Naloga 1 – Ponovitev C

Naloga je namenjena ponovitvi znanja programskega jezika C. Pri reševanju nalog ni potrebnega zunanjega vnosa, spremenljivke lahko definirate kar v kodi (*npr:* int32_t a = 43;). Pri definiciji spremenljivke uporabite bolj *informativne tipe* iz knjižnice *stdint.h* (link) (namesto int uporabite int32_t, namesto short uporabite int16_t, itd...). Zaželena (*ni pa nujna*) je uporaba razvojnega okolja STM32CubeIDE, ki ga bomo uporabljali tekom semestra.

Primer deljenja kode v različne datoteke:

sestej.h:

```c
#include <stdint.h>

int32_t sestej_dve_stevili(int32_t a, int32_t b);
```

sestej.c:

```c
#include "sestej.h"

int32_t sestej_dve_stevili(int32_t a, int32_t b){
    int32_t vsota = a + b;
    return vsota;
}
```

main.c:

```c
#include <stdio.h>
#include "sestej.h"

int32_t main(int32_t argv, char * args[]){
    int32_t prvo = 5;
    int32_t drugo = -16;
    int32_t rezultat = sestej_dve_stevili(prvo, drugo);
    printf("Rezultat: %d + %d = %d\n", prvo, drugo, rezultat);
}
```

Naloge

1. sklop: Osnove

a. Delitelji

Napiši funkcijo *ImaNatankoTriDelitelje*, ki za celo število (vhodni parameter) preveri in vrne 1 (oz. *true*), če ima natanko tri delitelje, drugače vrne 0 (oz. *false*). V glavnem programu prikažite delovanje implementirane funkcije, kjer izpišete vsa cela števila s tremi delitelji v intervalu med 1 in 500.

Nekaj primerov takšnih števil:

Primeri števil s tremi delitelji
25 *Razlaga: Delitelji so 1, 5 in 25*
841 *Razlaga: Delitelji so 1, 29 in 841*
1369 *Razlaga: Delitelji so 1, 37 in 1369*

b. Pretvorba ure

Napiši funkcijo, ki bo število (predstavlja čas v desetinkah sekunde (vrednost 1 je torej 100ms, vrednost 10 pa 1 sekunda)) pretvorilo v ure, minute, sekunde in desetinke sekunde. Pretvorjene vrednosti izpišite v formatu: xxh yym zzs qqqms. *xx* v tem primeru pomeni, da bosta v izpisu ur vedno (vsaj) dve cifri (enako velja potem tudi za minute in sekunde, pri milisekundah pa bodo vedno tri cifre). Za ustvarjanje niza si pomagajte s funkcijo *sprintf* (povezava), v pomoč pa vam bo še ta povezava.

Nekaj primerov:

Število [čas v desetinkah sekunde] (vhod)	Izpis časa v ura, minutah, sekundah in milisekundah (izhod)
1	00h 00m 00s 100ms
15	00h 00m 01s 500ms
50000	01h 23m 20s 000ms

c. Iskanje podniza

Napiši funkcijo IskanjePodniza(char* niz, int dol), ki kot vhod prejme polje znakov in število znakov v polju. Funkcija naj v nizu poišče vse podnize, ki se začnejo z 'A' in se zaključijo z 'Z'. Če najde 'A' drugič, preden najde 'Z', se ga vzame kot del podniza. Funkcija izpiše vse najdene podnize, vsakega v svoji vrsti. Funkcijo definirajte v *.h* datoteki, implementirajte pa jo v *.c* datoteki.

Nekaj primerov:

Primeri klicev funkcije (vhod)	Najdeni nizi (izhod)
IskanjePodniza("BLABLAZBLAZZA" , 13)	ABLAZ AZ
IskanjePodniza("AZABAZAAZZ" , 10)	AZ ABAZ AAZ
IskanjePodniza("BLZ" , 3)	/

2. sklop: Bitne operacije I

a. Izpis števila kot binarno

Napišite funkcijo, ki bo sprejelo 32-bitno število in ga izpisala kot binarno število.

