που μας ανταμοίβουν υποπολλαπλασιάζοντας την απαιτούμενη υπολογιστική ισχύ.

Αυτό πάντα δεν ισχύει, για παράδειγμα οι αλγόριθμοι διαγραφής εγγραφών και ενημέρωσης εγγραφών της ενότητας 2 του παρόντος κεφαλαίου είναι ευφυείς και δυσκολότεροι στη σύλληψη και υλοποίηση σε σχέση με τους απλοϊκότερους αλγορίθμους της ενότητας 4 του κεφαλαίου 10 αλλά μειώνουν ελάχιστα την υπολογιστική πολυπλοκότητα.

11.5 Ολοκληρωμένο παράδειγμα – Μισθοδοσία

Στη συνέχεια θα παρουσιάσουμε ένα πακέτο (σύνολο) προγραμμάτων που θα μπορούσαν να χρησιμοποιηθούν από το τμήμα διαχείρισης ανθρώπινου δυναμικού (Human Resources Management – HRM) και το λογιστήριο μιας επιχείρησης, για την έκδοση καταστάσεων μισθοδοσίας. Το πακέτο προγραμμάτων (ή σύστημα λογισμικού ή σύστημα) βασίζεται σε ταξινομημένα σειριακά αρχεία και αξιοποιεί τους αλγορίθμους που αναπτύξαμε, κυρίως, στις προηγούμενες ενότητες. Για την έκδοση καταστάσεων μισθοδοσίας ένα σύστημα λογισμικού θα πρέπει να γνωρίζει τα στοιχεία του προσωπικού, δηλαδή τα σχετικά σταθερά στοιχεία των υπαλλήλων της επιχείρησης (σε αυτά περιλαμβάνεται και ο βασικός μισθός), και τα μεταβλητά στοιχεία που αφορούν κυρίως τις ώρες υπερωριακής απασχόλησης κατά τη διάρκεια της μισθολογικής περιόδου. Αυτά τα στοιχεία, στο σύστημα λογισμικού που παρουσιάζουμε, βρίσκονται αποθηκευμένα σε δύο ταξινομημένα σειριακά αρχεία. Η δομές εγγραφών των αρχείων αυτών ορίζονται στο επόμενο πηγαίο (έστω types.h) αρχείο:

```
/* ************************************************
   * Author: Nikitas N. Karanikolas, Assistant Professor *
   * Creation Date: February 21, 2009                    *
   ************************************************** */

#define keysize          7
#define fnamesize       20
#define snamesize       25
#define specialitysize  30

typedef struct {
   char key[keysize];
   char fname[fnamesize];
```

```
   char sname[snamesize];
   char speciality[specialitysize];
   int year_of_birth;
   int year_of_employment;
   float salary;
} personnel_rec;

typedef struct {
   char key[keysize];
   int overtime_hours;
   int saturday_hours;
   int sunday_hours;
} overtime_rec;
```

Στα στοιχεία του ανθρώπινου δυναμικού, για κάθε υπάλληλο (`personnel_rec`), έχουμε το κλειδί (`key` – το οποίο θα εξηγήσουμε παρακάτω), το όνομα (`fname`), το επώνυμο (`sname`), την ειδικότητα (`speciality`), το έτος γέννησης (`year_of_birth`), το έτος πρόσληψης (`year_of_employment`) και το βασικό μισθό (`salary`). Σαφέστατα θα μπορούσαμε να έχουμε ενσωματώσει πολύ περισσότερα στοιχεία στο `personnel_rec`, όπως για παράδειγμα σπουδές, αριθμός τέκνων, κλπ, αλλά αυτά δεν θα προσέθεταν τίποτα περισσότερο στην κατανόηση του προβλήματος και στην προσέγγιση της λύσης. Ο επόμενος πίνακας αποτελεί παράδειγμα οπτικοποίησης του αρχείου προσωπικού:

key	fname	sname	speciality	year_of _birth	year_of _employment	salary
ΓΒ7501	Γεώργιος	Βασιλείου	Χειριστής	1975	2001	950.00
ΔΠ6699	Δημήτρης	Παππάς	Προγραμματιστής	1966	1999	1250.00

Στα στοιχεία της απασχόλησης, για κάθε υπάλληλο πέραν της βασικής, κατά τη διάρκεια της μισθολογικής περιόδου (`overtime_rec`), έχουμε το κλειδί (`key`), τις συνολικές ώρες που απασχολήθηκε υπερωριακά σε εργάσιμες ημέρες (`overtime_hours`), τις συνολικές ώρες που απασχολήθηκε τις ημέρες του Σαββάτου (`saturday_hours`) και τις συνολικές ώρες που απασχολήθηκε τις ημέρες της Κυριακής (`sunday_hours`). Ο επόμενος πίνακας αποτελεί παράδειγμα οπτικοποίησης του αρχείου υπερωριακής απασχόλησης:

