2. CIKLUS – 1. µMASS INSTRUKCIJE

(for petlja, beskonačna petlja, break, switch, jednodimenzionalna polja, polja znakova)

7. studeni 2009.

1. ZADATAK (FOR, POLJE)

Autor: Kristijan Franković

Unesite cijeli peteroznamenkasti broj (nije potrebno provjeravati jeli učitani broj peteroznamenkast). Potom broj na ekran ispisati broj obrnutim redoslijedom, gdje je svaka znamenka uvećana za 1!

Primjer:

Rješenje:

```c
#include <stdio.h>
int main ()
{
	int polje[5], broj, i;
	scanf ("%d",&broj);
	for (i=0;i<5;i++)
	{
		polje[i]=broj%10;
		broj/=10;
		polje[i]++;
	}
	for (i=0;i<5;i++)
		printf ("%d", polje[i]);
	printf ("\n");
	return 0;
}
```

Alternativno rješenje:

Zadnja for petlja se može napisati tako da umjesto ispisa obavlja zbrajanje:

```c
for (i=0;i<5;i++)
{
	suma+=polje[i]*koef;
	koef/=10;
}
printf ("%d\n", suma);
```

2. ZADATAK (BREAK, BESKONAČNA PETLJA)

Autor: Kristijan Franković

Učitavati i **pozitivne cijele brojeve** sve dok njihova suma ne premaši dopušteni opseg za **unsigned short int**. Može se očekivati da će uneseni brojevi biti u intervalu [0,65535]. Na ekran potom ispisati sumu bez zadnjeg učitanog broja (koji je premašio opseg) i vrijednost za koju je suma prekoračila dopušteni interval. Obavezno je koristiti **beskonačnu petlju sa ispitivanjem** uvjeta na početku te naredbu **break**.

Uputa:

Najveći dopušteni interval za unsigned short int iznosi $2^{16} - 1 = 65535 \rightarrow [0,65535]$

Primjer:

Broj 550 je premašio sumu. Vrijednost sume bez broja koji je premašio sumu je 65000, a sveukupno sa zadnjim broje suma je premašena za 15 (65550 - 6535 = 15).

Rješenje:

```
#define MAX 65535
#include <stdio.h>
int main ()
{
	int suma=0, broj;
	while (1)
	{
		scanf ("%d",&broj);
		if (suma+broj>MAX)
			break;
		else
			suma+=broj;
	}
	printf ("Suma: %d Preljev: %d\n",suma,suma+broj-MAX);
	return 0;
}
```

Alternativno rješenje:

```
#define MAX 65535
#include <stdio.h>
int main ()
{
	int suma=0, broj, radi=1;
	while (radi)
	{
		scanf ("%d",&broj);
		if (suma+broj> MAX)
			radi=0;
		else
			suma+=broj;
	}
	printf ("Suma: %d Preljev: %d\n",suma,suma+broj-MAX);
	return 0;
}
```

