Zapiski iz pouka Osnove programiranja II
Programiranje Mikrokontrolerjev

Matej Blagšič

1. marec 2018

Kazalo

1 Osnovno

Pri temu predmetu bomo obravnavali jezik C. Za uporabo lahko preneseš okolje Codeblocks z MinGW inštalacijo ali posebej MinGW compiler in poljubno okolje(Jetbrains).

Pomembno je, da imaš predznanje iz prejšnjega polletja pri Javascriptu, saj so tipi spremenljivk, sintaksa in drugo zelo podobno, tako da v detajle o stvarih, ki so enake ne bom šel.

Vsak dokument začnemo z `#include <stdio.h>` za standardne vhodne in izhodne ukaze. Vsaka koda se izvaja znotraj main funkcije:

```
int main(){
    printf("Hello!\n");
    return 0;
}
```

Prav tako je pomembno uporabiti PODPIČJE za vsakim ukazom/vrstico!!!

Če začnemo na začetku, opazimo `#include` ukaz. Ta se izvrši, preden se karkoli drugega. V temu primeru lahko vnesemo knjižnice. Te nam olajšajo programiranje s tem, da nam en ukaz izvede več ukazov, ki bi jih morali tipkati na roke. To datoteko/knjižnico navedemo lahko z "datoteka"navednicam. Če pa damo v <datoteka>, potem pa išče datoteke v sistemskih mapah okolja. Te datoteke so vrste **header** s končnico **.h**. V našem primeru je knjižnica za pisat in brat podatke - vhodne in izhodne podatke.
To je podobno kot v javascriptu: `<script src="datoteka">`

2 Podatkovni tipi

Si poglejmo zgled:

```
int main(){
    int a;
    float b; //spremenljivka

    printf("Vprisi prvo vrednost");
    scanf("%d", &a);
    printf("Vpisi drugo vrednos");
    scanf("%f", &b);
    printf("%d + %f = %f\n", a, b, a+b);
    return 0;
}
```

C je občutljiv na tip podatkov. Pravimo tudi, da je C statično tipiziran jezik. To pomeni, da moramo vrsto podatka navesti. To pomeni, da se moramo sami odločiti, kakšen tip podatka bo nosila spremenljivka.
Vemo, da v Javascriptu nismo rabili napisati tipa spremenljivke, le `var`, torej je Javascript dinamično tipiziran jezik.

Tipi spremenljivk:

TIP	DOLŽINA(bitov)	FORMATNO DOLOČILO
char	8	%d %c
short, int	vsaj 16	%d
long	vsaj 32	%ld
float	običajno 32	%f
double	običajno 64	%lf
void	0	

V C-ju Boolov tip ne obstaja, tako da primerjalni operatorji delujejo enako, le da vračajo 0 za false in 1 za vse, kar je različno od nič. **Ne obstaja TRUE ali FALSE**.

Spoznali bomo tudi, da je pri celoštevilskem tipu pomembna omejitev območja, pri realnem tipu pa natačnost!

2.1 Celoštevilski tip (n bitov)

Obstaja nepredznačen, ki je od 0 do 2^{32}

2.2 Realni tip (IEEE floating point)

p	eksp. (e)	mantisa (m)
1 bit	8 bitov	23 bitov

Ta ima enojno natančnost (single precision) ali `float` in so števila zapisana z 32 bitno velikostjo. Tako so v desetiškem sistemu števila natančna do 7,22 signifikantnih mest, sepravi 7 mest je natančnih, od 8. števila naprej pa je že vprašljivo. Torej, signifikantno pomeni pomembno, tisto, kar je natančno.

Če hočemo večjo natančnost, uporabimo `double` oz. dvojna natačnost (double precision). Ima kapaciteto 64 bitov, torej v desetiškem do 15,95 mest natančno. Po 15. mestu je že vprašljivo natančno.

3 Branje podatkov

Da nam program prebere podatek, uporabimo funkcijo:

```
scanf("formatno_dolocilo", &spremenljivka);
```

Vidimo, da moramo najprej deklarirati tip podatka, ki ga pričakuje operator Scanf. Potem pa določimo naslovni operator & in nato za njim spremenljivko, ki naj sprejme podatek.

4 Pisanje podatkov

Za pisanje podatkov uporabimo funkcijo:

```
printf("formatni_niz", izrazi)
```

Pomembne so tudi ubežne sekvence. To so `\r \n \t`, ki povejo, kaj se zgodi, ko se text izpiše. `\n` naredi novo vrstico(new line) po besedilu, `\t` je tabulator...

Tako v našem primeru, se a izpiše tam, kjer je njegov %d in b, kjer je %f ter vsota a+b tam, kjer je %f(glej izsek programske kode).

5 Operatorji

Pri C-ju so enaki operatorji, kot v JS, le da z nekimi izjemami: Operator === in !=== ne obstajata.
Prav tako operator za deljenje ne zapišemo kot "/"ne deluje enako. Problem prihaja iz tipa spremenljivk. Če obsoječo spremenljivko x, ki je tipa int, deljimo ali spreminjamo tako, da bi postala ta spremenljivka kateregakoli drugega tipa, kot prvotni int, potem vrne program 0. Primer:

```
int maint(){
    x = 7;
    x = x / 8 * 8

    printf("%d", x);
    return 0;
}
```

Če pa spremenimo prvo 8 z 8.0, potem bo program jo vzel za realno število in deljil in nato nazaj množil z 8, tako se te pokrajšata in program vrne 7.

6 Funkcije

Funkcije deklariramo:

```
float imeFunkcije(){/*telo funkcije*/return 0;}
```

Opazimo, da funkcijo deklariramo kot float oz. funkcijo, ki vrne realno število. V resnici lahko funkcije definiramo kot karkoli hočemo, glede na to, kaj naj bi vrnila.

Prav tako vidimo, da glavna zanka, v kateri se koda izvaja, je `main`. v tej kodi se izvajajo vsi programi in funkcije. Tako se koda, ki je napisana tu notri, se prevede in spremeni v izvršilno kodo(executable).

Whatafak.