key	overtime_hours	saturday_hours	sunday_hours
ΓΒ7501	20	10	0
ΔΠ6699	12	14	4

Το κλειδί (`key`), που έχουμε χρησιμοποιήσει σε αμφότερες τις εγγραφές, είναι ένα τεχνητό κατασκεύασμα που χρησιμοποιούμε τόσο για την ταξινόμηση (διάταξη) των εγγραφών των αρχείων όσο και σαν ταυτότητα που αντιπροσωπεύει μοναδικά κάθε υπάλληλο. Το κλειδί δημιουργείται από τον πρώτο χαρακτήρα του επωνύμου, τον πρώτο χαρακτήρα του ονόματος, τα δύο τελευταία ψηφία του έτους γέννησης και τα δύο τελευταία ψηφία του έτους πρόσληψης. Οπως αναφέραμε και πριν, σκοπός του κλειδιού είναι να ταυτοποιεί μοναδικά κάθε υπάλληλο. Θεωρούμε ότι ο τρόπος δημιουργίας του εξασφαλίζει ότι δεν θα υπάρχουν δύο υπάλληλοι με το ίδιο κλειδί, τουλάχιστο στο εύρος των υπαλλήλων μιας εταιρείας.

Η κύρια ιδέα στην οποία στηρίζεται το σύστημα λογισμικού που παρουσιάζουμε είναι ο συνδυασμός έξι προγραμμάτων τόσο για τη συντήρηση του αρχείου ανθρώπινου δυναμικού (προσωπικού), όσο και για την περιοδική (ανά μισθολογική περίοδο) έκδοση των καταστάσεων μισθοδοσίας. Τα προγράμματα αυτά και ο σκοπός τους ακολουθούν:

Σκοπός προγράμματος CREATE:

- για τη δημιουργία του βασικού αρχείου προσωπικού (τα σχετικά σταθερά στοιχεία των υπαλλήλων), και
- για τη δημιουργία του αρχείου των επιπλέων (νέων) υπαλλήλων (τα σχετικά σταθερά στοιχεία των νέων υπαλλήλων).

Σκοπός προγράμματος SORT:

- για τη διάταξη (ταξινόμηση) του βασικού αρχείου προσωπικού με βάση την τιμή του πεδίου key, και
- για τη διάταξη (ταξινόμηση) του αρχείου νέων υπαλλήλων με βάση την τιμή του πεδίου key.

Σκοπός προγράμματος MERGE:

- για τη συγχώνευση του βασικού αρχείου προσωπικού και του αρχείου νέων υπαλλήλων σε ένα νέο βασικό αρχείο προσωπικού, που διατηρεί τη διάταξη (ταξινόμηση) με βάση την τιμή του πεδίου key.

Σκοπός προγράμματος TRNCREAT:

- για τη δημιουργία των εγγραφών συναλλαγών (transactions) που περιέχουν τα μεταβλητά στοιχεία (ώρες υπερωριακής απασχόλησης κατά τη διάρκεια της μισθολογικής περιόδου), για όσους υπάλληλους εργάσθηκαν επιπλέον της βασικής (υποχρεωτικής) τους απασχόλησης.

Σκοπός προγράμματος TRNSORT:

- για τη διάταξη (ταξινόμηση) του αρχείου συναλλαγών (transactions).

Σκοπός προγράμματος PAYROLL:

- για την έκδοση μισθολογικών καταστάσεων με βάση το (ενημερωμένο) βασικό αρχείο προσωπικού και το αρχείο συναλλαγών.

Επίσης, στο σχεδιασμό των προγραμμάτων, ελήφθηκε η απόφαση τα ονόματα των αρχείων, στα οποία επενεργούν ή παραγάγουν, να προσδιορίζονται ως παράμετροι των εκτελέσιμων προγραμμάτων (command line parameters) και να μην είναι κωδικοποιημένα στα πηγαία προγράμματα. Το θέμα αυτό (παράμετροι των εκτελέσιμων προγραμμάτων) αναπτύσεται ξεχωριστά στο κεφάλαιο 15 (Επικοινωνία με το λειτουργικό σύστημα). Με βάση το σκοπό των προγραμμάτων και την τελευταία παρατήρηση μπορούμε να παρουσιάσουμε το σενάριο εκτέλεσης των προγραμμάτων για τη δημιουργία και συντήρηση του αρχείου προσωπικού:

1. CREATE MASTER.DAT
2. SORT MASTER.DAT
3. CREATE NEWEMPL.DAT
4. SORT NEWEMPL.DAT
5. MERGE MASTER.DAT NEWEMPL.DAT MASTER2.DAT

και την έκδοση της μισθολογικής κατάστασης μιας περιόδου:

