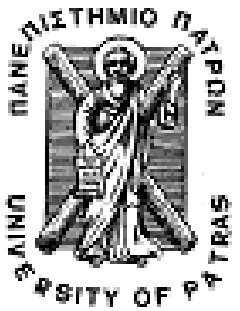
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ & ΠΛΗΡΟΦΟΡΙΚΗΣ



ΕΙΣΑΓΩΓΗ ΣΤΟ ΔΙΑΔΙΚΑΣΤΙΚΟ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ (2009-2010)

ΥΠΕΥΘΥΝΟΙ ΔΙΔΑΣΚΟΝΤΕΣ ΕΡΓΑΣΤΗΡΙΟΥ: Α. ΦΩΚΑ, Κ. ΣΤΑΜΟΣ

ΑΣΚΗΣΗ ΕΞΑΣΚΗΣΗΣ
ΧΩΡΙΣ ΕΞΕΤΑΣΗ ΚΑΙ ΒΑΘΜΟΛΟΓΗΣΗ

Η άσκηση σε αυτό το φυλλάδιο είναι μόνο για εξάσκηση, δεν θα εξεταστεί και δεν θα βαθμολογηθεί.

Θα ανακοινωθεί ενδεικτική λύση της άσκησης πριν την τελική εξέταση του εργαστηρίου.

Γράψτε πρόγραμμα για την διαχείριση του προσωπικού μιας μονάδας. Σας ζητείτε να παράγετε λογισμικό που να διαχειρίζεται το σύστημα του Γραφείου Προσωπικού της Μονάδας σύμφωνα με τις παρακάτω προδιαγραφές.

1) Για κάθε στρατιώτη κρατάμε πληροφορίες για:
    a. Τον Αριθμό Μητρώου του (ΑΣΜ) (εξαψήφιος ακέραιος)
    b. Το μικρό του όνομα (αλφαριθμητικό)
    c. Το επίθετό του (αλφαριθμητικό)
    d. Την ημερομηνία παρουσίασης στο στράτευμα (στη μορφή ΗΗ/ΜΜ/ΧΧΧΧ)
    e. Την προβλεπόμενη ημερομηνία απόλυσης από αυτό (στη μορφή ΗΗ/ΜΜ/ΧΧΧΧ)
    f. Τις συνολικές μέρες άδειας που έχει πάρει
    g. Τις συνολικές ώρες σκοπιάς και άλλων υπηρεσιών που έχει εκτελέσει.
2) Τα στοιχεία των στρατιωτών αποθηκεύονται σε αρχεία κειμένου. Τα αρχεία αυτά έχουν σε κάθε γραμμή πληροφορίες για έναν μόνο στρατιώτη και κάθε γραμμή είναι της μορφής: ΑΣΜ, Όνομα, Επίθετο, Ημερομηνία Παρουσίασης, Ημερομηνία Απόλυσης, Συνολικές Ημέρες Άδειας, Συνολικές Ώρες Υπηρεσίας
3) Το λογισμικό θα πρέπει να μπορεί να διαβάζει από το αρχείο τα στοιχεία και να τα αποθηκεύει στη μνήμη (η επιλογή του αρχείου θα γίνεται από τη γραμμή εντολών κατά την εκτέλεση του προγράμματος).
4) Το λογισμικό θα πρέπει να μπορεί να γράφει τα στοιχεία από τη μνήμη σε αρχείο κειμένου (το ίδιο που δόθηκε από τη γραμμή εντολών κατά την εκτέλεση του προγράμματος).

