/-d
        if(strcmp(argv[1], "-e") && strcmp(argv[1], "-d")) {
                fputs("Usage:\n\tvigenere -e/-d PASSWORD < [input file] > [output file]\n", stderr);
                fputs("\t-e ---- Encrypt\n", stderr);
                fputs("\t-d ---- Decrypt\n", stderr);                    return 2;  }
        int pwlen = strlen(argv[2]);     char pw[strlen(argv[2])+2];
        //Digest password
        int i, j=0;  for(i=0;i<pwlen;i++)
        if(isalpha(argv[2][i])) {   pw[j] = toupper(argv[2][i]);          j++;                  }
        pw[j] = 0;
        //Check password valid length
        pwlen = strlen(pw);
        if(pwlen < 1) {
        fputs("Invalid Password!\nUsage:\n\tvigenere -e/-d PASSWORD < [input file] > [output file]\n", stderr);
e ---- Encrypt\n", stderr);              fputs("\t-d ---- Decrypt\n", stderr);             return 3;  }            ...

   return 0;}
```

lab2

More comments ☺

more testcase,more segment fault ☹

```
int main(int argc, char **argv)
{
            char *sprt;     Char *ignoreline;
            char *word;
            int args[argc-7];
            int ch;
            fprintf(stderr,"size of args : %d ,argc : %d\n",sizeof(args),argc);
            while((ch=getopt(argc,argv,"s:i:"))!=-1)
            {           switch(ch)
                        {           case 's':
                                    {           sprt=optarg;                  break;
                                    }
                                    case 'i':
                                    {       ignoreline=optarg;       int i8;
                                            for(i8=optind;i8<argc;i8++)
                                            {
                                                args[i8-optind]=atoi(argv[i8]);
                                            }
                                                break;
                                    }
                        }
            }
```

```
int *exact;
if (optind < argc) {
        exact = (int *) malloc(sizeof(int) * size);
    }
…
int i = optind;
    for (; i < argc; ++i) {//copy argv to exact
        exact[i - optind] = atoi(argv[i]);
        // printf("argv%d ", atoi(argv[i]));
    }
…
free(exact);
```

```
#include <stdio.h>
#include <unistd.h>
static char row[1000];
static int count = 0;
void reader() {
   int i=0,j=0,k=0;
   char c,a;
             fflush(stdout);
             c = getchar();
             while(c != EOF) {                                              //if the first  character is EOF , finish the loop
                      while(1){
                                   if(c=='\"') {
                                            row[j] = c;j++;
                                            c = getchar();
                                            for(;c != '\"';j++) {     row[j]= c;   c = getchar();           }
                                            row[j] = c;    j++;
                                            c = getchar();
                                   }
...
            }
}
```

```
char *contand;
contand = strtok(line,delimiter1);
int o;
for(o = 0; o < strlen(contand); o++){
        if(contand[o] == '?')
        contand[o] = delimiter;  // get the content in the specify field
}
```

```
int main(int argc, char *argv[])
{
            char *sep;     int x;
            int num[100];
            int opt;
            while((opt=getopt(argc,argv,":s:i:")) != -1)
            {
                        switch(opt)
                        {     case 's': //the first optarg for separator
                                    sep = optarg;     break;
                              case ':':     printf("invalid option char!\n");     break;
                              case 'i':      ....  break;
                     }
            }
            while((ch=getchar())!=EOF){
                        if(ch==""){      quota++;    }
                        if('\n'==ch){     lines++;   sepnum=0;     printf("\n");     }
                        if(*sep==ch){
                                    if((quota%2)==0){       sepnum++;        }
                        }
         }
}
```

```
int main (int argc,char *argv[]){
    char *separater;       // store the argument of '-s'
    int *ignore;           // store the argument of '-i’     //int igd;
    int ch;                // ignore=&igd;

    while((ch=getopt(argc,argv,"s:i:"))!=-1){
        switch(ch){
            case 's':
                if(*optarg!=','){               printf(“\nhint:。 。 。 ");      }
                separater=optarg;
                  break;
            case 'i':
                if(atoi(optarg)==0){
                fprintf(stderr,"\nwarning! the command you entered for '-i' is not a integer!\n");
                   return 1;
                }

                *ignore=atoi(optarg)-1;
                fprintf(stderr,"The rows you want to ignore is lower than : %d\n",atoi(optarg));
                break;
        }
    }
```

```
int main(int argc, char *argv[])
{
        int ch;    opterr = 0;
        int line;  char replace= ' ‘;
        while((ch = getopt(argc,argv,"s::i:"))!=-1)
        {
                fprintf(stderr,"ch: %c ,optarg: %s\n",ch,optarg);
                switch(ch)
                {
                        case's':
                        sscanf(optarg,"%c",&replace);
                        case'i':
                        sscanf(optarg,"%d",&line);
        }
        }
      。 。 。
}
```

```
int main(int argc, char *argv[]){

    int i=0,l=0,n=0,k=0,delete,l1,l2,l3;
    char c,mark;
    while((c = getopt(argc, argv, "s:i:")) != -1){
                switch(c){
                  。。。
        }
    }

    l1=atoi(argv[optind-1]);
    l2=atoi(argv[optind+1-1]);
    l3=atoi(argv[optind+2-1]);
    if(l1==l2||l2==l3||l3==l1){
        printf("choose different field,please.");
        exit(0);
    }
    。。。
```

```
while((ch = getopt(argc, argv, "s:i:")) != -1)
{。 。 。   }

while(fgets(line1, 1000, stdin) != NULL)
{
        int iargc = optind;
        if( (count_commas + 1) == atoi(argv[iargc])&&iargc < argc)
        {
                printf("%s\n", fragment);
                iargc++;
        }
}
```