6. TRNCREAT TRANSACT.DAT
7. TRNSORT TRANSACT.DAT
8. PAYROLL MASTER2.DAT TRANSACT.DAT >PAYROLL.LST

Τα προγράμματα CREATE, SORT, TRNCREAT και TRNSORT δέχονται μία παράμμετρο (command line parameter) και δεν απαιτείται άλλη ερμηνεία. Για τα άλλα δύο προγράμματα διευκρινίζουμε ότι: το πρόγραμμα MERGE δέχεται τρεις παραμέτρους, με την πρώτη προσδιορίζεται το όνομα του αρχείου που περιέχει τα προϋπάρχοντα ταξινομημένα δεδομένα, με τη δεύτερη προσδιορίζεται το όνομα του αρχείου που περιέχει τα νέα (επίσης ταξινομημένα) δεδομένα και με την τρίτη παράμετρο προσδιορίζεται ποιο θα είναι το όνομα του νέου αρχείου στο οποίο θα αποθηκευτούν τα συγχωνευμένα δεδομένα. Το πρόγραμμα PAYROLL δέχεται δύο παραμέτρους, με την πρώτη προσδιορίζεται το όνομα του αρχείου με τα στοιχεία προσωπικού και με την δεύτερη παράμετρο προσδιορίζεται το όνομα του αρχείου που περιέχει τις συναλλαγές (υπερωρίες υπαλλήλων). Η έξοδος του PAYROLL είναι το τερματικό και απαιτείται ανακατεύθυνση εξόδου αν θέλουμε να αποθηκεύσουμε την έξοδο σε αρχείο αποτελεσμάτων (όπως το PAYROLL.LST στην εντολή 8).

Η λογική που διέπει τα προγράμματα CREATE, SORT, TRNCREAT και TRNSORT έχει αναπτυχθεί στα προηγούμενα κεφάλαια και δεν θα μας απασχολήσει ιδιαίτερα. Η λογική του προγράμματος MERGE αναπτύχθηκε στη προηγούμενη ενότητα και παρουσιάσθηκαν δύο εκδόσεις του σχετικού αλγορίθμου (όταν η γλώσσα υλοποίησης υποστηρίζει την πρόβλεψη του τέλους αρχείου και όταν δεν την υποστηρίζει). Η λογική του PAYROLL, όπως θα δούμε, είναι παρόμοια με τη λογική του MERGE. Στη συνέχεια θα παραθέσουμε τα έξι πηγαία προγράμματα και θα σχολιάσουμε ότι επιπλέον απαιτείται για καθένα από τα πηγαία. Επίσης θα εντοπίσουμε κάποιες αδυναμίες που έχει η παρούσα υλοποίηση.

Πηγαίο πρόγραμμα CREATE.C:

```
/* ************************************************
   * Author: Nikitas N. Karanikolas, Assistant Professor *
   * Creation Date: February 21, 2009                *
   ************************************************ */
```

```
#include <stdio.h>
#include "types.h"
#include "commonf.c"

main (int argc, char *argv[]) {
   personnel_rec e1;
   char tmp_str[snamesize];
   int i, k, howmany;
   FILE *fp;

   if (argc<2) {
      printf("You should provide the name of the file "
             "to be created.");
      exit(1);
   }

   fp=fopen(argv[1],"wb");
   if (fp==NULL) {
      printf("Can not open [%s] for writing data.", argv[1]);
      exit(2);
   }

   do {
      blank(&e1,sizeof(personnel_rec));
      printf("\nGive surname: ");
      gets(e1.sname);
      printf("[%s]\n",e1.sname);
      strcpy(tmp_str,e1.sname);
      upper(tmp_str);
      if (strcmp(tmp_str,"QUIT")==0) break;
      printf("\nGive first name: ");
      gets(e1.fname);
      printf("[%s]\n",e1.fname);
      printf("\nGive the speciality: ");
      gets(e1.speciality);
      printf("[%s]\n",e1.speciality);
      printf("\nGive \"year of birth\", \"year of "
             "employment\", \"salary\": ");
      scanf("%d %d %f", &e1.year_of_birth,
            &e1.year_of_employment, &e1.salary);
      gets(tmp_str); /* garbage collection */
      /* create a unique key for the employee */
      sprintf(e1.key, "%c%c%2d%2d", e1.sname[0], e1.fname[0],
         e1.year_of_birth % 100, e1.year_of_employment %100);
      /* convert "xy 5 9" to "xy05 9" */
```

```
      if (e1.key[2]==' ') e1.key[2]='0';
      /* convert "xy05 9" to "xy0509" */
      if (e1.key[4]==' ') e1.key[4]='0';
      /* convert "xy0509" to "XY0509" */
      upper(e1.key);
      printf("the new employee is assigned the key "
             "[%s]\n",e1.key);
      howmany=fwrite(&e1,sizeof(personnel_rec),1,fp);
      /* printf("\nhowmany=%d\n", howmany); */
   } while (1);
   fclose(fp);
}
```

Το πρόγραμμα αυτό ακολουθεί τη δομή του προγράμματος create.c του κεφαλαίου 10, τη δομή του προγράμματος cr_cat2.c του κεφαλαίου 9 και τη δομή του προγράμματος cr_cat.c του κεφαλαίου 8. Η πρώτη (ουσιαστική) διαφορά που παρουσιάζει βασίζεται στην επόμενη εντολή:

```
fp=fopen(argv[1],"wb");
```

Η εντολή αυτή δεν ανοίγει ένα αρχείο του οποίου το όνομα είναι κωδικοποιημένο στο πηγαίο πρόγραμμα αλλά ένα αρχείο που δίδεται ως πρώτη παράμετρος (μετά το όνομα του εκτελέσιμου προγράμματος) στην εντολή που υποβάλει ο χρήστης από το λειτουργικό σύστημα (αναλυτικές πληροφορίες για τις παραμέτρους προγραμμάτων υπάρχουν στο κεφάλαιο 15).

