Dokumentacja projektu nr 3
Baza Danych

Adam Narożniak

grupa środy godzina 14:15-16:00

1 Opis

Użytkownik może:

- dodawać, usuwać oraz edytować rekordy,
- sortować po dowolnym polu
- czyść cała baze.

Rekordy przechowywane sa w pliku tekstowym. Wybór pliku możliwy jest na poczatku programu.

2 Propozycja rozwiazania

Użytkownik podaje plik który chce otworzyć (jeśli plik o tej nazwie nie istniejacej - zostanie utworzony). Nastepnie wybiera: wyświetlanie, dodawanie, usuwanie, sortowanie z menu podajac odpowiedni numer z wyświetlonego menu. Nastepnie użytkownik postepuje analogicznie intuicyjnie odpowiadajac na komunikaty programu.

3 Struktury, tablice

```
#define DL 50
struct typ {
        char typ_szkolenia[DL];
        struct typ * next;
};
typedef struct typ Typ;

typedef Typ *Lista;

struct szkolenie {
        char nazwa[DL];
        char typ_szkolenia[DL];
        double cena;
```

```
        int czas;
        char szkoleniowiec_imie[DL];
        char szkoleniowiec_nazwisko[DL];
        int liczba_uczestnikow;
};
typedef struct szkolenie Szkolenie;

Szkolenie** tab_szkolenie
```

- Struktura szkolenie służy do przechowywania pojedynczego rekordu.
- Struktura typ to struktura służaca przechowywaniu typu szkoleń używanych jako pole szkolenia(zaimplementowane jako lista).
- Tablica do przechowywania wskaźników do struktur, przydatna do sortowania.

4 Pliki, funkcje

4.1 main

Służy do wywoływania funkcji;

4.2 tekst

Odpowiada za funkcje wyświetlajace menu, oraz inne funkcje tekstowe, m. in. wypisujace rekordy, jej pola oraz każde komendy;
wybrane funckje:

```
void wyswietl_baze(Szkolenie ** tab_szkolenie,
    int * liczba_szkolen)
```

Funkcja korzysta z tablicy wskaźników do szkoleń, przez co wyświetlanie realizowane jest w petli po indeksach tablicy.

```
void wyswietl_pola_szkolen(void);
void menu_typy(void);
void wyswietl_menu(void);
void wyswietl_typy(Lista *wlista);
```

Ta funkcje odpowiadaja za wyświetlanie nazw pól z odpowiednia numeracja potrzebna do dokonania wyboru.

4.3 obsluga_Typy

Zawiera wszystkie funkcje odpowiadajace za operacje na rekordach np. dodawanie elementu, edytowanie itp.

```
int Dodaj_Typ_Pamiec(char *typ_szkolenia, Lista *wlista)
```

Funckja przydziela pamiec dla typu sturktur. Jako argument przyjmuje wskżnik na łańcuch oraz wskaźnik na wskaźnik na głowe.

```
void Edytuj_Typy(Lista *wlista, int *liczba_typow)
```

Funkcja edytuje podany przez użytkownika typ szkolenia. Nie jest zabronione usuwanie obecnie używanych Typów przez sposób reprezentacji danych. Jako argument przyjmuje wskaźnik na wskaźnik na głowe, oraz liczbe argumentów potrzebna aby zapobiec błedom użytkownika.

```
int Dodaj_Typ_Pamiec(char *typ_szkolenia, Lista *wlista);
```

Funkcja alokuje pamieć na nowy typ szkolenia. Oraz odpowiednio uzależnia powstały element od pozostałych(przydziela elementowi wcześniejszemu wskaźnik na nowy element, a jemu wskaźnik do NULL'a). Zwraca 0 jeśli operacja jest nieudana oraz 1 przy udanej operacji.

```
void Usun_Typ(Lista *wlista, int *liczba_typow)
```

Funkcja usuwa jeden (podany przez użytkownia) typ szkolenia. Po podaniu przez użytkownika przypisanego do niego numeru.

```
int wybierz_typ(Szkolenie ** szkolenie, Lista *wlista,
    int *liczba_typow) {
```

Funkcje umozliwia wybranie jednej z powyższych funkcji. Zabezpiecza przed błednym wprowadzaniem danych.

```
void usun_pamiec_typy(Lista *wlista, int * liczba_typow)
```

Funkcja zwalnia pamieć używana do przechowywania Typów szkoleń. Oraz zeruje dana: liczba typów.