5) Το λογισμικό θα πρέπει με τη χρήση ενός μενού να δίνει τη δυνατότητα:
    a. Εισαγωγής ενός καινούριου στρατιώτη (πάντα στο τέλος της δομής που χρησιμοποιείται).
    b. Εύρεσης και Διαγραφής ενός υπάρχοντος στρατιώτη.
    c. Εύρεσης και Εκτύπωσης εγγραφής συγκεκριμένου στρατιώτη ανάλογα με αριθμό μητρώου ή επίθετο.
    d. Εκτύπωσης όλων των εγγραφών.
    e. Ταξινόμησης των εγγραφών κατά το πεδίο ΕΠΙΘΕΤΟ και ανάλογα με τη φορά (αύξουσα ή φθίνουσα) που θα ορίζει ο χρήστης.
    f. Επιλογής Ν πρώτων στρατιωτών για άδεια
    g. Επιλογής Ν πρώτων στρατιωτών για υπηρεσία
    h. Αποθήκευσης της παρούσας κατάστασης σε αρχείο κειμένου ανάλογα με την αρχική επιλογή του χρήστη.

Ο χρήστης θα πρέπει να καλεί το πρόγραμμα από τη γραμμή εντολών με τις εξής παραμέτρους:

executable filename

Όπου executable, το όνομα του εκτελέσιμου αρχείου του προγράμματός σας και filename το όνομα του αρχείου που περιέχει τα στοιχεία του Γραφείου Προσωπικού της Μονάδας (αν δεν υπάρχει αγνοείται).

Με το ξεκίνημα της εφαρμογής θα φορτώνονται στη μνήμη από το αρχείο τα στοιχεία των στρατιωτών (με τη σειρά που διαβάζονται από το αρχείο) – η διαδικασία θα γίνεται αυτόματα – και θα παρουσιάζεται στο χρήστη η δυνατότητα να επιλέξει από ένα μενού αν θέλει να πραγματοποιήσει:

1. Εισαγωγή στρατιώτη
2. Εύρεση (κατά ΑΣΜ ή επίθετο) και Διαγραφή στρατιώτη (στη διαγραφή πρέπει να απελευθερώνεται και η χρησιμοποιούμενη μνήμη)
3. Εύρεση (κατά ΑΣΜ ή επίθετο) και Εκτύπωση στρατιώτη
4. Εκτύπωση όλων των στρατιωτών
5. Ταξινόμηση των στρατιωτών κατά το ΕΠΙΘΕΤΟ (αύξουσα ή φθίνουσα σειρά επιλέγεται από το χρήστη) και εκτύπωσή τους

6. Επιλογής των Ν πρώτων στρατιωτών για άδεια Μ ημερών με βάση των αριθμό των ωρών υπηρεσίας, δηλαδή όποιος έχει τις περισσότερες ώρες υπηρεσίας παίρνει άδεια. Θα πρέπει να προσέχετε πως οι μέρες άδειας μπορούν να είναι το πολύ MAX_ADEIA (το ορίζετε ως σταθερά) και πως κάποιος πρέπει να την εξαντλήσει το πολύ μία μέρα πριν απολυθεί, σε αυτή την περίπτωση ο συγκεκριμένος στρατιώτης έχει προτεραιότητα ανεξάρτητα από το αν έχει λιγότερες ή περισσότερες ώρες υπηρεσίας. Αποθήκευση των ονομάτων που έχουν άδεια σε ένα αρχείο adeies.dat.
7. Επιλογής των Ν πρώτων στρατιωτών για υπηρεσία, εξαιρουμένων όσων έχουν άδεια αυτή τη στιγμή. Ενημέρωση του αντίστοιχου πεδίου.
8. Αποθήκευση της παρούσας κατάστασης (π.χ. αν οι εγγραφές έχουν ταξινομηθεί σε προηγούμενη ενέργεια του χρήστη τότε αποθηκεύονται ταξινομημένες) στο ίδιο αρχείο που έχει δοθεί από τη γραμμή εντολών αρχικά.
9. Έξοδο από το λογισμικό (η ευθύνη της αποθήκευσης βαρύνει τον χρήστη). Μπορείτε να αποθηκεύετε τις εγγραφές στη μνήμη σας σε απλά ή διπλά συνδεδεμένη λίστα. Αποφασίστε τι προτιμάτε και να είστε έτοιμοι να δικαιολογήσετε την απόφασή σας αυτή. Τα πρότυπα των συναρτήσεων τα καθορίζετε εσείς και πρέπει επίσης να είστε σε θέση να δικαιολογήσετε τις επιλογές σας.