Η δεύτερη (ουσιαστική) διαφορά που παρουσιάζει, το προηγούμενο πηγαίο πρόγραμμα, βασίζεται στην επόμενη ακολουθία εντολών που κατασκευάζουν ένα πεδίο με βάση τις τιμές άλλων πεδίων:

```
      sprintf(e1.key, "%c%c%2d%2d", e1.sname[0], e1.fname[0],
      e1.year_of_birth % 100, e1.year_of_employment %100);
      /* convert "xy 5 9" to "xy05 9" */
      if (e1.key[2]==' ') e1.key[2]='0';
      /* convert "xy05 9" to "xy0509" */
      if (e1.key[4]==' ') e1.key[4]='0';
```

Η εντολή sprintf έχει παρόμοια σύνταξη με την fprintf, διαφοροποιείται μόνο ως προς το πρώτο της όρισμα, και η επικεφαλίδα της είναι:

int sprintf (char *buffer, char *format, argument1, argument2, …);

Λειτουργεί με τον ίδιο ακριβώς τρόπο που λειτουργεί και η printf με τη διαφορά ότι, αντί να γράφει στο standard output, η έξοδός της καταγράφεται στο πρώτο όρισμα της (στο αλφαριθμητικό buffer).

Πηγαίο πρόγραμμα SORT.C:

```
/* ****************************************************
   * Author: Nikitas N. Karanikolas, Assistant Professor *
   * Creation Date: February 21, 2009                    *
   **************************************************** */

#include <stdio.h>
#include "types.h"

int partial_sort(FILE *fp1, FILE *fp2) {
   personnel_rec e1, e2;
   int howmany;
   int sorted=1; /* sorted <- true */
   howmany=fread(&e1,sizeof(personnel_rec),1,fp1);
   if (howmany<1)
      return sorted;
   do {
      howmany=fread(&e2,sizeof(personnel_rec),1,fp1);
      if (howmany<1) {
         fwrite(&e1,sizeof(personnel_rec),1,fp2);
         break;
      }
      if (strcmp(e1.key,e2.key)<=0) {
         fwrite(&e1,sizeof(personnel_rec),1,fp2);
         memcpy(&e1,&e2,sizeof(personnel_rec));
      }
      else {
         fwrite(&e2,sizeof(personnel_rec),1,fp2);
         sorted=0; /* sorted <- false */
      }
   } while (1);
   return sorted;
}

main (int argc, char *argv[]) {
   int i=0;
   int sorted;
   FILE *fp1, *fp2;
```

```
   if (argc<2) {
      printf("You should provide the name of the file "
             "to be created.");
      exit(1);
   }

   do {
      if (i % 2 == 0) {
         fp1=fopen(argv[1],"rb");
         fp2=fopen("tmp$$.tmp","wb");
      }
      else {
         fp1=fopen("tmp$$.tmp","rb");
         fp2=fopen(argv[1],"wb");
      }
      sorted=partial_sort(fp1,fp2);
      fclose(fp1);
      fclose(fp2);
      i++;
   } while (!sorted);
   printf("\nNumber of file traverses: %d\n", i);
   if (i % 2 == 1) {
      /* delete the original file */
      unlink(argv[1]);
      /* rename the temporary file to the original */
      rename("tmp$$.tmp",argv[1]);
   }
   else {
      /* delete temporary file */
      unlink("tmp$$.tmp");
   }
}
```

Το πρόγραμμα αυτό ακολουθεί τη δομή του προγράμματος sort.c του κεφαλαίου 10. Η πρώτη (ουσιαστική) διαφορά που παρουσιάζει βασίζεται στις επόμενες δύο εντολές:

```
         fp1=fopen(argv[1],"rb");
         ...
         ...
         fp2=fopen(argv[1],"wb");
```

Οι εντολές αυτες δεν ανοίγουν ένα αρχείο του οποίου το όνομα είναι κωδικοποιημένο στο πηγαίο πρόγραμμα αλλά ένα αρχείο που δίδεται ως πρώτη παράμετρος (μετά το όνομα του εκτελέσιμου προγράμματος) στην εντολή που υποβάλει ο χρήστης από το λειτουργικό σύστημα.

