Strukture u C-u

Strukture može objasniti kao složeni tip podataka koji definira oblik željenog zapisa, a u sebi sadrži imena i tipove podataka svojih članova. Pravilnom definicijom strukture omogućeno nam je koristiti strukturu kao tip podataka u deklariranju bilo kakvih objekata, pokazivača na te objekte ili polja takvih struktura.

Definiranje struktura

Strukture možemo definirati na mnogo načina, a ovdje će biti opisano njih nekoliko. Prvo ću proći kroz cijeli proces igranja s „klasnično“ deklariranom strukturom, a poslije toga slijede varijacije na temu u c-duru. ☺

Općenito možemo reći da nam prilikom deklaracije strukture treba sljedeći uzorak:

struct [naziv_strukture] {popis članova strukture}

Sam naziv strukture možemo i izostaviti ili ga napisati iza vitičaste zagrade, ovisno o tome što želimo postići.

Jedan od školskih primjera deklaracije strukture prema gore opisanom uzorku izgledao bi ovako:

```
struct adresa{

        char grad[20];
        char ulica[20];
        int postBroj;
        short kucniBroj;
};
```

Inače, nije bitno pišemo li naziv strukture malim ili velikim slovom, iako neki prakticiraju pisanje velikim slovom kako bi se strukture razlikovale od funkcija, varijabli, polja… Ja ću koristiti mala slova, a nove tipove podataka koje ću definirati pomoću struktura (doći ćemo i do toga) započet ću velikim slovom.

Dakle, što smo napravili u gornjem primjeru? Deklarirali smo novu strukturu koju smo nazvali adresa i dodijelili joj članove grad, ulica, postBroj i kucniBroj. Nadam se da je jasno kojeg su tipa podataka članovi novodeklarirane strukture. ☺

Sada kada smo si stvorili strukturu, možemo kreirati i neke objekte toga tipa da se imamo čime igrati. Recimo da želimo kreirati dva objekta i pokazivač na strukturu.

```
struct adresa mojaAdresa, tvojaAdresa, *njegovaAdresa = &tvojaAdresa;
```

Sad imamo kanticu i grabljicu, ali nemam pijesak, pa je vrijeme da i njega dovedemo u priču. ☺ Dodijelit ćemo članovima neke vrijednosti. Koristiti ću više ekvivalentnih načina kako biste vidjeli da se ne mora koristiti samo „.“, koja je super za dinamičko pridjeljivanje vrijednosti.

```
strcpy(mojaAdresa.grad, „Babina Greda“);
strcpy(tvojaAdresa.grad, „Bibinje“);
```

Primijetite da članu grad ne možemo izravno dodijeliti neku vrijednosti, nego moramo koristiti kopiranje stringa funkcijom strcpy, koju smo naučili koristiti na pipiju. ☺

```
mojaAdresa.postBroj = 10000;
mojaAdresa.kucniBroj = 25;
tvojaAdresa = {.postBroj = 10040, .kucniBroj = 16};
```

Ova dva načina punjenja članova nekim vrijednostima ekvivalentna su. U svojim primjerima vjerojatno ćete koristiti ovaj prvi. Napisao sam i alternativu za one koji žele znati više, ali neću ulaziti u detalje i pobliže objašnjavati taj način. Ako vas zbunjuje ili vam se ne sviđa, zaboravite ga i pravite se da ga nikada niste vidjeli. ;)

Parkić u kojem se igramo nije baš velik i nema puno dodatnog sadržaja, pa ćemo samo ispisivati vrijednosti. ☹ Prvo ćemo koristiti „normalan“ pristup, a onda ćemo sve začiniti pokazivačima. ☺

Recimo da želimo ispisati kućne brojeve koje smo uvalili u svoje varijable.

```
printf(„Moj kucni broj je %d, a tvoj %d.“, mojaAdresa.kucniBroj, tvojaAdresa.kucniBroj);
```

Sjetite se da smo na početku definirali i pokazivač na strukturu (i to na objekt tvojaAdresa), pa onda i njega možemo koristiti za ispisivanje sadržaja članova strukture.

```
printf(„Tvoj kucni broj je %d.“, (*njegovaAdresa).kucniBroj);
```

Zagradu smo stavili da se ne zezenemo zbog različitih prioriteta operatora.