Η σχεδίαση των συναρτήσεων πρέπει να είναι τέτοια ώστε να μπορούν να επαναχρησιμοποιηθούν στον κώδικά σας: π.χ., η υλοποίηση της εκτύπωσης όλων των στρατιωτών (που ζητείται στο ερώτημα (4)) μπορεί να χρησιμοποιεί τη συνάρτηση εκτύπωσης ενός στρατιώτη που ζητείται στο ερώτημα (3).

Κάντε τρεις διαφορετικές υλοποιήσεις για την αποθήκευση των εγγραφών σας χρησιμοποιώντας:

- Πίνακα (ο οποίος θα μεγαλώνει ανάλογα με τις ανάγκες)
- Απλά συνδεδεμένη λίστα
- Διπλά συνδεδεμένη λίστα

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#define MAX_ADEIA 45

struct soldier{
        char* am;
        char* name;
        char* surname;
        char* indate;
        char* outdate;
        int adeia;
```

```
        int ypiresies;
};

int flength(FILE *file)
{
    int i=0;
    char *line;
    char x;
    while((x=fgetc(file)) != EOF)
       {
              if (x == '\n')
                     i++;
       }
    fclose(file);
    return i;
}

char*  stringcopy(char *str, char ch)
{
     int i, ii, k, l;
     l = strlen(str);
     i=k=0;
     char *tmp, *tmp1;


     while(i < l && str[i] != ch)
            i++;

     tmp = (char *)malloc((i+1)*sizeof(char));

     tmp1 = (char *)malloc((l-i+1)*sizeof(char));

     for (ii = 0; ii< i; ii++)
         tmp[ii] = str[ii];
     tmp[ii] = '\0';

     i++;
     while(i < l)
            tmp1[k++] = str[i++];

     tmp1[k] = '\0';

     str = strcpy(str, tmp1);

     return tmp;
}

struct soldier SoldierFromLine(char *line)
{
     int i;
     struct soldier temp;

     temp.am = stringcopy(line, ',');
     temp.name = stringcopy(line, ',');
     temp.surname = stringcopy(line, ',');
     temp.indate = stringcopy(line, ',');
     temp.outdate = stringcopy(line, ',');
     temp.adeia = atoi(stringcopy(line, ','));
     temp.ypiresies = atoi(stringcopy(line, ','));

     return temp;
}

void copyFileToStruct(FILE *file, char *filename, struct soldier *soldiers)
{
     int i=0, d=0;
     char line[255];
     char x;
        file = fopen(filename, "a+" );

     while((x=fgetc(file)) != EOF)
        {
              if (x == '\n')
              {
                line[d] = '\0'; d = 0;
          soldiers[i++] = SoldierFromLine(line);
              }
```

```
                else
                        line[d++] = x;
        }
            fclose(file);

        return;
}

void printSoldierInfo(struct soldier cur)
{
        printf("%6s %15s %15s %15s %15s %6d %6d\n", cur.am, cur.name, cur.surname, cur.indate,
cur.outdate, cur.adeia, cur.ypiresies);
}

void printAll(struct soldier *soldiers, int l)
{
        int i;
        printf("Printing %d soldiers info....\n", l);

        printf("    %6s %15s %15s %15s %15s %6s %6s\n", "AM", "Onoma", "Epwnymo", "Hm. Eisodou",
"Hm. Apolyshs", "Adeies", "Yphresies");

        for (i = 0; i < l; i++)
        {
                printf("%3d. ", i+1);
                printSoldierInfo(soldiers[i]);
        }
        return;
}