Η δεύτερη διαφορά είναι στις επόμενες δύο εντολές του προγράμματός μας. Αυτές διαγράφουν και μετονομάζουν, αντίστοιχα, ένα αρχείο το όνομα του οποίου (όνομα διαγραφόμενου και νέο όνομα μετονομαζόμενου αρχείου) δεν είναι κωδικοποιημένο στο πηγαίο πρόγραμμα, αλλά δίδεται ως πρώτη παράμετρος (μετά το όνομα του εκτελέσιμου προγράμματος) στην εντολή που υποβάλει ο χρήστης από το λειτουργικό σύστημα:

```
      unlink(argv[1]);
      rename("tmp$$.tmp",argv[1]);
```

Πηγαίο πρόγραμμα TRNCREAT.C:

```
/* ****************************************************
   * Author: Nikitas N. Karanikolas, Assistant Professor *
   * Creation Date: February 21, 2009                    *
   **************************************************** */

#include <stdio.h>
#include "types.h"
#include "commonf.c"

main (int argc, char *argv[]) {
   overtime_rec ot1;
   char tmp_str[keysize];
   int i, k, howmany;
   FILE *fp;

   if (argc<2) {
      printf("You should provide the name of the file "
             "to be created.");
      exit(1);
   }

   fp=fopen(argv[1],"wb");
   if (fp==NULL) {
      printf("Can not open [%s] for writing data.", argv[1]);
      exit(2);
   }
```

```
   do {
      blank(&ot1,sizeof(overtime_rec));
      printf("\nGive key: ");
      gets(ot1.key);
      printf("[%s]\n",ot1.key);
      strcpy(tmp_str,ot1.key);
      upper(tmp_str);
      if (strcmp(tmp_str,"QUIT")==0) break;
      printf("\nGive \"overtime hours\", \"saturday "
             "hours\", \"sunday hours\": ");
      scanf("%d %d %d", &ot1.overtime_hours,
            &ot1.saturday_hours, &ot1.sunday_hours);
      gets(tmp_str); /* garbage collection */
      howmany=fwrite(&ot1,sizeof(overtime_rec),1,fp);
      /* printf("\nhowmany=%d\n", howmany); */
   } while (1);
   fclose(fp);
}
```

Το πρόγραμμα αυτό ακολουθεί τη δομή του προγράμματος create.c του κεφαλαίου 10, τη δομή του προγράμματος cr_cat2.c του κεφαλαίου 9 και τη δομή του προγράμματος cr_cat.c του κεφαλαίου 8. Η μοναδική (ουσιαστική) διαφορά που παρουσιάζει, σε σχέση με τα προγράμματα των προηγούμενων κεφαλαίων, είναι ότι βασίζεται στην παραμετρική υποβολή του ονόματος του αρχείου που θα δημιουργηθεί (από την εντολή που υποβάλλει ο χρήστης από το λειτουργικό σύστημα).

Πηγαίο πρόγραμμα TRNSORT.C:

```
/* ************************************************
   * Author: Nikitas N. Karanikolas, Assistant Professor *
   * Creation Date: February 21, 2009                    *
   ************************************************** */

#include <stdio.h>
#include "types.h"

int partial_sort(FILE *fp1, FILE *fp2) {
   overtime_rec ot1, ot2;
   int howmany;
   int sorted=1; /* sorted <- true */
   howmany=fread(&ot1,sizeof(overtime_rec),1,fp1);
```

```
   if (howmany<1)
      return sorted;
   do {
      howmany=fread(&ot2,sizeof(overtime_rec),1,fp1);
      if (howmany<1) {
         fwrite(&ot1,sizeof(overtime_rec),1,fp2);
         break;
      }
      if (strcmp(ot1.key,ot2.key)<=0) {
         fwrite(&ot1,sizeof(overtime_rec),1,fp2);
         memcpy(&ot1,&ot2,sizeof(overtime_rec));
      }
      else {
         fwrite(&ot2,sizeof(overtime_rec),1,fp2);
         sorted=0; /* sorted <- false */
      }
   } while (1);
   return sorted;
}

main (int argc, char *argv[]) {
   int i=0;
   int sorted;
   FILE *fp1, *fp2;

   if (argc<2) {
      printf("You should provide the name of the file "
             "to be created.");
      exit(1);
   }

   do {
      if (i % 2 == 0) {
         fp1=fopen(argv[1],"rb");
         fp2=fopen("tmp$$$.tmp","wb");
      }
      else {
         fp1=fopen("tmp$$$.tmp","rb");
         fp2=fopen(argv[1],"wb");
      }
      sorted=partial_sort(fp1,fp2);
      fclose(fp1);
      fclose(fp2);
      i++;
   } while (!sorted);
   printf("\nNumber of file traverses: %d\n", i);
```

```
   if (i % 2 == 1) {
      /* delete the original file */
      unlink(argv[1]);
      /* rename the temporary file to the original */
      rename("tmp$$$.tmp",argv[1]);
   }
   else {
      /* delete temporary file */
      unlink("tmp$$$.tmp");
   }
}
```

Το πρόγραμμα αυτό λειτουργεί με τον ίδιο ακριβώς τρόπο που λειτουργεί και το sort.c που παρουσιάσαμε λίγο παραπάνω. Στο μόνο που διαφέρουν τα δύο προγράμματα είναι ότι το παρόν πρόγραμμα ταξινομεί δεδομένα τύπου `overtime_rec` αντί για δεδομένα `personnel_rec` που ταξινομούσε το sort.c πρόγραμμα.