Nekaj primerov:

Število (vhod)	Binaren izpis števila (izhod)
0	0000 0000 0000 0000 0000 0000 0000 0000
1	0000 0000 0000 0000 0000 0000 0000 0001
31	0000 0000 0000 0000 0000 0000 0001 1111
-2	1111 1111 1111 1111 1111 1111 1111 1110
534	0000 0000 0000 0000 0000 0010 0001 0110

b. Sodo ali liho število

Napiši funkciji, ki bosta z uporabo bitnih operacij (*& (and), | (or), ~ (not), ^ (xor), <<(left shift) ali >> (right shift)*) ugotovili, ali je:

a) sodo ali liho,

b) deljivo s štiri.

V nalogi **ne smete** uporabiti operatorja % (modul).

Pomoč: *zapišite si prvih 10 naravnih števil binarno in poskusite ugotoviti vzorec.*

c. Sprememba predznaka

Napišite funkciji, ki bosta z uporabo dvojiškega komplementa podanem številu spremenili predznak. Prva metoda naj kot vhodni parameter **sprejme 32-bitno celo število** ter vrne enako število s spremenjenim predznakom. Druga metoda kot vhodni parameter sprejme **naslov 32-bitnega števila** in spremembo predznaka opravi na tej spremenljivki. Druga metoda ne vrača nič. Funkciji implementirajte v zunanji datoteki (npr. *SpremeniPredznak*(.c in .h)), ki jo dodate v glavni program, v katerem demonstrirate uporabo obeh funkcij.

d. Zlog po zlog z uporabo logičnih operacij

Napiši funkcijo, ki bo podano 32-bitno število razdelili na 4 zloge (uint8_t) in vrnila polje 4 elementov tipa uint8_t (zlogi združeni v polje). Za razdelitev uporabite bitne operacije (<<, >>, &).

e. Zlog po zlog z uporabo kazalcev

Napiši funkcijo, ki bo podano 32-bitno število razdelili na 4 zloge (uint8_t) in vrnila polje 4 elementov tipa uint8_t (zlogi združeni v polje).. Za razdelitev uporabite kazalce.

f. Postavi/počisti bit funkcija

Napiši funkciji, ki bosta določen bit (drugi parameter funkcije; predstavlja zaporedno vrednost bita, ki ga želimo spremeniti – skrajno desni ima vrednost 0) počistil oziroma postavil. Spremembo morate opraviti nad 16-bitnim število (prvi parameter funkcije). Funkcijo ustvarite v drugi datoteki (npr. BitneOperacije(.c in .h)), ki jo dodate v glavno datoteko s stavkom include.

Število (vhod)	Številka bita (vhod)	Postavi bit (izhod)	Počisti bit (izhod)
7 0000 0000 0000 0111	0	7 *Prej:* 0000 0000 0000 0111 *Po:* 0000 0000 0000 0111	6 *Prej:* 0000 0000 0000 0111 *Po:* 0000 0000 0000 0110
-7 1111 1111 1111 1001	1	-5 *Prej:* 1111 1111 1111 1001 *Po:* 1111 1111 1111 1011	-7 *Prej:* 1111 1111 1111 1001 *Po:* 1111 1111 1111 1001
255 0000 0000 1111 1111	4	255 *Prej:* 0000 0000 1111 1111 *Po:* 0000 0000 1111 1111	239 *Prej:* 0000 0000 1111 1111 *Po:* 0000 0000 1110 1111
-1 1111 1111 1111 1111	14	-1 *Prej:* 1111 1111 1111 1111 *Po:* 1111 1111 1111 1111	-16385 *Prej:* 1111 1111 1111 1111 *Po:* 1011 1111 1111 1111

g. Postavi/počisti bit define

Enako navodilo kot pri prejšnji nalogi, samo da namesto funkcijo napišite makro (z uporabo define stavka).

3. sklop: Bitne operacije II

a. Števec bitov

Napišite funkcijo, ki bo za podano 16-bitno število preštela, koliko bitov je postavljenih – imajo vrednost 1. Funkcijo ustvarite v ločeni datoteki (npr. StevecBitov(.c in .h)), ki jo dodate v glavno datoteko s stavkom *include*.