4.4 obsluga_Szkolenia

```
void pamiec_tablica_do_szkolen(Szkolenie *** tab_szkolenie,
    int *liczba_szkolen);
```

Funkcja przydziela pamieć do tablicy gdzie przechowywane sa adresy szkoleń. Tablica jest używana aby pozwolić na szybkie sortowanie które polega jedynie na zmianie kolejności wskaźników.

```
void dodaj_szkolenie(Szkolenie *** tab_szkolenie,
    int * liczba_szkolen, int * granica_szkolen, Lista*wlista,
    int * liczba_typow);
```

Funkcja wywołuje inna funkcje(wypelnij_szkolenie) która odpowiada za obsługe tekstowa i komunikacje z użytkownikiem. Sama zarzadza wywołaniem zwiekszenia pamieci oraz przydzielenie adresu odpowiedniemu indeksowi w tablicy. Aby było to możliwe potrzebne jest przekazanie ***.

```
void wypelnij_szkolenie(Szkolenie ** szkolenie, Lista *wlista,
    int * liczba_typow)
```

Funkcja przyjmujaca dane tekstowe i komunikujaca sie z użytkownikiem. Wywoływana w środku jest funkcja wczytaj oraz jej bliźniaki zaimplementowana we wczytaj.

```
void sortuj_szkolenia(Szkolenie *** tab_szkolenie,
    int * liczba_szkolen)
```

Funkcja sortujaca, pobiera najpierw pole po jakim ma sortować a nastepnie typ(rosnaco\malejaco)

```
void usun_wszystko(Szkolenie ** tab_szkolen, Lista *wlista,
    int * liczba_szkolen, int * liczba_typow,
    FILE *f, char* nazwa_pliku)
```

Funkcja usuwa wszystkie typy szkolen oraz szkolenia(wywojujac dwie funkcje). Powoduje także usuniecie tekstu z pliku.

```
void usun_pamiec_szkolenia(Szkolenie ** tab_szkolen, int * liczba_szkolen)
```

Funkcja zwalnia pamieć przeznaczona do przechowywania szkolen.

```
void usun_pamiec_tablca_szkolen(Szkolenie ** tab_szkolen)
```

Funkcja zwalnia pamiec przechowywana na przetrzymywanie talicy szkoleń. Jest wywoływana tylko raz przy zamykaniu programu.

```
void edytuj_szkolenie(Szkolenie *** tab_szkolenie, int * liczba_szkolen,
    int * granica_szkolen, Lista*wlista,
    int *liczba_typow)
```

Funkcja pozwala na edytowanie dowolnego pola także typu. Zapisuje dane po sprawdzeniu że sa one poprawnego formatu.

```
void usun_szkolenie(Szkolenie *** tab_szkolenie,
    int * liczba_szkolen)
```

Funkcja usuwa konkretne jedno szkolenie podane przez użytkownika. Nastepnie przesuwa wskaźniki w tablicy tak aby były ciagłe. Nastepnie zmniejsza dana liczba_szkolen.

4.5 plik

zawiera funkcje do odczytywania danych z pliku oraz do zapisywania danych;

```
char* otworz_plik(FILE **f);
```

Funkcja wzraca wskaźnik na nazwe pliku. Jako argument przyjmuje wskaźnik na wskaźnik na plik utoworzyony w main.

```
int zamknij_plik(FILE **f);
```

Fucnkja zamyka plik.

```
void zapisz_do_pliku(Szkolenie ** tab_szkolenie, int * liczba_szkolen,
    int * granica_szkolen, Lista * wlista, FILE * f, int * liczba_typow,
    char* nazwa_pliku)
```

Funkcja zapisuje dane do pliku w nastepujacy sposób. Liczba typów spacja liczba szkoleń enter. Każde pole szkolenia po enterze.

```
void otworz_z_pliku(Szkolenie *** tab_szkolenie, int * liczba_szkolen,
    int * granica_szkolen, Lista * wlista, FILE * f, int * liczba_typow,
    char* nazwa_pliku)
```

Funkcja przydziela pamieć i przepisuje dane z pliku. Dane musza być podane w odpowiednim formacie. Szczegółowe działanie jest analogiczne do funkcji opisanych wyżej.