Πηγαίο πρόγραμμα MERGE.C:

```
/* ******************************************************
   * Author: Nikitas N. Karanikolas, Assistant Professor *
   * Creation Date: February 21, 2009                    *
   ****************************************************** */

#include <stdio.h>
#include "types.h"

main (int argc, char *argv[]) {
   personnel_rec e1, e2;
   int eof1, eof2;
   FILE *fp1, *fp2, *fp3;

   if (argc<4) {
      printf("You should provide:\n1. the name of "
             "the existing master file");
      printf("\n2. the name of the file with the "
             "new records\n3. the new master file.");
      exit(1);
   }
   fp1=fopen(argv[1],"rb");
   if (fp1==NULL) {
      printf("Can not open [%s] for reading.\n",argv[1]);
      exit(2);
```

```
}
fp2=fopen(argv[2],"rb");
if (fp2==NULL) {
   printf("Can not open [%s] for reading.\n",argv[2]);
   fclose(fp1);
   exit(3);
}
fp3=fopen(argv[3],"wb");
if (fp3==NULL) {
   printf("Can not open [%s] for writing.\n",argv[3]);
   fclose(fp1);
   fclose(fp2);
   exit(4);
}

eof1=fread(&e1,sizeof(personnel_rec),1,fp1)<1;
eof2=fread(&e2,sizeof(personnel_rec),1,fp2)<1;
while ((!eof1) && (!eof2))
   if (strcmp(e1.key, e2.key)<0) {
      fwrite(&e1,sizeof(personnel_rec),1,fp3);
      eof1=fread(&e1,sizeof(personnel_rec),1,fp1)<1;
   }
   else if (strcmp(e1.key, e2.key)>0) {
      fwrite(&e2,sizeof(personnel_rec),1,fp3);
      eof2=fread(&e2,sizeof(personnel_rec),1,fp2)<1;
   }
   else /* strcmp(e1.key, e2.key)==0 */ {
      /* abnormal case, in case that the new master file
         should have unique keys */
      printf("Already exist a record with key [%s] in "
             "the existing master file [%s]. ",
             e2.key, argv[1]);
      printf("The record of the file with the new "
             "records [%s] is discarded.\n", argv[2]);
      eof2=fread(&e2,sizeof(personnel_rec),1,fp2)<1;
   }
while (!eof1) {
   fwrite(&e1,sizeof(personnel_rec),1,fp3);
   eof1=fread(&e1,sizeof(personnel_rec),1,fp1)<1;
}
while (!eof2) {
   fwrite(&e2,sizeof(personnel_rec),1,fp3);
   eof2=fread(&e2,sizeof(personnel_rec),1,fp2)<1;
}

fclose(fp1);
```

```
   fclose(fp2);
   fclose(fp3);
}
```

Το πρόγραμμα merge.c ακολουθεί σχεδόν πιστά τον αλγόριθμο που παρουσιάσαμε στο τέλος της προηγούμενης ενότητας. Δεν χρησιμοποιεί τη συνάρτηση feof αλλά επιτυγχάνει το ίδιο αποτέλεσμα εξετάζοντας την τιμή που επιστρέφει η συνάρτηση fread (και σε αυτή την περίπτωση έχουμε καθυστερημένη διαπίστωση του τέλους αρχείου). Η σημαντικότερη όμως διαφορά σε σχέση με τον αλγόριθμο, κατά την παράλληλη σάρωση των δύο αρχείων, εμφανίζεται όταν οι δύο εγγραφές έχουν το ίδιο κλειδί. Στην περίπτωση αυτή αγνοούμε την εγγραφή του δεύτερου αρχείου (νέες εγγραφές) καθώς δεν επιτρέπουμε να υπάρχουν δύο υπάλληλοι με το ίδιο κλειδί.