struct soldier getSoldierInfo(void)
{
        struct soldier tmp;

        tmp.am = (char *)malloc(100*sizeof(char));
        tmp.name = (char *)malloc(100*sizeof(char));
        tmp.surname = (char *)malloc(100*sizeof(char));
        tmp.indate = (char *)malloc(100*sizeof(char));
        tmp.outdate = (char *)malloc(100*sizeof(char));

        printf("AM: ");
        gets(tmp.am);
        gets(tmp.am);
        printf("Onoma: ");
        gets(tmp.name);
        printf("Epwnymo: ");
        gets(tmp.surname);
        printf("Indate: ");
        gets(tmp.indate);
        printf("Outdate: ");
        gets(tmp.outdate);
        printf("Adeia: ");
        scanf("%d", &tmp.adeia);
        printf("Yphresies: ");
        scanf("%d", &tmp.ypiresies);

        return tmp;
}

struct soldier*  addSoldier(struct soldier *soldiers, int l)
{
     struct soldier *tmp;
     int i;

     tmp = (struct soldier *)malloc(l * sizeof(struct soldier));

     for(i=0; i < l-1; i++)
        tmp[i] = soldiers[i];

     tmp[l-1] = getSoldierInfo();

     return tmp;
}
int findSoldier(struct soldier *soldiers, int k, char str[], int pos)
{
        int i;
```

```
        for (i=pos; i < k; i++)
                if (strcmp(soldiers[i].am, str) == 0 || strcmp(soldiers[i].surname, str) == 0)
                        return i;

        return -1;
}

struct soldier* removeSoldier(struct soldier *soldiers, int pos, int k)
{
        struct soldier *tmp;
        int i, j;

        tmp = (struct soldier *)malloc(k*sizeof(struct soldier));

        j=0;
        for (i=0; i <= k; i++)
                if (i != pos)
                        tmp[j++] = soldiers[i];

        return tmp;

}

struct soldier* deleteSoldier(struct soldier *soldiers, int *l)
{
        char str[100];
        int pos=0;
        int k, i;
        k = *l;

        printf("Dwste AM 'h epi8eto stratiwth gia diagrafh: ");
        gets(str);
         gets(str);
        do{
                pos = findSoldier(soldiers, k, str, pos);
                if (pos != -1)
                {
                        k--;
                        soldiers = removeSoldier(soldiers, pos, k);
                }
        }while(pos != -1);
        printf("Deleted %d soldiers ...\n\n", (*l - k));
        *l = k;
        return soldiers;
}

void anazitisiSoldier(struct soldier *soldiers, int l)
{
         char str[100];
         int pos=0;
         int k, i;
         k = l;
        i = 1;

         printf("Dwste AM 'h epi8eto stratiwth gia anazitisi: ");
         gets(str);
         gets(str);

         do{
                 pos = findSoldier(soldiers, k, str, pos);
                 if (pos != -1)
                 {
                         printf("%d.   ",i++);
                         printSoldierInfo(soldiers[pos++]);
                 }
         }while(pos != -1);
}

struct soldier * sortSoldiersYp(struct soldier *soldiers, int l)
{
        int a, b;
        struct soldier tmp;

        for (a=1; a < l; ++a)
                for (b = l-1; b >= a; --b)
                        if (soldiers[b-1].ypiresies > soldiers[b].ypiresies)
                        {
```

```
                                tmp = soldiers[b-1];
                                soldiers[b-1] = soldiers[b];
                                soldiers[b] = tmp;
                        }

        return soldiers;
}

void yphresies(struct soldier *soldiers, int l)
{
        int k, i;

        soldiers = sortSoldiersYp(soldiers, l);
        printf("Plh8os stratiwtwn gia yphresia: ");
        scanf("%d", &k);

        for (i=0; i < k; i++)
        {
                printf("%d.  ", i+1);
                printSoldierInfo(soldiers[i]);
                soldiers[i].ypiresies++;
        }

