Napomene:
- *Savjetuje se navedene zadatke riješiti ubrzo nakon predavanja*
- *Savjetuje se ne gledati rješenja prije nego se pokuša samostalno riješiti zadatke*

19. vježbe uz predavanja

U svim zadacima u kojima se traži definiranje funkcije, treba napisati odgovarajući glavni program (tj. funkciju `main`) u kojem ćete po potrebi definirati stvarne argumente, pozvati funkciju i ispisati rezultat.

1. Napišite funkciju koja u zadanom nizu znakova (jednodimenzijskom polju znakova terminiranom sa znakom '\0') pronalazi sve samoglasnike i ispisuje ih na zaslon. Npr. za zadani niz *Antigona*, ispisuje *Aioa*.
2. Napišite funkciju koja iz zadanog niza znakova **izbacuje** sve samoglasnike. Npr. ako se funkciji zada niz znakova *Antigona*, funkcija ga mora **promijeniti** u niz znakova *ntgn*.

3. Što će se ispisati sljedećim programom:

```
#include <stdio.h>

void f (int *p) {
   int i1 = *p;
   int i2 = (*p)++;
   int i3 = *p;
   int i4 = *++p;
   int i5 = *p;
   int i6 = *p++;
   int i7 = *(p-1);
   int i8 = *p;
   printf ("%d %d %d %d %d %d %d %d\n", i1, i2, i3, i4, i5, i6, i7, i8);
}

int main (void) {
   int polje[3][2] = {3, 6, 9, 12, 15, 18};
   f(&polje[0][0]);
   return 0;

}
```

4. Što će se ispisati sljedećim programom:

```
#include <stdio.h>

void f (int *p) {
   printf ("%d %d\n", *p, *p+1);
}

int main (void) {
   int polje[3][2] = {1, 2, 3, 4, 5, 6};
   int *pp;
   pp = &polje[0][0];
   f(pp++);
   f(++pp);
   return 0;
}
```

5. Što će se ispisati sljedećim programom:

```
#include <stdio.h>

void f (int *p) {
   static int i = 2;
   printf ("%d\n", *(p + ++i));
}

int main (void) {
   int polje[3][2] = {1, 2, 3, 4, 5, 6};
   f(&polje[0][0]);
   f(&polje[0][0]);
   f(&polje[0][0]);
   return 0;
}
```

6. Što će se ispisati sljedećim programom:

```
#include <stdio.h>

void f (int *p) {
   static int i = 0;
   i++;
   printf ("%d\n", *(p + --i));
}

int main (void) {
   int polje[3][2] = {1, 2, 3, 4, 5, 6};
   f(&polje[0][0]);
   f(&polje[1][0]);
   f(&polje[1][1]);
   return 0;
}
```

7. Napisati funkciju `genmat` koja u dvodimenzijskom cjelobrojnom polju definiranom s dimenzijama 4 retka i 4 stupca (funkcija radi isključivo s poljima dimenzija 4x4) upisuje četiri jedinice na glavnu dijagonalu.

   U pozivajućem programu definirajte polje, sve članove polja inicijalizirajte na nulu, pozovite funkciju `genmat`, te na zaslon ispišite dobiveni rezultat. Ispis mora izgledati ovako:

```
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

   Nakon toga pokušajte napisati drugačiji glavni program: definirajte polje od 5 redaka i 5 stupaca, inicijalizirajte sve elemente polja na nulu, pozovite **istu** funkciju `genmat` i ispišite svih 5x5 članova polja. Možete li predvidjeti kako će izgledati ispis? Zašto ispisana tablica ne izgleda (što bi se možda moglo očekivati) ovako:

```
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 0
```

8. Napisati funkciju `tablica` koja u dvodimenzijsko cjelobrojno polje definirano u pozivajućem programu upisuje tablicu množenja od m redaka i n stupaca.

   Funkcija mora biti u stanju baratati s poljem proizvoljnih dimenzija. Npr. u glavnom programu može se definirati polje dimenzija 20x10, a zatim pozvati funkciju tako da se polje napuni tablicom množenja dimenzija 5x7; ista funkcija također mora na ispravan način generirati npr. tablicu množenja dimenzija 6x4 u polju koje je u glavnom programu definirano s dimenzijama 30x20.

   U pozivajućem programu definirati polje, s tipkovnice učitati m i n (sigurno manji ili jednaki 15), pozvati funkciju `tablica`, te na zaslon ispisati dobivenu tablicu množenja. Npr. tablica množenja od 3 retka i 4 stupca izgleda ovako:

```
1    2    3    4
2    4    6    8
3    6    9   12
```

9. Napišite funkciju `transp` za transponiranje cjelobrojne matrice od m redaka i n stupaca. Funkcija mora transponirati matricu, ali također i zamijeniti vrijednosti u varijablama m i n pozivajućeg programa u kojima su evidentirane dimenzije matrice.

   U funkciji definirajte, te za transponiranje koristite pomoćno polje. Ideja za korištenje pomoćnog polja: prepisati elemente `mat[i][j]` iz zadane matrice u elemente `pom[j][i]` pomoćne matrice, a zatim svaki element `pom[i][j]` upisati natrag u element `mat[i][j]`.

   Jednako kao u prethodnom zadatku, funkcija mora biti u stanju baratati s poljem proizvoljnih dimenzija, uz ograničenje da zadano polje nikad neće imati dimenzije veće od 100x100.

   Treba napomenuti da je ovakvo rješenje (s pomoćnom matricom) dobro samo za vježbu. Stoga načinite i **bolju (tj. ispravnu)** funkciju koja matricu transponira bez korištenja pomoćnog polja! Primjer transponiranja matrice bez korištenja pomoćnog polja prikazan je na predavanjima o dvodimenzijskim poljima.

Rješenja

Rješenje 1. zadatka