Πηγαίο πρόγραμμα PAYROLL.C:

```
/* ****************************************************
   * Author: Nikitas N. Karanikolas, Assistant Professor *
   * Creation Date: February 21, 2009                    *
   **************************************************** */

#include <stdio.h>
#include "types.h"

#define overtime_multiplyer 1.50
#define saturday_multiplyer 1.75
#define sunday_multiplyer   2.00

void employee_payroll(personnel_rec emp, int overtime_hours,
                      int saturday_hours, int sunday_hours) {
   int i;
   char tmp_str[snamesize+1];
   float total = emp.salary +
      emp.salary/175*overtime_hours*overtime_multiplyer +
      emp.salary/175*saturday_hours*saturday_multiplyer +
      emp.salary/175*sunday_hours*sunday_multiplyer;
   printf("%s ",emp.key);
   strcpy(tmp_str,  emp.sname);
   for (i=strlen(emp.sname); i<snamesize; i++)
      tmp_str[i]=' ';
   tmp_str[snamesize]='\0';
   printf("%s ",tmp_str);
   strcpy(tmp_str,  emp.fname);
```

```
   for (i=strlen(emp.fname); i<fnamesize; i++)
      tmp_str[i]=' ';
   tmp_str[fnamesize]='\0';
   printf("%s ",tmp_str);
   printf("%8.2f\n", emp.salary);
   printf("%32c Overtime hours: %2d\n", ' ', overtime_hours);
   printf("%32c Saturday hours: %2d\n", ' ', saturday_hours);
   printf("%32c Sunday hours  : %2d\n", ' ', sunday_hours);
   printf("%53c %8.2f\n\n", ' ', total);
}

main (int argc, char *argv[]) {
   personnel_rec emp;
   overtime_rec ot;
   int eof1, eof2;
   FILE *fp1, *fp2;

   if (argc<3) {
      fprintf(stderr,"You should provide:\n1. the name of "
      "the existing master file");
      fprintf(stderr,"\n2. the name of the overtime "
      "transactions file.\n");
      exit(1);
   }
   fp1=fopen(argv[1],"rb");
   if (fp1==NULL) {
      fprintf(stderr,"Can not open [%s] for reading.\n",
      argv[1]);
      exit(2);
   }
   fp2=fopen(argv[2],"rb");
   if (fp2==NULL) {
      fprintf(stderr,"Can not open [%s] for reading.\n",
      argv[2]);
      fclose(fp1);
      exit(3);
   }

   eof1=fread(&emp,sizeof(personnel_rec),1,fp1)<1;
   eof2=fread(&ot,sizeof(overtime_rec),1,fp2)<1;
   while ((!eof1) && (!eof2))
      if (strcmp(emp.key, ot.key)<0) {
         employee_payroll(emp,0,0,0);
         eof1=fread(&emp,sizeof(personnel_rec),1,fp1)<1;
      }
      else if (strcmp(emp.key, ot.key)==0) {
```

```
            employee_payroll(emp,ot.overtime_hours,
            ot.saturday_hours,ot.sunday_hours);
            eof1=fread(&emp,sizeof(personnel_rec),1,fp1)<1;
            eof2=fread(&ot,sizeof(overtime_rec),1,fp2)<1;
         }
         else /* strcmp(emp.key, ot.key)>0 */ {
            /* abnormal case */
            fprintf(stderr,"There is an overtime transaction "
            "with key [%s] that does not corresponds to an "
            "employee. ", ot.key);
            fprintf(stderr,"This overtime transaction will be "
            "ignored.\n");
            eof2=fread(&ot,sizeof(overtime_rec),1,fp2)<1;
         }
      while (!eof1) {
         employee_payroll(emp,0,0,0);
         eof1=fread(&emp,sizeof(personnel_rec),1,fp1)<1;
      }

      fclose(fp1);
      fclose(fp2);
}
```

Το πρόγραμμα αυτό θυμίζει επίσης τον τελευταίο αλγόριθμο της προηγούμενης ενότητας. Αποτελείται από δύο επαναληπτικά σχήματα (σε αντίθεση με τον αλγόριθμο της προηγούμενης ενότητας που είχε τρία επαναλληπτικά σχήματα). Το πρώτο επαναληπτικό σχήμα κάνει μία παράλληλη σάρωση του κύριου αρχείου και του αρχείου μεταβολών. Σε κάθε βήμα της επανάληψης, αν το κλειδί της εγγραφής του υπαλλήλου είναι μικρότερο από το κλειδί της συναλλαγής (ώρες επιπλέον από τη βασική απασχόληση) τότε διαβιβάζει την εγγραφή του υπαλλήλου στη ρουτίνα `employee_payroll` για να εκτυπώσει την μισθοδοσία του υπαλλήλου, χωρίς επιπλέον ώρες από τη βασική του απασχόληση. Αντίθετα, αν οι δύο συγκρινόμενες εγγραφές (υπαλλήλων και συναλλαγών) έχουν την ίδια τιμή στο κλειδί τότε διαβιβάζει την εγγραφή του υπαλλήλου καθώς και τα στοιχεία της εγγραφής συναλλαγής (ώρες κάθε κατηγορίας υπερωρίας) στη ρουτίνα `employee_payroll` για να υπολογίσει και να εκτυπώσει την μισθοδοσία του υπαλλήλου. Η περίπτωση που το κλειδί της εγγραφής του υπαλλήλου είναι μεγαλύτερο από το κλειδί της συναλλαγής μπορεί να οφείλεται μόνο σε λάθος (καταχώρηση ανύπαρκτου κωδικού υπαλλήλου κατά την εισαγωγή μίας συναλλαγής ή διπλοεισαγωγή μιας συναλλαγής) και παραλείπεται (εκτυπώ-

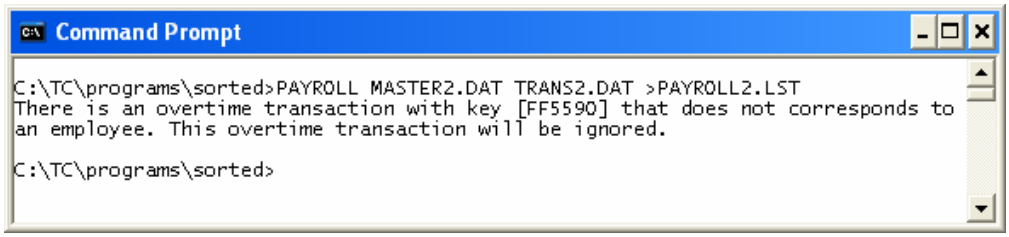
νεται μήνυμα λάθους). Το δεύτερο επαναληπτικό σχήμα σαρώνει τους υπόλοιπους υπαλλήλους (που δεν σαρώθηκαν κατά το πρώτο επαναλληπτικό σχήμα) και για καθένα από αυτούς καλεί τη ρουτίνα `employee_payroll` για να εκτυπώσει την μισθοδοσία τους, χωρίς επιπλέον ώρες υπερωριακής απασχόλησης.