}

time_t to_seconds(char *src_date)
{
        char *fmt="%m/%d/%Y"; /* mm/dd/yy format for date << change as needed */
        struct tm tmp_time;
        if(! *src_date) return 0; /* this means an error */
        strptime(src_date, fmt, &tmp_time);
        return mktime(&tmp_time);
}

void adeies(struct soldier *soldiers, int l)
{
        time_t lt, st;
        int N, M, c, msec, i, k;

        printf("Hmeres adeias: ");
        scanf("%d", &M);
        printf("Plh8os stratiwtwn: ");
        scanf("%d", &N);

        msec = M*24*60*60;
        lt = time(NULL);

        c = 0;
        for (i = 0; i < l; i++)
        {
                st = to_seconds(soldiers[i].outdate);
                if (lt + msec >= st && soldiers[i].adeia < MAX_ADEIA)
                {
                        printf("%d.  ", c+1);
                        printSoldierInfo(soldiers[i]);
                        soldiers[i].adeia = MAX_ADEIA;
                        c++;
                }
        }

        soldiers = sortSoldiersYp(soldiers, l);

        i = l-1;

        while (c < N && i >=0)
        {
                if (soldiers[i].adeia < MAX_ADEIA)
                {
                        printf("%d.  ", ++c);
                        printSoldierInfo(soldiers[i]);
                        soldiers[i].adeia += M;
                }
                i--;
         }

}
void saveAll(struct soldier *soldiers, int l, char filename[])
```

```
{
        FILE *file;
        int i;
        char str[255];
        file = fopen(filename, "w");
        for (i=0; i < l; i++)
                fprintf(file,
"%s,%s,%s,%s,%s,%d,%d\n",soldiers[i].am,soldiers[i].name,soldiers[i].surname,
soldiers[i].indate, soldiers[i].outdate, soldiers[i].adeia, soldiers[i].ypiresies);

        fclose(file);


}


int main(int argc, char *argv[])
{
  char *filename;
  FILE *file;
  int s;
  int l;

  struct soldier *soldiers;

  if (argc == 2)
          filename = argv[1];
  else
          filename = "soldiers.txt";

  printf("Opening file %s .....\n", filename);

  if ((file = fopen(filename, "a+" ))==NULL)
  {
           printf("File cannot be opened or created\n");

  }
  else
  {
          l = flength(file);
                printf("Found %d soldiers .... \n", l);
          soldiers = (struct soldier *)malloc(l*sizeof(struct soldier));
          copyFileToStruct(file, filename, soldiers);
  }

  do{
          printf("------- M E N U -----\n");
          printf("1. Eisagwgh Neou Stratiwth\n");
          printf("2. Diagrafh Stratiwth\n");
          printf("3. Euresh Stratiwth\n");
          printf("4. Ektypwsh Olwn\n");
          printf("5. Proteraiothta Adeias\n");
          printf("6. Epilogh Stratiwtwn gia Yphresies\n");
          printf("7. Apo8hkeush Allagwn\n");
          printf("8. Exodos\n");
          printf("Dwste thn epilogh sas: ");
          scanf("%d", &s);

          switch(s){
               case 1:
                       l++;
                       soldiers = addSoldier(soldiers, l);
                       break;
               case 2:
                       soldiers = deleteSoldier(soldiers, &l);
                       break;
               case 3:
                       anazitisiSoldier(soldiers, l);
                       break;
               case 4:
                       printAll(soldiers, l);
                       break;
               case 5:
                       adeies(soldiers, l);
                       break;
               case 6:
                       yphresies(soldiers, l);
```

```
                        break;
                case 7:
                        saveAll(soldiers, l, filename);
                        break;
                case 8:
                        printf("Exodos....\n\n");
                        break;
                default:
                        printf("Mh egkyrh epilogh!!\n\n");
          }
        }while(s >=1 && s < 8);


  return 0;
}
```