3. ZADATAK (POLJA, FOR) 

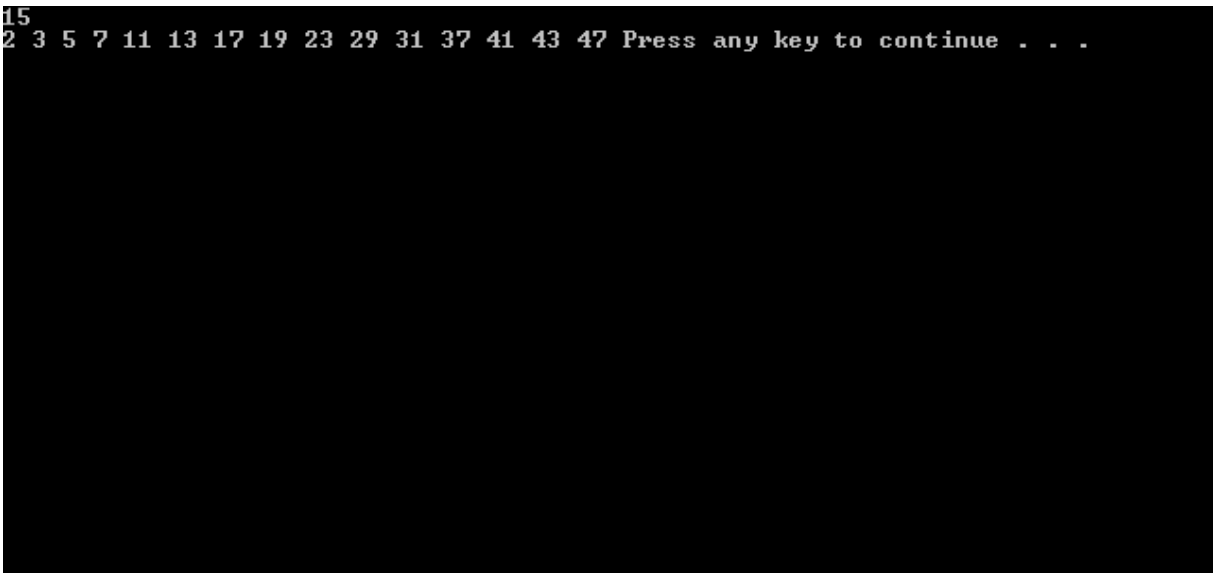
Autor: Alen Rakipović

Napišite program koji će sa tipkovnice računala učitati N, a potom u polje pohraniti prvih N prostih brojeva. Može se pretpostaviti da n neće biti veći od 100.

Uputa:

Za traženje prostih brojeva koristiti algoritam Eratostenovog sita.

Primjer:



Više o Eratostenovom situ i prostim brojevima:

http://hr.wikipedia.org/wiki/Eratostenovo_sito

http://en.wikipedia.org/wiki/Prime_numbers

Napomena:

Dana su dva rješenja – sa Eratostenovim sitom i drugo koje pretpostavi za svaki broj da je prost i onda pokušava oboriti tu tezu.

Rješenje:

```
#include <stdio.h>
#include <stdlib.h>
#define MAXP 100
#define MAXN 550
int main()
{
   int k, i, n, j = 0;
   int prosti[MAXP], polje[MAXN] = {0};
   scanf("%d", &n);

    for( i = 0; i < MAXN; ++i){
        polje[i] = i+2;
   }
   for(i = 0; i < MAXN; ++i) {
      if(polje[i] != 0) {
         for(k = i+1; k < MAXN; ++k) {
            if(polje[k] != 0) {
               if(polje[k] % polje[i] == 0)
                  polje[k] = 0;
            }
         }
      }
   }
   i = 0;
   while(j < n){
      if(polje[i] != 0){
            prosti[j] = polje[i];
            ++j;
      }
      ++i;
   }
   for(i = 0; i < j; ++i)
       printf("%d ", prosti[i]);
   return 0;
}
```

Alternativno rješenje:

Razmotrimo što su to uopće prosti brojevi. Prosti brojevi su oni brojevi veći od 1 djeljivi samo sa 1 ili samim sobom. Tu činjenicu možemo iskoristiti na način da uzmemo potencijalne djelitelje broja i probamo podijeliti broj sa svim tim brojevima. Ukoliko nađemo da je broj negdje djeljiv bez ostataka onda se očito NE radi o prostom broju:
Primjer: uzmemo broj **19**... da bi bio prost ne smije se podijeliti ni sa jednim brojem iz intervala 2-18 (jer sa 1 je djeljiv i sa 19 je djeljiv, ali to nisu uvjeti da bi broj bio prost)...
Redom pokušamo podijeliti 19 sa 2, sa 3, sa 4... pa sve do 18... vidjet ćemo da neće biti djeljiv. Već ovdje vidimo jedno poboljšanje... ne moramo nužno ići do kraja, već samo do polovice broja kandidata Recimo broj 34 sigurno nije djeljiv sa 25, ali je zato zadnji broj koji ga dijeli 17 (njegova polovica).
Pokušajmo sad podijeliti neki broj koji nije prost – 8. Krećemo sa dijeljenjem iz intervala [2-4] (jer je 8/2 = 4) i već vidimo da će ga 2-jka uspješno podijeliti. Već tu možemo zaključiti da broj nije prost i biramo sljedećeg kandidata.