Σε αντίθεση με τον τελευταίο αλγόριθμο της προηγούμενης ενότητας, το πρόγραμμα payroll.c αγνοεί τις υπόλοιπες συναλλαγές (που δεν σαρώθηκαν κατά το πρώτο επαναληπτικό σχήμα) καθώς αυτές δεν πρέπει να υπάρχουν και αν τελικά υπάρχουν είναι αποτέλεσμα λανθασμένων καταχωρήσεων συναλλαγών.

Μία άλλη χρήσιμη τεχνική είναι να διαχωρίζουμε τη «σωστή» έξοδο από τα μηνύματα λάθους. Αυτό γίνεται στις εντολές εκτύπωσης που έχουν τη μορφή:

```
fprintf(stderr, ...);
```

Στις εντολές αυτές στέλνουμε την έξοδο στο προκαθορισμένο αρχείο λαθών (standard error, `stderr`). Αυτή η τακτική είναι πολύ χρήσιμη γιατί μας επιτρέπει να διαχωρίσουμε τα μηνύματα λάθους από την αναμενόμενη έξοδο (στο παράδειγμά μας από την κατάσταση μισθοδοσίας). Ακολουθεί ένα τέτοιο παράδειγμα:



Το βοηθητικό πηγαίο αρχείο COMMONF.C:

```
/* ******************************************************
   * Author: Nikitas N. Karanikolas, Assistant Professor *
   * Creation Date: February 21, 2009                    *
   ****************************************************** */

void upper(char *str) {
   int i;
```

```
   for (i=0; i<strlen(str); i++)
      if ((str[i] >= 97) && (str[i] <= 122))
         str[i] = str[i] - 32;
}

void blank(char buffer[], int size) {
   int i;
   for (i=0; i<size; i++) buffer[i]=0;
}
```

Το τελευταίο αρχείο (commonf.c) χρησιμοποιείται από άλλα προγράμματα της μισθοδοσίας (από το create.c και trncreat.c).

Αδυναμίες υλοποίησης:

Το πακέτο προγραμμάτων μισθοδοσίας, στην παρούσα μορφή του, παρουσιάζει ορισμένες αδυναμίες της οποίες πρέπει να αναδείξουμε. Όπως έχουμε αναφέρει, στα δεδομένα χρησιμοποιούμε ένα κλειδί (το πεδίο key στις εγγραφές personnel_rec και overtime_rec) προκειμένου να βασιζόμαστε σε αυτό και να διατάσουμε τα αντίστοιχα αρχεία. Επιπλέον τα δεδομένα του αρχείου προσωπικού πρέπει να περιέχουν μόνο μία φορά το κάθε κλειδί. Αυτό εξασφαλίζεται μερικώς καθώς το πρόγραμμα merge δεν επιτρέπει την ενσωμάτωση (στο νέο κύριο αρχείο προσωπικού) ενός νέου υπαλλήλου (από το αρχείο νέων υπαλλήλων), αν το κλειδί του νέου υπαλλήλου υπάρχει στο παλαιό κύριο αρχείο προσωπικού. Όμως μόνο αυτό δεν αρκεί. Θα πρέπει η δημιουργία αρχείου προσωπικού (είτε κατά την αρχική δημιουργία, είτε κατά τη δημιουργία του αρχείου νέων – επιπλέον – υπαλλήλων) να μην επιτρέπει τη διπλή καταχώρηση του ιδίου κλειδιού. Για να είναι πλήρως σωστή η λύση μας θα πρέπει μετά την ταξινόμηση ενός αρχείου προσωπικού (είτε κατά την αρχική δημιουργία, είτε κατά τη δημιουργία του αρχείου νέων – επιπλέον – υπαλλήλων) να γίνεται έλεγχος ώστε να μην υπάρχουν δύο διαδοχικές εγγραφές με το ίδιο κλειδί. Δηλαδή το ζεύγος προγραμμάτων create και sort πρέπει να συνοδεύεται από ένα πρόγραμμα ελέγχου μοναδικών κλειδιών. Ανάλογη προσέγγιση πρέπει να ακολουθείται και για το αρχείο των συναλλαγών. Δηλαδή, θα πρέπει το ζεύγος προγραμμάτων trncreat και trnsort να συνοδεύεται από ένα πρόγραμμα ελέγχου μοναδικών κλειδιών.