Ghidul Micului Spiriduș: Pregătire pentru examenul de PC

1. STRUCTURI

Ce sunt structurile?

Structurile sunt tipuri de date ce permit stocarea unor date de tipuri diferite.

Cum se definesc structurile?

```
typedef struct nume_structura {
        tip_data  nume_data;
        tip_data  nume_data;
        .....
        tip_data  nume_data;
}nume_variabila1, nume_variabila2;
```

Ce înseamnă fiecare parte și la ce ne ajută?

Scriem *„ typedef struct nume_structura*" ci nu *„struct nume_structura"* pentru ca typedef are rolul de a transforma structura intr-un nou tip de variabila si astfel in loc sa scriem mereu *„struct nume_structura"* cand vrem sa folosim structura, vom scrie doar *„nume_structura"* (e ca si cu celelalte tipuri de variabila, cand scrii int banane, nu mai scrii si definitia int-ului, asa e cu structura, in loc sa scrii „ struct bananier banana" scrii doar „ bananier banana").

Scriem la final *nume_variabila1, nume_variabila2* pentru ca putem declara variabile imediat dupa crearea structurii, e mai recomandat sa le creezi in functii decat global.

Transmitere variabile de tip structura ca argumente pentru functii:

1.Normal

Cum ar arata antetul functiei:

```
tip_functie nume_functie(nume_structura nume_variabila);
```

Cum transmitem?

```
nume_functie(nume_variabila);
```

Cum se apeleaza datele din structura?

nume_variabila.data_structura;

Exemplu:

Avem structura:

```
typedef struct cos_fructe{
        int banane;
        int mere;
        int kiwi;
};
```

Avem o functie ce va citii valori pentru fiecare element din structura:

```
void citire_fructe(cos_fructe fruct){
scanf(" %d", &fruct.banane);
scanf(" %d", &fruct.mere);
scanf(" %d", &fruct.kiwi);
}
```

Acum vom defini in main o variabila de tipul cos_fructe si vom apela functia de citire:

```
int main()
{
        cos_fructe fruct;
        citire_fructe(cos_fructe fruct);
        return 0;
}
```

2.Folosind pointeri:

Cum ar arata antetul functiei:

tip_functie nume_functie(nume_structura *nume_variabila);

Cum transmitem?

nume_functie(&nume_variabila);

Cum se apeleaza datele din structura?

nume_variabila->data_structura;

Rescriem exemplul de mai sus folosind pointeri:

Avem structura:

```
typedef struct cos_fructe{
        int banane;
        int mere;
        int kiwi;
};
```

Avem o functie ce va citii valori pentru fiecare element din structura:

```
void citire_fructe(*cos_fructe fruct){
scanf(" %d", fruct->banane);
scanf(" %d", fruct->mere);
scanf(" %d", fruct->kiwi);
}
```

Acum vom defini in main o variabila de tipul cos_fructe si vom apela functia de citire:

```
int main()
{
        cos_fructe fruct;
        citire_fructe(&fruct);
        return 0;
}
```

2.Alocare dinamica

Cum se aloca dinamic?

```
vector=(tip_vector*)malloc(lungime*sizeof(tip_vector));
```

Cum se realoca dinamic?

```
vector=realloc(lungime_noua*sizeof(tip_vector));
```

Cum se elibereaza memoria?

```
free(vector);
```

Exemplu:

```
int main()
{
int *vector, nr_elemente_initial, nr_elemente_final,contor,contor2;
scanf( „%d”, &nr_elemente_initial);
//alocare initiala a vectorului
vector=(int*)malloc(nr_elemente_initial*sizeof(int));
//citire elemente vector
for(contor=0;contor<nr_elemente_initial;contor++)
scanf(„%d”,&vector[contor]);
//decidem ca vrem mai multa memorie
scanf( „%d”, &nr_elemente_final);
//realocarea memoriei
vector=realloc(nr_elemente_final*sizeof(int));
//continuam citirea elementelor
for(contor2=contor;contor2<nr_elemente_final;contor2++)
scanf(„%d”,&vector[contor2]);
//eliberam memoria
free(vector);
return 0;
}
```

3.Functiile qsort si bsearch

Ce este qsort?

Este o functie ce sorteaza un vector.