Nekaj primerov:

Število(vhod)	Število postavljenih bitov števila (izhod)
7	3
-7	14
255	8
-1	16
0	0

b. Postavljanje dveh bitov

Napiši funkcijo, ki bo v 16-bitnem številu (prvi parameter metode) spremenil dva zaporedna bita (drugi parameter funkcije je indeks desnega – skrajno desni bit je 0) z vrednostjo, ki jo podamo (tretji parameter funkcije – ima lahko vrednosti 0 [00], 1 [01], 2[10] ali 3[11] (2 bita)). Funkcija vrne novo število.

Število (vhod)	Številka bita (vhod)	Vrednost (vhod)	Postavi bit (izhod)
7 0000 0000 0000 0111	0	2 Binarno: 10	*Prej:* 0000 0000 0000 0111 *Po:* 0000 0000 0000 0110
-7 1111 1111 1111 1001	1	3 Binarno: 11	*Prej:* 1111 1111 1111 1001 *Po:* 1111 1111 1111 1111
255 0000 0000 1111 1111	4	1 Binarno: 01	*Prej:* 0000 0000 1111 1111 *Po:* 0000 0000 1101 1111
-1 1111 1111 1111 1111	14	0 Binarno: 00	*Prej:* 1111 1111 1111 1111 *Po:* 0011 1111 1111 1111

Za razmisliti: Nalogo se da rešiti brez if/switch stavkov.

c. Iskanje vzorca

Napiši funkcijo, ki bo preštela, koliko krat se v 32 bitnem številu pojavi nek 3-bitni vzorec (npr. 101).

Število (vhod)	Vzorec	Število postavljenih bitov števila (izhod)
-1329578751 Bin: 1011 0000 1100 0000 0011 1101 0000 0001	5 Bin: 101	2 Razlaga: 1011 0000 1100 0000 0011 1101 0000 0001
1.431.655.765 Bin: 101 0101 0101 0101 0101 0101 0101 0101	5 Bin: 101	15 Razlaga: 0101 0101 0101 0101 0101 0101 0101 0101

d. Binarni palindrom

Napiši funkcijo, ki bo na vhodu prejela poljubno 16-bitno število in vrnila *true* (oz 1), če je število binarno gledano palindrom (link na wiki), drugače vrne *false (oz. 0)*.

Število (vhod)	Izhod
-21931 Bin: 1010 1010 0101 0101	1 Razlaga: 1010 1010 0101 0101
7 Bin: 0000 0000 0000 0111	0 Razlaga: 0000 0000 0000 0111
960 Bin: 0000 0011 1100 0000	1 Razlaga: 0000 0011 1100 0000

4. sklop: Dodatne naloge

a. Direkten dostop do pomnilnika

Pri vajah (no, vsaj na začetku) bomo direktno dostopali do lokacij na pomnilniku, kjer bomo imeli znano lokacijo, na kateri je zapisana vrednost, ki jo želimo spremeniti (recimo, da želimo spremeniti en bit v tej vrednosti). Poskusite napisati kodo, kako bi to naredili (tega verjetno ne boste mogli testirati, saj vam operacijski sistem na računalniku tega ne bo dovolil).

Primer: na lokaciji *0x40021400* imamo shranjeno 32-bitno število in ga moramo spremeniti tako, da nastavimo poljuben bit (0-31) na 1.

Namig 1: uporaba kazalca (tip kazalca je določen z velikostjo podatka, do katerega želimo dostopati)

Namig 2: uporabite funkcije zapisane zgoraj

b. Enosmerno povezan seznam opravil

Napišite strukturo Opravilo, ki vsebuje naslednje podatke:

- ime (niz znakov največje dolžine 20),
- prioriteta (celo število – 1 zlog),
- čas izvajanja (celo število – 1 zlog).

V glavnem programu ustvarite 3 opravila, ki jih uredite po prioriteti (od najmanjše do največje) in jih povežite v enosmerno povezan seznam z uporabo kazalcev.