Ovo je način koji bismo koristili da se vratimo u korijen ovog predmeta, ali smo na ASP-u i znamo i drugačiji način. Štoviše, stalno ga koristimo kad se igramo s listama . ;)

printf(„Tvoj kucni broj je %d.“, njegovaAdresa->kucniBroj);

Stvaranje sinonima za novi tip podatka

Sad kad smo se lijepo poigrali strukturama, možemo neke stvari malo i pojednostaviti i poljepšati. Nema smisla da za svako inicijaliziranje novog objekta strukture koristimo oblik *struct naziv_strukture*. Prirodnije bi bilo kreirati čime bismo postigli da definicija izgleda kao kada to radimo sa cijelim brojevima, realnim brojevima, poljima i ostalim ljepotama. U tu svrhu kreirat ćemo sinonim za novi tip podatka koristeći ključnu riječ typedef.

Još uvijek koristimo mudro smišljeni primjer iz parkića ☺ pa ćemo novi tip podataka kreirati na sljedeći način:

typedef struct adresa Adresa.

Gornjom linijom rekli samo da ćemo starom tipu podatka pristupati ključnom riječi Adresa. Ako vam ovo nije baš jasno, zamislite to kao ovo:

typedef unsigned int superUrbaniBroj.

Ovdje definiramo sinonim za tip podatka unsigned int, baš kao što smo u primjeru sa strukturom definirali sinonim za struct adresa.

Varijacije na deklaraciju strukture

Budući da nemam uvijek iste primjere i zahtjeve, ponekad je strukture dobro i malo drugačije deklarirati. Ovdje ću navesti nekoliko takvih primjera.

Recimo da želimo odmah kreirati i nekoliko objekata deklarirane strukture (primjer kao i onaj prvi kojim smo se igrali):

```
struct adresa{

        char grad[20];
        char ulica[20];
        int postBroj;
        short kucniBroj;
}mojaAdresa, tvojaAdresa;
```

Možemo odmah definirati i određen broj varijabli tipa strukture kada znamo da nam više od toga neće biti potrebno.

```
struct {

        char grad[20];
        char ulica[20];
        int postBroj;
        short kucniBroj;
}mojaAdresa;
```

Jedna od mogućnosti je da odmah kreiramo i novi sinonim koji ćemo koristiti kroz svoj program.

```
typedef struct {

        char grad[20];
        char ulica[20];
        int postBroj;
        short kucniBroj;
}adresa;

adresa mojaAdresa
```

Složenije strukture

Kao članove strukture možemo koristiti objekte neke druge strukture. Ovo nam se javlja kod stoga i reda ostvarenima listom kada trebamo pokazivače na nove atome. Na primjer:

```
struct at{

	int element;
	struct at *sljed;
};

typedef struct at atom;

typedef struct{

	atom *vrh;
}Stog;
```

Uočavate li usput sličnost s primjerima koja sam naveo kao varijacije? ☺ Prva struktura je „klasična“ struktura i što se tiče same deklaracije, nema ništa posebno. Druga je, pak, varijacija na temu i odgovara zadnjem primjeru i popisa varijacija. Odmah smo kreirali sinonim na novi tip podatak i njega koristimo kroz svoj program, točnije u glavnom programu kada definiramo stog koji punimo ili praznimo kroz program. ;)

U ovim primjerima vidite da kao članove možemo koristiti i pokazivače na strukture, što nam je zbilja bitno kod igranja s listama jer jedan član mora biti neki element koji stavljamo na stog/red, a drugi je pokazivač na sljedeći član.

Ovime smo došli do kraja. Ako imate pitanja ili kritike jer je nešto loše/netočno/nepotpuno objašnjeno, slobodno se javite. ;)

Za sve one koji žele znati više od toga, preporučam knjigu C in a Nutshell (ja sam je koristio kao literaturu), primjere koje možete pronaći na Mreži svih mreža ili neku drugu literaturu koju smatrate dovoljno kvalitetnom.

Da ne bi bilo zabude; parkić, lopatice i ostale stvari izmislio sam da meni ne bude dosadno ovo pisati, a ne zato što smatram da je tako lakše nešto shvatiti. ☺