U konačnici, kako realizirati takav algoritam:

Vanjska petlja while sa uvjetom (n>0) nam isključivo služi za kontrolu koliko točno prostih brojeva trebamo. Smanjuje se svaki puta kada pronađemo prosti broj. Potom ulazimo u petlju i za svaki potencijalni kandidat koji je MOŽDA prost (krećemo od 2) **pretpostavimo da je prost** (prost_je stavimo 1, odmah na početku!). Potom ga u FOR petlji pokušamo podijeliti sa brojevima između 2 i polovice njegove vrijednosti... ovdje vidimo da piše djelitelj<kandidat/2+1 što je vrlo bitno, jer želimo proći sredinu da nemamo dvojbe... Ukoliko ovdje ne stoji ovo +1 ili ne piše djelitelj<=kandidat/2 (svejedno je pisalo <= ili +1) u proste brojeve će se upisati i 4 jer nam je prvi kandidat 2, a broj 4/2=2 i a 2<2 će za uvjet dati False (krivo) i neće uči nikada u FOR petlju i ostat će pretpostavka da je broj 4 prost – što je naravno krivo.

Međutim vratimo se na početak, dakle brojeve djelimo sa djeliteljima koji idu od 2 do polovice samog broja... Ukoliko nađemo da je naš kandidat % djelitelj == 0, odnosno ostatak dijeljenja je jednak 0, očito se radi o broju koji nije prost. U to postavimo varijablu prost_je na 0. (srušili smo pretpostavku da je broj prost!) i prekidamo petlju sa break (kakvog smisla ima ispitivati za daljnje djelitelje kad već sad znamo da naš broj kandidat nije prost, a recimo broj nam je 100000000 koji nije prost jer je odmah na početku djeljiv sa 2...)

Na kraju pitamo... ukoliko je u prost_je ostalo 1, tada je broj prost i spremamo ga u polje, pomičemo kontrolnu varijablu polja, smanjujemo n (brojač preostalih prostih koje moramo pronaći). Na kraju neovisno o tome jeli prost ili nije, povećavamo kandidata za +1, čime biramo sljedeći broj.

Na samom kraju izvan petlje ispisujemo sve brojeve, gdje je t granica (napomena, n uništimo u while-u pa ga moramo spremiti u t!)

Napomena – ovo radi samo za prvih 100 jer je polje veličine 100! Ukoliko stavimo veće polje radit će za više prostih brojeva.

```c
#include <stdio.h>
int main ()
{
	int n; //ovdje unosimo koliko prostih želimo
	int prosti[100]; //ovdje spremamo proste brojeve, indeksi 0-99
	int kandidat = 2; //prvi broj kandidat kojeg moramo ispitati
	int djelitelj; //ovdje stavljamo potencijalne djelitelje broja
	int i=0; // sa i se krećemo po polju u koje stavljamo proste brojeve
	int prost_je, t;
	scanf ("%d",&n); //unosimo zeljeni broj prostih brojeva
	t=n;
	while (n>0) //vanjska petlja koja određuje koliko još prostih trebamo
	{
		prost_je = 1; //pretpostavimo da je prost (za svaki kandidat)!
		for(djelitelj=2;djelitelj<kandidat/2+1;djelitelj++) //prol. kroz sve pot. djel.
			if (kandidat%djelitelj == 0) //ukoliko nađemo djelitelj
			{
				prost_je = 0; //zaključujemo da nije prost
				break; //prekidamo petlju
			}
		if (prost_je == 1) //ukoliko je broj ipak bio prost
		{
			prosti[i]=kandidat; //stavi broj u polje
			i++; //povećaj brojač polja
			n--; //smanji broj ukupnih preostalih brojeva koje moramo pronaći
		}
		kandidat++; //slijedeći broj je sada kandidat

	}
	for (i=0;i<t;i++) //ispisujemo sve proste brojeve (t je zato jer smo n uništili!)
	{
		printf ("%d ", prosti[i]);
	}
	return 0;
}
```

