Programozás alapjai 1. (BMEVIEEA100)

Gyakorlat anyaga az 10. oktatási héten

Mindkét csoportban most írják a 3. ZH-t. Az utolsó 3 hét múlva.

Hétfői csoportokban a feladat: main függvény változatai, függvénypointerek és rekurzió (qsort is), azaz a múlt héten küldött anyag. A többieknek a láncolt listákat kell nyúznia: legyen olyan, ahol a main épít egyszerű listát, és olyan, ahol függvényekkel dolgozunk, utóbbit rendezve építse. Javasolt egyirányú és két irányban láncolt listát is mutatni.

3. ZH (használható hozzá kétoldalas C összefoglaló és kézzel írott puska).

Név:		FV sikeres:		× 3 pont=	
Neptun kód:		FV sikertelen:		×–3 pont=	
Hallgató aláírása:		MF sikeres:		× 2 pont=	
		MF sikertelen:		×–2 pont=	
		FV=FeleletVálasztós, MF=Megoldandó Feladat		Össz:	

Pontozás: -2: 1, 3-8: 2, 9-14: 3, 15-18: 4, 19-22: 5

Új anyag

Feladat: Írjon programot, mely ismeretlen számú azonosító-hőmérsékelt adatpárt olvas be a szabványos bemenetről, az adatsor végét 0 hőmérséklet érték jelzi. Írja ki az átlagnál alacsonyabb hőmérsékleteket!

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

typedef struct stlist{
    char azonosito[50];
    double homerseklet;
    struct stlist * next;
}lista,*plista;

int main(){
    lista strazsa;
    plista gyoker=&strazsa,p;
    int mehet=1,db=0;
    double atlag=0;
    strazsa.next=NULL;
    do{
        printf("Kerem az azonositot!\n");
        scanf("%s",strazsa.azonosito);
        printf("Kerem a komersekletet!\n");
        scanf("%lg",&strazsa.homerseklet);
        if(strazsa.homerseklet==0.0)mehet=0;
        else{
            p=(plista)malloc(sizeof(lista));
            if(p==NULL){puts("Memoriafoglalas sikertelen");exit(-1);}
            *p=strazsa;
            p->next=gyoker;
            gyoker=p;
            atlag+=p->homerseklet;
            db++;
        }
    }while(mehet);
    if(db)atlag/=db;
    else return 0;
    printf("Atlag alatti homersekletek:\n");

    for(p=gyoker;p->next!=NULL;p=p->next)
        if(p->homerseklet<atlag)
            printf("%s: %g\n",p->azonosito,p->homerseklet);
    while(gyoker!=&strazsa){
        p=gyoker;
        gyoker=gyoker->next;
        free(p);
    }
}
```

Feladat: Írjon programot, amely hallgatók neveit és ZH-n elért pontszámait olvassa be a szabványos bemenetről. A hallgatók száma ismeretlen. A beolvasás után kérdezze meg a felhasználót, hogy hányan mehetnek elővizsgázni, írja ki pontszám szerint csökkenő sorrendbe az elővizsgán résztvevő hallgatók nevét és pontszámát!

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

typedef struct stlist{
    char nev[50];
    unsigned pontszam;
    struct stlist *next,*prev;
}lista,*plista;

void betesz(plista gy,lista uj){
    plista p=gy->next,q;
    while(p->next!=NULL&&p->pontszam>uj.pontszam)p=p->next;
    q=(plista)malloc(sizeof(lista));
    if(q==NULL){printf("Memfoglalasi hiba\n");exit(-1);}
    *q=uj;
    q->next=p;
    q->prev=p->prev;
    p->prev=q;
    q->prev->next=q;
}

unsigned darab(plista gy){
    plista p=gy->next;
    unsigned n=0;
    while(p->next!=NULL)n++;
    return n;
}

void mehet_ev_re(plista gy,unsigned db){
    plista p=gy->next;
    unsigned n=0;
    for(;p->next!=NULL&&n<db;n++,p=p->next)
        printf("%40s: %u\n",p->nev,p->pontszam);
}

void torol(plista gy){
    plista p=gy->next;
    while(p->next){
        gy->next=p->next;
        free(p);
        p=gy->next;
    }
    p->prev=gy;
}

int main(){
    lista s1,s2;
    s1.prev=s2.next=NULL;
    s1.next=&s2;
    s2.prev=&s1;
    do{
        fflush(stdin);
        printf("Nev: ");
        gets(s1.nev);
        printf("Pontszam: ");
        scanf("%u",&s1.pontszam);
        if(s1.pontszam!=0)betesz(&s1,s1);
    }while(s1.pontszam!=0);
    printf("Ferogely: ");
    scanf("%u",&s1.pontszam);
    mehet_ev_re(&s1,s1.pontszam);
    torol(&s1);
}
```

-

Házi feladat

Most nincsenek tesztkérdések.

Példák

1. Adott az alábbi struktúra zöldségek láncolt listában történő nyilvántartására:

```
struct zd {
  char *nev;
  int ar;
  struct zd* next;
};
```

   a.) Készítsen függvényt (add_zd), mellyel a láncolt lista végéhez új elemet hozzáadhatunk. Ha még üres a lista, a hozzáadni kívánt elem legyen az első. Bemenő paraméterek, lista első elemére mutató kettős pointer (a pointerre mutató pointer), zöldség neve (konstans), ára.
   b.) Készítsen függvényt (del_zd), mellyel a láncolt lista egy adott elemét tudjuk törölni. A bemenő paraméterek között szereplő "nev" mező a törlendő elem nevét tartalmazza. Bemenő paraméterek: lista első elemére mutató kettős pointer (a pointerre mutató pointer), és a név.
   c.) Készítsen függvényt (mod_zd), mellyel a lista egy elemét módosíthatjuk. A reginev bemenő paraméter szolgáljon az azonosításra (ezt a nevű tételt akarjuk módosítani), az ujnev és ujar értékekkel módosítsuk a tételt. További bemenő paraméter a lista első elemére mutató pointer.
   d.) Készítsen függvényt (list_zd), mely a láncolt listát kilistázza a képernyőre. Bemenő paraméterek: első elemre mutató pointer.
2. Adott egy futóverseny résztvevőinek listája:

```
struct versenyzo {
  char *nev;
  int ind,erk;
  struct versenyzo *next;
};
```

   Az ind adat az indulási időpontját, az erk az érkezését tárolja UnixTime formátumban (1970. jan. 1. 0:00 óta eltelt másodpercek száma). Készítsen programot, mely meghatározza, hogy melyik futó kapja az első, második és harmadik díjat! Vigyázzon, mert aki benevezett, de nem indult, vagy nem ért célba, azok is rajta vannak a listán, de érkezési vagy indulási időpontjuk nulla.: