Προγραμματισμός Υπολογιστών

Εφαρμογή με δομή δεδομένων

Νικόλαος Ζ. Ζάχαρης
Καθηγητής

Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Παράδειγμα

Να κατασκευάσετε ένα πρόγραμμα εξομοίωσης μιας τράπεζας. Η υλοποίηση θα γίνει με την χρήση ενός πίνακα με μέγιστο πλήθος στοιχείων 20, όπου το κάθε ένα είναι μια δομή τύπου Account. Οι ενέργειες που θα μπορούμε να κάνουμε σε ένα Account είναι η κατάθεση (deposit) και η ανάληψη (withdraw) και αντίστοιχα το υπόλοιπο (balance) θα αυξάνεται ή θα μειώνεται. Επίσης η δομή Account θα περιέχει το όνομα του πελάτη.

Οι ενέργειες που θα μπορούμε να κάνουμε στην τράπεζα είναι η προσθήκη ενός νέου λογαριασμού (new account), η διαγραφή ενός λογαριασμού (delete account), μια συναλλαγή με ένα λογαριασμό (transaction), η επίδειξη των στοιχείων (όνομα πελάτη και υπόλοιπο) της τράπεζας και έξοδος από την εφαρμογή.

Η υλοποίηση της εφαρμογής θα έχει την παρακάτω μορφή :

Bank

0	1	2	....	18	19

Account

Όνομα Πελάτη (char name[20])
Υπόλοιπο (float balance)
Ενεργός (State active)
Υλοποίηση της Συναλλαγής() void doTransaction()

Η τράπεζα (Bank) αναπαριστάται σαν ένας πίνακας με μέγεθος 20 θέσεων.

Η κάθε θέση περιέχει ένα στοιχείο τύπου δομής με το όνομα Account. Η δομή εμπεριέχει το όνομα του πελάτη σαν string και το υπόλοιπο σαν float αριθμός. Επίσης υπάρχει η υλοποίηση της Συναλλαγής, όπου ο χρήστης της εφαρμογής επιλέγει την συναλλαγή, πληκτρολογεί το ποσό και ανάλογα με το είδος της συναλλαγής το ποσό προστίθεται ή αφαιρείται από το Υπόλοιπο.

Η κατάσταση της τρέχουσας θέσης του πίνακα δηλώνεται με την active η οποία είναι τύπου enum και παίρνει δύο τιμές (TRUE, FALSE). Αρχικά όλες οι θέσεις του πίνακα έχουν την active με τιμή FALSE. Εάν προσθέσουμε (δημιουργήσουμε) ένα νέο λογαριασμό τότε θα τον τοποθετήσουμε στην

πρώτη θέση του πίνακα που η state είναι FALSE – αυτή η θέση συμπίπτει με την πρώτη θέση του πίνακα – και φυσικά μετά την δημιουργία θα πρέπει να αλλάξουμε την τιμή της state από FALSE σε TRUE.

Αν επαναλάβουμε ακόμα 3 φορές την ενέργεια προσθήκη νέου λογαριασμού στο πίνακα τότε η τράπεζα θα έχει την παρακάτω μορφή.

0	1	2	3	4	....	18	19
TRUE	TRUE	TRUE	TRUE	FALSE		FALSE	FALSE

Άρα η active μας εξυπηρετεί να γνωρίζουμε ποίες θέσεις του πίνακα είναι ενεργές. Εάν για παράδειγμα επιθυμούμε να διαγράψουμε την 2 θέση του πίνακα τότε δεν χρειάζεται να κάνουμε τίποτα άλλο παρά να τροποποιήσουμε την τιμή της active της θέσης 2 από TRUE σε FALSE και να έχουμε :

0	1	2	3	4	....	18	19
TRUE	TRUE	FALSE	TRUE	FALSE		FALSE	FALSE

Αν θέλαμε να βρούμε το σύνολο των χρημάτων της τράπεζας θα έπρεπε να προσθέσουμε τα υπόλοιπα των λογαριασμών που είναι ενεργοί (active = TRUE) δηλαδή τα υπόλοιπα των θέσεων 0 και 1 και 3.

Αν στην συνέχεια προσθέσουμε ένα νέο λογαριασμό θα τον τοποθετήσουμε στην θέση 2 γιατί αυτή είναι η πρώτη θέση στο πίνακα με τιμή active = FALSE.

Αν σε περίπτωση θέλουμε να προσθέσουμε ένα νέο λογαριασμό και όλες οι θέσεις του πίνακα έχουν τιμή active = TRUE τότε θα εμφανίσουμε το μήνυμα ότι η ενέργεια προσθήκη δεν ήταν επιτυχής λόγω του ότι δεν υπάρχουν ελεύθερες θέσεις στο πίνακα.

#define noCustome

typedef enum {FALS

Ο κώδικας της εφαρμογής

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

#define noCustomers 20

typedef enum {FALSE, TRUE} State;

typedef struct    {
    char name[20];
    double balance;
    State active;
} Account;

Account Bank[noCustomers];
```

Δηλώνουμε το πλήθος των θέσεων σαν σταθερά.

Οι δύο τιμές που περιγράφουν την κατάσταση της κάθε θέσης του πίνακα

Το όνομα του πελάτη
Το υπόλοιπο του λογαριασμού
Η κατάσταση του τρέχοντος λογαριασμού.

```c
double readDNumber() {
    double x;

    char line[100];
    gets(line);

    return atof(line);
}

int readINumber() {
    double x;

    char line[100];
    gets(line);

    return atoi(line);
}

char getch()
{
    char line[100];
    gets(line);
    return line[0];
}
```

Υλοποίηση της Συναλλαγής

```
void doTransaction(int thesi)
{
        char ch;
        double amount;
        printf("\nChoose (D)eposit (W)ithdraw (E)xit  :  ");
        ch = getch();
        switch(ch)   {
            case 'e' :
            case 'E' :
                        break;
            case 'd' :
            case 'D' :
                        printf("Amount : ");
                        amount = readDNumber();
                        Bank[thesi].balance += amount;
                        break;
            case 'w' :
            case 'W' :
                        printf("Amount : ");
                        amount = readDNumber();
                        if(Bank[thesi].balance - amount >= 0)
                           Bank[thesi].balance -= amount;
                        break;
            default:
              printf("Error : %c is not a valid option\n", ch);
        }
}
```

Ανάλογα με το είδος της συναλλαγής προσθέτουμε ή αφαιρούμε το ποσό που πληκτρολόγησε ο χρήστης στο υπόλοιπο του λογαριασμού.

Όλοι οι λογαριασμοί της τράπεζας

```
int showAccounts(void) {
    int i;
    int total = 0;

    printf("\n");
    for(i = 0; i < noCustomers; i++)
      if(Bank[i].active == TRUE) {
         printf("%d ) \t %s \t %lf \n", i , Bank[i].name , Bank[i].balance);
         total = total + 1;
      }

    if(total == 0)  {
        printf("\n\nThe Bank is empty.\n\n");
    }

    return total;
}
```

Η showAccounts επιδεικνύει το όνομα και το υπόλοιπο των ενεργών λογαριασμών. Επίσης επιστρέφει το πλήθος των ενεργών λογαριασμών ώστε να ενημερώσει άλλες συναρτήσεις που την χρησιμοποιούν αν η τράπεζα έχει ή όχι ενεργούς λογαριασμούς. Για παράδειγμα η επόμενη συνάρτηση ListAccounts χρησιμοποιεί την showAccounts.

```
int ListAccounts(void) {
    int i;
    int pos;

    if(showAccounts() > 0)    {
       printf("Choose an account number : ");
       pos = readINumber();

      if(((pos >= 0) && (pos < noCustomers)) && (Bank[pos].active == TRUE)){
          return pos;
       }
      else {
          printf("\nError : %d is not a valid option.\n", pos);
       }
    }
    return -1;
}
```

Η ListAccounts χρησιμοποιεί την showAccounts για να δείξει τα στοιχεία των ενεργών λογαριασμών και αν υπάρχουν ενεργοί λογαριασμοί, δηλαδή η τιμή που επιστρέφει η showAccounts είναι μεγαλύτερη του 0 τότε αλληλεπιδρά με το χρήστη της εφαρμογής ώστε να επιλέξει ένα λογαριασμό. Αν η τιμή που πληκτρολόγησε ο χρήστης είναι αποδεκτή τότε αυτή επιστρέφεται από την συνάρτηση ListAccounts διαφορετικά επιστρέφεται το -1. Με την σειρά της η ListAccounts χρησιμοποιείται από άλλες συναρτήσεις, όπως για παράδειγμα η DeleteAccount με σκοπό την διαγραφή ενός λογαριασμού.