Najčešće greške kod ovog algoritma:

Da se ne postavi prost_je na 1 svaki puta prije nego se uopće pokuša ispitati. Dakle prije nego ispitujemo na početku petlje obavezno postaviti pretpostavku.

Pogrešno odredimo onaj interval (<= ili +1 obavezno!)

Da n koristimo kao iterator polja! (tada se brojevi naopačke i krivo upišu, jer krećemo od 100, pa 99, pa 98... i smanjujemo do 0... osim što ćemo upisati 2 u polje[100] koje nemamo u najgorem slučaju, brojevi su naopako upisani).

Da pokušamo prilikom ispisa kao graničnik postaviti n... a njega smo već uništili smanjivanjem.

4. ZADATAK (POLJA, FOR)

Autor: Alen Rakipović

Napišite program koji učitava prvo učitava N i potom učitava N prirodnih brojeva u polje. Može se pretpostaviti da N neće biti veći od 100. Nakon toga na ekran ispisati aritmetičku sredinu brojeva čiji je indeks polja prost broj. Za traženje prostog broja iskoristiti algoritam iz prethodnog zadatka.

Primjer:

```
5
3
6
9
4
6
Aritmeticka sredina je: 6.500000Press any key to continue . . .
```

Rješenje:

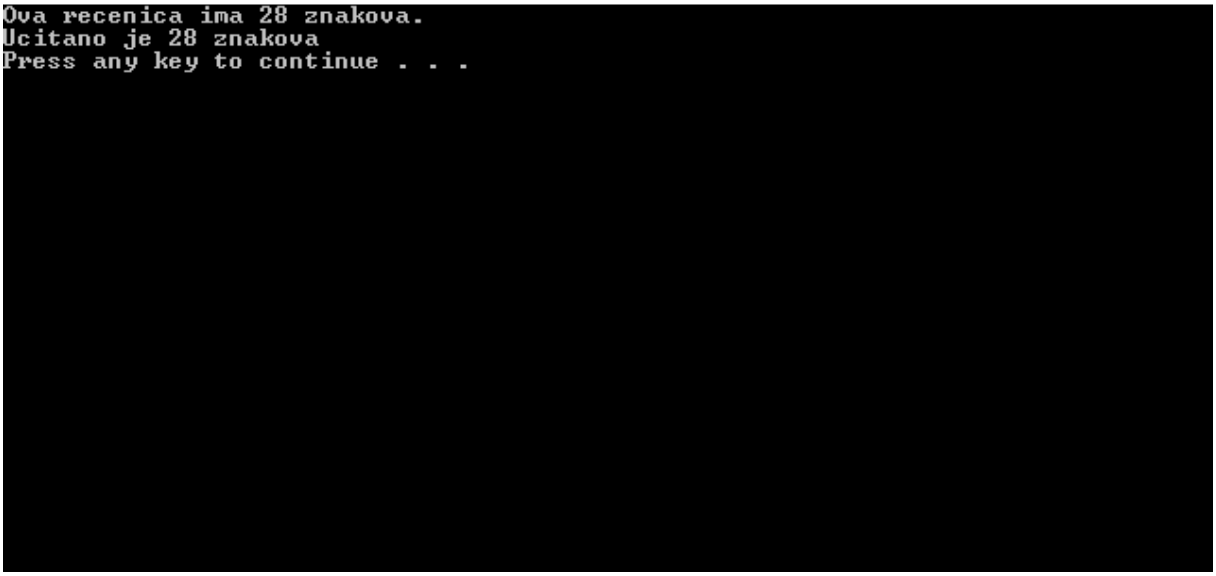
```
#include <stdio.h>
#include <stdlib.h>
#define MAXP 100
#define MAXN 550
int main()
{
   int k, i, n, j = 0;
   int brojac = 0, brojevi[100];
   float AS = 0;
   int prosti[MAXP], polje[MAXN] = {0};
   scanf("%d", &n);
   for( i = 0; i < MAXN; ++i)
        polje[i] = i+2;
   for(i = 0; i < MAXN; ++i) {
      if(polje[i] != 0) {
         for(k = i+1; k < MAXN; ++k) {
            if(polje[k] != 0) {
               if(polje[k] % polje[i] == 0)
                  polje[k] = 0;
            }
         }
      }
   }
   for(k = 0, i = 0; k < MAXN; ++k){
       if(polje[k] != 0){
            prosti[i] = polje[k];
            i++;
       }
   }
   for(i = 0; i < n; ++i)
        scanf("%d", &brojevi[i]);
   for(k = 0; k < n; ++k){
       for(j = 0; j < n; ++j){
           if(prosti[j] == k){
                AS += brojevi[k];
                brojac++;
           }
        }
   }
   printf("Aritmeticka sredina je: %f", AS/brojac);
   return 0;
}
```

5. ZADATAK (POLJA ZNAKOVA, WHILE) 

Autor: Kristijan Franković

Unesite polje znakova i potom na ekran ispišite kolika je stvarna duljina učitanog polja.

Primjer:



Rješenje:

```
#include <stdio.h>
int main ()
{
	char polje[100+1];
	int brojac = 0,i=0;
	gets (polje);
	while (polje[i]!='\0')
	{
		brojac++;
		i++;
	}
	printf ("Ucitano je %d znakova\n", brojac);
	return 0;
}
```

Napomena:

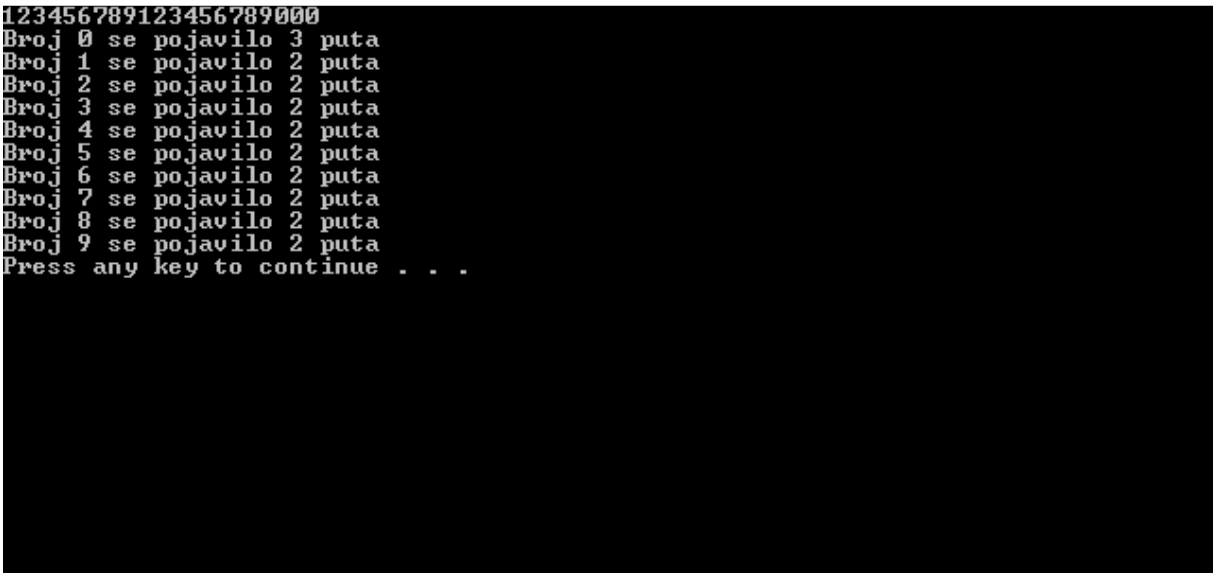
Jednaku funkcionalnost kao gore riješeni zadatak obavlja funkcija strlen iz string.h

6. ZADATAK (POLJA, SWITCH-CASE)

Autor: Alen Rakipović

Učitajte n znamenkast cijeli broj (može se pretpostaviti da neće biti učitan broj koji ima više od 50 znamenki. Potom na ekran računala ispišite koliko se puta pojavila koja znamenka u učitanom broju. Obavezno koristiti switch-case naredbu.

Primjer:



Rješenje:

```c
#include <stdio.h>
int main ()
{
      char broj[50+1];
      int i = 0, brojaci[10] = {0};
      scanf("%s", broj);
      while(broj[i] != '\0'){
          switch(broj[i]){
              case '0': brojaci[0]++; break;
              case '1': brojaci[1]++; break;
              case '2': brojaci[2]++; break;
              case '3': brojaci[3]++; break;
              case '4': brojaci[4]++; break;
              case '5': brojaci[5]++; break;
              case '6': brojaci[6]++; break;
              case '7': brojaci[7]++; break;
              case '8': brojaci[8]++; break;
              case '9': brojaci[9]++; break;
              default: printf("Unio si nesto sto nije znamenka!"); exit(0);
          }
        ++i;
      }
      for (i = 0; i < 10; ++i)
            printf ("Broj %d se pojavilo %d puta\n", i, brojaci[i]);
      return 0;
}
```

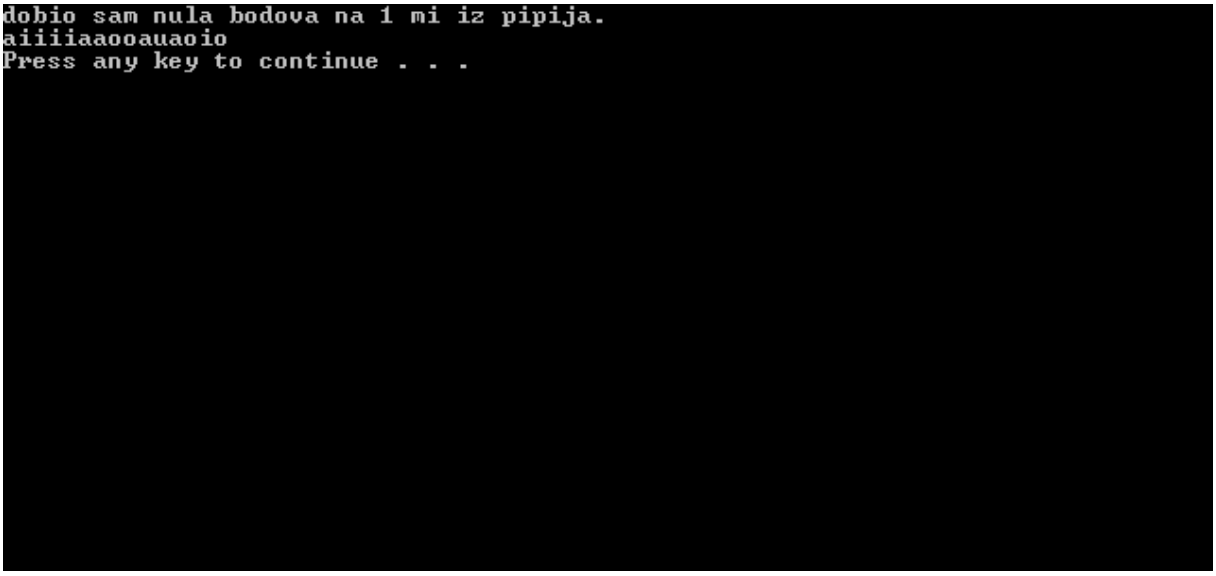
Alternativno rješenje:

```
#include <stdio.h>
#include <string.h>
int main ()
{
	char broj[50+1];
	int i, brojaci[10] = {0};
	scanf("%s", broj);
	for(i=0;i<strlen(broj);i++)
	 brojaci[broj[i]-'0']++;
	for (i = 0; i < 10; ++i)
		printf ("Broj %d se pojavilo %d puta\n", i, brojaci[i]);
	return 0;
}
```

7. ZADATAK (POLJA ZNAKOVA, WHILE)

Učitavati znakove dok se ne unese znak točka ('.'). Pretpostaviti da neće biti uneseno više od 50 znakova i da će zadnji znak uvijek biti točka. Također se može pretpostaviti da neće biti upisana velika slova. Ispisati sve suglasnike obrnutim redoslijedom.

Primjer:



Rješenje:

```
#include <stdio.h>
int main ()
{
	char znakovi[50+1], pomocno[50+1];
	int i=0,j;
	do
	{
		scanf ("%c",&znakovi[i]);
		i++;
	}while (znakovi[i-1]!='.');
	for (j=i-2;j>=0;j--)
		if ((znakovi[j]=='a')||(znakovi[j]=='e')||
		   (znakovi[j]=='i')||(znakovi[j]=='o')||(znakovi[j]=='u'))
		   printf ("%c",znakovi[j]);
	printf ("\n");
	return 0;
}
```

8. ZADATAK (POLJE, WHILE, FOR) 

Autor: Kristijan Franković

Unesite n (maksimalna vrijednost n iznosi 40!) i potom polje popunite sa prvih 0-N **fibbonacijevih brojeva**. Zatim na ekran ispišite sve brojeve iz polja koji su **strogo manji** od aritmetičke sredine svih brojeva u polju.

Obratite pažnju na činjenicu da korisnik programa može unijeti broj veći od 40 ili broj manji od 1 - u tom slučaju je potrebno korisniku ispisati poruku o pogrešci i omogućiti novi unos broja.

Također obratiti pažnju na činjenicu da je aritmetička sredina relan broj, a brojevi u polju su cijeli brojevi -> potrebno je osigurati ispravan ispis ukoliko je primjerice naša aritmetička sredina 89.7, a naše polje sadrži broj 89. (U ovom slučaju mora se ispisati i broj 89, jer 89.7 zaokruženo = 90).

Fibbonacijev niz:

0, 1, 1, 2, 3, 5, 8, 13, 21...

$$F_0 = 0 \quad F_1 = 1 \quad F_n = F_{n-1} + F_{n-2}$$

Primjer:

```
10
0 1 1 2 3 5 8 Press any key to continue . . .
```

Prvih 0 - 10 fibbonacijevih brojeva:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55

Suma = 0 + 1 + 1 + 2 + 3 + 5 + 8 + 13 + 21 + 34 + 55 = 143

Aritmetička sredina = 143/11 = 13 = 13

Potrebno je na ekran ispisati brojeve 0, 1, 1, 2, 3, 5 i 8.

Više o fibbonacijevim brojevima:

http://en.wikipedia.org/wiki/Fibonacci_number

Rješenje:

```
#include <stdio.h>
int main ()
{
	int n, i, suma=0, sredina;
	int brojevi [40+1];
	do
	{
		scanf ("%d",&n);
		if ((n<0)||(n>40))
			printf ("Niste unijeli ispravan broj!\n");
	}while ((n<0)||(n>40));
	brojevi[0]=0;
	brojevi[1]=1;
	for (i=2;i<=n;i++)
		brojevi[i]=brojevi[i-1]+brojevi[i-2];
	for (i=0;i<=n;i++)
		suma += brojevi[i];
	if ((int)(((float)suma/(n+1))*10)%10>=5)
		sredina=suma/(n+1)+1;
	else
		sredina=suma/(n+1);
	i=0;
	while (brojevi[i]<sredina)
	{
		printf ("%d ",brojevi[i]);
		i++;
	}
	return 0;
}
```

DODATNI ZADATAK:

9. ZADATAK (POLJA ZNAKOVA, STRLEN, FOR)

Autor: Kristijan Franković

Unesite rečenicu. Napomena, rečenica mora nužno završiti sa točkom, upitnikom ili uskličnikom! Ukoliko to nije slučaj, ispisati poruku o pogrešci i završiti program! Može se pretpostaviti da rečenica neće sadržavati više od 200 stotine znakova. Ukoliko je rečenica ispravno završila, na ekran ispisati **broj pojavljivanja svakog slova engleske abecede** (evenutalne ostale znakove nije potrebno brojati). Nije potrebno odvojeno brojati velika i mala slova. Također na ekran ispišite koje se slovo u rečenici **pojavljuje najviše puta**, a ukoliko više slova ima jednaku najveću vrijednost pojavljivanja ispišite ono bliže **kraju abecede**.

Uputa:

Engleska abeceda ima 26 slova!

Primjer:

```
Ja cu proci PIPI!
Slovo A se pojavilo 1 puta
Slovo B se pojavilo 0 puta
Slovo C se pojavilo 2 puta
Slovo D se pojavilo 0 puta
Slovo E se pojavilo 0 puta
Slovo F se pojavilo 0 puta
Slovo G se pojavilo 0 puta
Slovo H se pojavilo 0 puta
Slovo I se pojavilo 3 puta
Slovo J se pojavilo 1 puta
Slovo K se pojavilo 0 puta
Slovo L se pojavilo 0 puta
Slovo M se pojavilo 0 puta
Slovo N se pojavilo 0 puta
Slovo O se pojavilo 1 puta
Slovo P se pojavilo 3 puta
Slovo Q se pojavilo 0 puta
Slovo R se pojavilo 1 puta
Slovo S se pojavilo 0 puta
Slovo T se pojavilo 0 puta
Slovo U se pojavilo 1 puta
Slovo V se pojavilo 0 puta
Slovo W se pojavilo 0 puta
Slovo X se pojavilo 0 puta
Slovo Y se pojavilo 0 puta
Slovo Z se pojavilo 0 puta

Najcesce se pojavilo slovo P i to 3 puta
Press any key to continue . . .
```

```c
#include <stdio.h>
#include <string.h>
int main ()
{
	char recenica [200+1], slovo='A', najvece = 'Z';
	int i, slova[26] = {0}, t, kolicina;
	gets(recenica);
	t=strlen(recenica)-1;
	if ((recenica[t]!='.')&&(recenica[t]!='?')&&(recenica[t]!='!'))
	{
		printf ("Greska - recenica ne zavrsava na ispravan nacin!\n");
		return -1;
	}
	for (i=0;i<strlen(recenica);i++)
	{
		if ((recenica[i]>=65)&&(recenica[i]<=91))
			slova[recenica[i]-65]++;
		if ((recenica[i]>=97)&&(recenica[i]<=123))
			slova[recenica[i]-97]++;
	}
	kolicina = slova[25];
	for (i=25;i>=0;i--)
		if (kolicina < slova[i])
		{
			kolicina = slova[i];
			najvece = 65+i;
		}
	for (i=0;i<26;i++)
		printf ("Slovo %c se pojavilo %d puta\n",slovo++, slova[i]);
	printf ("\nNajcesce se pojavilo slovo %c i to %d puta\n", najvece,
kolicina);
	return 0;
}
```

Napomena:

Umjesto ovakvih uvjeta:

```c
if ((recenica[i]>=65)&&(recenica[i]<=91))
	slova[recenica[i]-65]++;
if ((recenica[i]>=97)&&(recenica[i]<=123))
	slova[recenica[i]-97]++;
```

Moglo se napisati:

```c
if ((recenica[i]>='A')&&(recenica[i]<='Z'))
	slova[recenica[i]-'A']++;
if ((recenica[i]>='a')&&(recenica[i]<='z'))
	slova[recenica[i]-'a']++;
```