Cum arata qsort?

qsort(vector, nr_elemente_vector, sizeof(tip_vector), cmpfunc);

vector este vectorul care trebuie sortat

nr_elemente_vector reprezinta numarul de elemente al vectorului

sizeof(tip_vector) este reprezentat de tipul vectorului, de exemplu sizeof(int) daca ai un vector de tip int

cmpfunc este o functie de comparare, poate sa difere de la problema la problema, dar de obicei e asta:

```
int cmpfunc (const void * a, const void * b) {
  return ( *(int*)a - *(int*)b );
}
```

Exemplu:

```
//creare functie de comparare
int cmpfunc (const void * a, const void * b) {
  return ( *(int*)a - *(int*)b );
}
int main () {
  int vector,nr_elemente_vector,contor;
  //citire vector
  scanf(“%d”, &nr_elemente_vector);
  for( contor = 0 ; contor < nr_elemente_vecotr; contor++ ) {
    scanf("%d ", &vector[contor]);
  }
```

```
 // afisare vector nesortat
   for( contor = 0 ; contor < nr_elemente_vecotr; contor++ ) {
      printf("%d ", vector[contor]);
   }
//sortare cu qsort
   qsort(vector, nr_elemente_vector, sizeof(int), cmpfunc);
  //afisare vector sortat
   for( contor = 0 ; contor < nr_elemente_vecotr; contor++ ) {
      scanf("%d ", &vector[contor]);
   }
   return(0);
}
```

Ce este bsearch?

Aceasta este o functie ce cauta intr-un vector un anumit element si returneaza un pointer catre el.

Cum arata?

item = (tip_vector*) bsearch (&key, vector, nr_elemente_vector, sizeof (tip_vector, cmpfunc);

item este un pointer in care o sa fie stocata adresa de memorie a elementului cautat daca este gasit(se declara de tipul: tip_vector *item, deoarece trebuie sa fie de acelasi tip cu elemental vectorului)

key este defapt doar elementul pe care il cautam(de ex daca vrem sa gasim intr-un vector de tip int elementul 32, key va fi egal cu 32)

vector este vectorul in care o sa cautam elementul

nr_elemente_vector reprezinta numarul de elemente al vectorului

sizeof(tip_vector) este reprezentat de tipul vectorului, de exemplu sizeof(int) daca ai un vector de tip int

cmpfunc este o functie de comparare, poate sa difere de la problema la problema, dar de obicei e asta:

Exemplu:

```
int cmpfunc(const void * a, const void * b) {
   return ( *(int*)a - *(int*)b );
}
int main () {
   int vector,nr_elemente_vector,contor,item,key;
   //citire vector
   scanf(“%d”, &nr_elemente_vector);
   for( contor = 0 ; contor < nr_elemente_vecotr; contor++ ) {
      scanf("%d ", &vector[contor]);
   //citim de la tastatura elementul pe care vrem sa il gasim
  scanf(“%d”,&key);
   //folosim bsearsch sa il gasim
   item = (int*) bsearch (&key, vector, nr_elemente_vector, sizeof (int), cmpfunc);
   if( item != NULL ) {
      printf("Am gasit elementul = %d\n", *item);
   } else {
      printf("Item = %d Nu am gasit elementul\n", *item);
   }
   return(0);
}
```

4.Fisiere

Ce sunt fisierele?

Ai invatat la USO!!

La ce sunt folositoare in programare?

Poti sa dai programului date de intrare mult mai usor, poti sa stochezi anumite date importante din program in fisier pentru a fi folosite mai tarziu.

Ce tipuri de fisiere cunoastem de la curs?

Nu am fost la curs.

Dar stim de **fisiere text** si **fisiere binare**.

Cum declaram un fisier in C?

FILE *fisier;

Cum deschidem un fisier in C?

fisier=(“nume_fisier”,”mod”);

fisier este numele variabilei declarate la intrebarea de mai sus

nume_fisier este numele fisierului de pe disc pe care vrem sa-l deschidem

mod este reprezentat de modul de prelucrare al informatiei, deci putem deschide un fisier in urmatoarele moduri:

r – read

w- write

rb- read binary (pentru fisierele binare)

wb – write binary (pentru fisierele binare)

mai sunt si alte moduri dar nu primesti la examen, pup

Cum citim date dintr-un fisier text?

Folosim functia fscanf care e echivalentul functiei scanf pentru fisiere text.

Are structura:

fscanf(fisier,”%tip_variabila”,&variabila);

Cum scriem date intr-un fisier text?

Folosim functia fprinf care e echivalentul functiei prinf pentru fisiere text.

Are structura

fprintf(fisier,”%tip_variabila”,variabila);

Cum citim date dintr-un fisier binar?

Folosim functia fread.

Are structura:

fread(&variabila, sizeof(tip_variabila),1,fisier);

Cum scriem date intr-un fisier binar?

Folosim functia fwrite.

Are structura:

fwrite(&variabila, sizeof(tip_variabila),1,fisier);