```
void DeleteAccount(void){
    int pos;

    if((pos = ListAccounts()) > -1)
       Bank[pos].active = FALSE;
}
```

Η DeleteAccount χρησιμοποιεί την ListsAccounts για την διαγραφή ενός λογαριασμού. Αφού επιλέξουμε τον λογαριασμό που θέλουμε να διαγράψουμε δεν έχουμε να κάνουμε τίποτα άλλο από το να θέσουμε τη τιμή FALSE στην μεταβλητή active της αντίστοιχης θέσης του πίνακα.

```
void TransAccount(void){
    int pos;

    if((pos = ListAccounts()) > -1)
       doTransaction(pos);
}
```

Η TransAccount λειτουργεί παρόμοια με την DeleteAccount δηλαδή χρησιμοποιεί την ListsAccounts για να κάνει συναλλαγή με ένα λογαριασμό. Αφού επιλέξουμε τον λογαριασμό εν συνεχεία καλούμε την μέθοδο doTransaction.

```
void NewAccount(void) {
    char name[20];
    int i, pos = -1;

    for(i = 0; i < noCustomers; i++)
      if(Bank[i].active == FALSE)  {
         pos = i;
         break;
      }

    if(pos < 0)  {
        printf("\nThe bank array is full.\n");
    }
    else  {
        printf("Type customer name : ");
        scanf("%s", name);
        char ch = getch();

        Bank[pos].active = TRUE;
        Bank[pos].balance = 0.0;
        strcpy(Bank[pos].name, name);

    }
}
```

Εύρεση ελεύθερης θέσης στο πίνακα.

Αν υπάρχει ελεύθερη θέση τότε ο χρήστης πληκτρολογεί το όνομα του πελάτη. Ο λογαριασμός στην αντίστοιχη θέση του πίνακα αρχικοποιείται με το ανωτέρω όνομα, υπόλοιπο 0 και την δήλωση ότι είναι ενεργός (active = TRUE)

```c
int DisplayMenu(void) {
    char ch, temp;
    int  t, ans = -1;

    printf("(N) New Account\n");
    printf("(D) Delete Account\n");
    printf("(T) Transaction\n");
    printf("(L) List Accounts\n");;
    printf("(E) Exit\n");
    printf("Type your choice : ");
    ch = getch();
    switch(ch)  {
            case 'n' :
            case 'N' : NewAccount();  break;
            case 'd' :
            case 'D' : DeleteAccount();   break;
            case 't' :
            case 'T' : TransAccount();   break;
            case 'l' :
            case 'L' : t = showAccounts();   break;
            case 'e' :
            case 'E' :  ans = 1;     break;
            default:
                printf("Error : %c is not a valid option\n", ch);
       }
       return ans;
}
```

```
void initializeBank(void) {
 int i;
 for(i = 0; i < noCustomers; i++)
     Bank[i].active = FALSE;
}


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

  while(DisplayMenu() < 0) {}

  return 0;
}
```

Η initializeBank αρχικοποιεί όλες τις θέσεις του πίνακα με την τιμή FALSE στην μεταβλητή active. Πρακτικά δηλώνουμε ότι ο πίνακας είναι άδειος.

Η main αρχικοποιεί τον πίνακα και εν συνεχεία τρέχει μια επανάληψη με σκοπό να δείχνει συνέχεια το menu επιλογών. Η DisplayMenu εμφανίζει τις ενέργειες που μπορεί να κάνει ο χρήστης και επιστρέφει πάντα την τιμή -1 εκτός από την περίπτωση που ο χρήστης πατήσει το πλήκτρο E ή e το οποίο σημαίνει έξοδος από το πρόγραμμα και επιστρέφει 1.

Να αναπτύξετε τις παρακάτω τρεις λειτουργίες :

float CalcSum – επιστρέφει το σύνολο των χρημάτων στην τράπεζα

void ShowMax – εμφανίζει τα στοιχεία του λογαριασμού με το μεγαλύτερο υπόλοιπο.

void ShowMin – εμφανίζει τα στοιχεία του λογαριασμού με το μικρότερο υπόλοιπο.