4.6 wczytaj

zawiera funkcje odczytujace dane;

```
char *wczytaj(char * tab, int ile);
```

Funkcja pobiera maksymalna liczbe znaków do wczytania. W środku uruchamiana jest funkcja fgets i odpowiednio modyfikowana oraz sprawdzana pod katem poprawności danych. Zwraca wskaźnik na łańcuch lub w przypadku niepowodzenia NULL. Zmienia pierwszy znak na duży. Ta funkcja pozwala na wprowadzenie jedynie liter.

```
char *wczytaj2(char * tab, int ile);
```

Analogicznie jak funkcja wyżej, tyle że pozwala na wprowadzenie dowolnych znaków.

```
char *wczytaj3(char * tab, int ile);
```

Analogicznie jak wyżej tylko nie zmienia pierwszego znaku na duży.

```
int wybor_menu(Szkolenie *** tab_szkolenie, int * liczba_szkolen,
    int * granica_szkolen, Lista*wlista, int * liczba_typow,
    FILE *f,char *nazwa_pliku)
```

Funkcja w zależności od wyboru użytkownika uruchamia odpowiednie funkcje do edycji/dodawania/wyświetlania/sortowania bazy danych lub kończy program.

```
void wczytaj_liczbe(double * liczba)
```

Funkcja pozwala na wczytanie dowolnej liczby rzeczywistej. W przypadku błednych danych prosi ponownie o ich wprowadzenie oraz czyści bufor.

```
void wczytaj_liczbe_int(int * liczba2)
```

Funkcja pozwala na wczytanie liczby całkowitej dodatniej.

Pliki posiadaja odpowiedniki .h zawierajace prototypy funkcji.

5 Algorytm ogólny

1. Wczytaj nazwe pliku.
2. Zainicjalizuj liste typów, przydziel pamieć na tablice struktur.
3. Otwórz\utwórz plik.
4. Wczytaj rekordy do struktur.
5. Wczytaj wybór użytkownika(dodaj, usuń, wyświetl,(q) ZAKOŃCZ PROGRAM...)
   Użytkownik wybrał:
   1) Wyświetl baze
   2) Dodaj rekord
   3) Czyść cała baze
   4) Edytuj szkolenie
   5) Usuń szkolenie
   6) Sortuj baze
   q) Zakończ program

6. Wróć do pkt 5
7. Czyść pamieć.
8. Zamknij plik

6 Algorytmy szczegółowe 5 punktu

6.1 Wyświetl baze

1. Jeśli baza jest pusta zakończ ten pod-algorytm
2. Wyświetl liste możliwych sortowań z odpowiadajacymi im numerami
3. Pobierz tryb sortowania
4. Wyświetl baze
5. Zapisz plik w obecnym sortowaniu

6.2 Dodaj rekord

1. Jeśli plik jest pusty poproś o dodanie Typów szkolenia, w innym przypadku przejdź do 2
2. Dodaj szkolenie
3. Zapisz plik

6.3 Czyść cała baze

1. Usuń wszystkie typy - zwolnij pamieć na nie zarezerwowana
2. Usuń wszystkie struktury - zwolnij pamieć na nie zarezerwowana

6.4 Edytuj szkolenie

1. Jeśli baza jest pusta zakończ ten pod-algorytm
2. Wyświetl wszystkie szkolenia
3. Pobierz numer szkolenia które użytkownik chce edytować
4. Wyświetl pola z odpowiadajacymi im numerami
5. Pobierz numer pola który ma być edytowany
6. Edytuj pole
7. Zapisz do pliku

6.5 Usuń szkolenie

1. Jeśli baza jest pusta zakończ ten pod-algorytm
2. Wyświetl wszystkie szkolenia
3. Pobierz numer szkolenia które użytkownik chce usunać
4. Usuń szkolenie

6.6 Sortuj baze

1. Jeśli baza jest pusta zakończ ten pod-algorytm
2. Wyświetl liste możliwych sortowań z odpowiadajacymi im numerami
3. Pobierz tryb sortowania
4. Zapisz plik w obecnym sortowaniu

7 Testowanie

Baza danych została przetestowana dla plików o różnych nazwach(istniejacych lub nie), dla rekordów przekraczajacych podstawy rozmiar danych potrzeby do alokacji. Dla wielokrotnej liczby typów. Wszystkie funkcje zostały użyte.

8 Odstepstwa

Baza danych celowo nie pozwala na tworzenie rekordów bez istniejacych typów szkoleń.