****

Tema 5

****

TEME

- Funkcije u C
  - Sintaksa
  - Uporedjivanje sa Pythonom
- Zapis broja u binarnom zapisu
  - Stepeni dvojke
  - Prebacivanje broja iz binarnog u dekadni
    - Primer: 00100101 -> 32+4+1 = 37
  - Prebacivanje broja iz dekadnog u binarni
    - Primer: 85 -> 64+16+4+1 => 01010101
- Operatori nad bitovima
  - Bitsko i & (1&1 daje 1, 1&0 daje 0, 0&0 daje 0)
  - Bitsko ili | (1|1 daje 1, 1|0 daje 1, 0|0 daje 0)
  - Bitsko eksplicitno ili ^ (1^1 daje 0, 1^0 daje 1, 0^0 daje 1)
  - Bitska negacija ~ (~1 daje 0, ~0 daje 1)
  - Šiftovanje bitova
    - Na levo << (pomera bitove na levo za dati broj mesta, popunjava nulama)
    - Na desno >> (pomera bitove na desno za dati broj mesta, popunjava nulama)
      - a=01000101 a=a>>3 daje a=00001000
- Tipovi podataka i bitovi
  - Podaci
    - Integer 4 bajta
    - Float 4 bajta
    - Double 8 bajta
    - Char 1 bajt
  - Funkcija u C sizeof koja moze da vrati broj bajtova za dati podatak
- Negativni brojevi - zapis u bitovima (vodeci bit 0 ili 1 u zavisnosti da li je pozitivan ili negativan broj)
- Decimalni brojevi - zapis u bitovima (zasebno ide integer deo, zasebno decimalni deo, odredjen broj bitova i za jedan i za drugi podatak)

PRIMER 1: Funkcija za ispis bitova nekog broja

KORACI RESAVANJA:

- Prvi put se vidi kako se pise funkcija
- Void povratna vrednost
- Razlike u odnosu na python pri definisanju funkcija
- Sizeof funkcija objasnjenje

- Algoritam koji cemo koristiti:
    - Konacna petlja ponavljanja (jer znamo tacno koliko imamo bitova)
    - U svakoj iteraciji ispisujemo jedan bit koji se provera sa bitskim i (&) da li je nula ili 1
    - For petlja mora da ide opadajuce, da bi se bitovi ispisivali od najtezeg do najlakseg
- Poziv funkcije je u main() funkciji
- Probati za razne brojeve

FINALNA IMPLEMENTACIJA:

```
#include <stdio.h>

void printB(int num){
    int size = sizeof(int);
    int i;
    for(i = size*8-1; i>= 0; i--){
        printf("%d",(num >>i) & 1 );
    }
    printf("\n");
}

int main() {

    printB(5);  // isprobati razna brojeve

    return 0;

}
```

OBJASNJENJE RESENJA

- Objasniti kakve bi promene bile da funkcija treba da ispise bitove od float broja
- Objasniti delove zapisa funkcije i njen poziv

PRIMER 2: Primer bitskih operatora

KORACI RESAVANJA:

- Radi kraceg zapisa, pisacemo bitove samo u znacajnom bajtu
- Potrebno je raditi korak po korak operatore, jedan po jedan, i svaku verziju testirati:
    - Potrebno je ispisivati u komentarima vrednosti bitova za resenje za svaki operator rucno, pa onda prebaciti u dekadni sistem, pa onda pokretanjem programa proveriti da je to taj broj
    - Potrebno je pozivati funkciju za ispis bitova iz prethodnog primera

FINALNA IMPLEMENTACIJA:

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

```
void printB(int num){
    int size = sizeof(int);
    int i;
    for(i = size*8-1; i>= 0; i--){
        printf("%d",(num >>i) & 1 );
    }
    printf("\n");
}

int main()
{
    // a = 5(00000101), b = 9(00001001)
    int a = 15, b = 9;

    // rezultat je 00000001
    printf("a = %d, b = %d\n", a, b);
    printf("a&b = %d\n", a & b);
    printB(a&b);

    // rezultat je 00001101
    printf("a|b = %d\n", a | b);
    printB(a | b);

    // rezultat je 00001100
    printf("a^b = %d\n", a ^ b);
    printB(a ^ b);

    // rezultat je 11111010
    printf("~a = %d\n", a = ~a);
    printB(a);

    // rezultat je 00010010
    printf("b<<1 = %d\n", b << 1);
    printB(b);

    // rezultat je 00000100
    printf("b>>1 = %d\n", b >> 1);
    printB(b);

    return 0;

}
```

OBJASNJENJE RESENJA

- Raditi zadatak korak po korak sa stalnim testiranjem
- Brojevi mogu da budu drugi, bitno je da se isprobaju sve opcije za operatore

PRIMER 3: operacija mnozenja i deljenja kroz operator siftovanja

KORACI RESAVANJA:

- Objasniti da se siftovanjem zapravo desavaju ove dve operacije

FINALNA IMPLEMENTACIJA:

```
#include <stdio.h>

int main()
{
    int x = 19;
    printf("x << 1 = %d\n", x << 1);
    printf("x >> 1 = %d\n", x >> 1);
    return 0;
}
```

OBJASNJENJE RESENJA

- Probati jos neke brojeve
- Sta ako se siftuje za vise od 1?

PRIMER 4: Detekcija vrednosti bita u nekom broju na odredjenoj poziciji

KORACI RESAVANJA:

- Unos broja i pozicije
- Kreiranje shiftovanog broja za onoliko mesta koliko je trazena pozicija (pozicije se racunaju od 1)
- Koristenje operatora & za detekciju bita

FINALNA IMPLEMENTACIJA:

```
#include <stdio.h>

int main()
{
    int a;
    printf("Unesi broj\n");
    scanf("%d", &a);

    int k;
    printf("Unesi poziciju\n");
    scanf("%d", &k);

    a = a >> (k-1);
```

```
    printf("%d", a&1);

    return 0;

}
```

OBJASNJENJE RESENJA

- Istestirati program
- Za unos 13 i poziciju 1 bit je 1, za unos 4 i poziciju 1 bit je 0