```
#include <stdio.h>

void ispisSamoglasa (char niz[]);

int main (void) {
   char niz[] = "Antigona";
   ispisSamoglasa(niz);
   printf("\n");
   return 0;
}

void ispisSamoglasa (char niz[]) {      /* ili bolje (char *niz) */
   int i = 0;
   while (niz[i] != '\0') {
      if (niz[i] == 'a' || niz[i] == 'A' ||
          niz[i] == 'e' || niz[i] == 'E' ||
          niz[i] == 'i' || niz[i] == 'I' ||
          niz[i] == 'o' || niz[i] == 'O' ||
          niz[i] == 'u' || niz[i] == 'U')
         printf("%c", niz[i]);
      ++i;
   }
}
```

Rješenje 2. zadatka

```
#include <stdio.h>

void izbaciSamoglase (char *niz);

int main (void) {
   char niz[] = "Antigona";
   printf("%s\n", niz);
   izbaciSamoglase(niz);
   printf("%s\n", niz);
   return 0;
}

void izbaciSamoglase (char *niz) {      /* ili (char niz[]) */
   int i = 0, potroseno = 0;
   while (*(niz + i) != '\0') {
      if (*(niz + i) != 'a' && *(niz + i) != 'A' &&
          *(niz + i) != 'e' && *(niz + i) != 'E' &&
          *(niz + i) != 'i' && *(niz + i) != 'I' &&
          *(niz + i) != 'o' && *(niz + i) != 'O' &&
          *(niz + i) != 'u' && *(niz + i) != 'U')
         *(niz + potroseno++) = *(niz + i);
      ++i;
   }
   *(niz + potroseno) = '\0';
  /* VAZNO PITANJE: sto bi bio rezultat bez prethodne naredbe? */
}
```

Ovdje je važno uočiti da u ovoj funkciji **nije moguće*** koristiti pomoćni niz jer nije poznata najveća dopuštena duljina ulaznog niza (a taj je podatak potreban pri definiciji pomoćnog niza). Tek kad bi zadatak glasio: napišite funkciju koja iz zadanog niza znakova izbacuje sve samoglasnike, pri čemu zadani niz **sigurno nije dulji od 1000 znakova**, tada bi postojala mogućnost u funkciji koristiti pomoćni niz (tada bi se u funkciji mogao definirati pomoćni niz duljine 1001 znaka).

* moguće je npr. uz korištenje funkcije malloc, međutim ta se funkcija ne razmatra na ovom predmetu

Rješenje 7. zadatka

```
#include <stdio.h>

void genmat (int *polje) {
   int i;
   for (i = 0; i < 4; ++i)
      *(polje + i + i*4) = 1;
}

int main (void) {
   int mat[4][4] = {0};
   int i, j;
   genmat(&mat[0][0]);
   for (i = 0; i < 4; ++i) {
      for (j = 0; j < 4; ++j)
         printf("%d ", mat[i][j]);
      printf("\n");
   }
   return 0;
}
```

Rezultat:

```
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

Slijedi poziv iste funkcije s poljem koje je definirano s dimenzijama 5x5:

```
int main () {
   int mat[5][5] = {0};
   int i, j;
   genmat(&mat[0][0]);
   for (i = 0; i < 5; ++i) {
      for (j = 0; j < 5; ++j)
         printf("%d ", mat[i][j]);
      printf("\n");
   }
   return 0;
}
```

Rezultat:

```
1 0 0 0 0
1 0 0 0 0
1 0 0 0 0
1 0 0 0 0
0 0 0 0 0
```

Zašto? Funkcija genmat u prvom koraku for petlje upiše vrijednost 1 na adresu *(polje+0+0), a time zapravo upiše jedinicu u element polja mat[0][0], što je u redu. Međutim, kada u drugom koraku for petlje upiše vrijednost 1 na adresu *(polje+1+1*4), zapravo je upisala jedinicu u element polja mat[1][0]. Pojednostavljeno, funkcija (zato što je tako napisana) do prvog člana drugog retka dolazi tako da se preskoči 4 člana polja (a to je istina samo onda kada se funkciji preda polje koje je definirano tako da mu je broj stupaca jednak 4). Očito, ako funkcija mora baratati s poljima različitih dimenzija, **mora** imati informaciju o **stvarnom** broju stupaca polja (broju stupaca pri definiciji polja).

Rješenje 8. zadatka

```
#include <stdio.h>

#define MAXRED 15
#define MAXSTUP 15

void tablica(int *polje, int m, int n, int brclanstup);

int main (void) {
   int m, n;
   int polje[MAXRED][MAXSTUP];
   int i, j;

   printf ("Upisite m i n koji su manji ili jednaki %d i %d: ", MAXRED, MAXSTUP);
   scanf("%d %d", &m, &n);
   /* ovdje bi, naravno, trebalo napraviti provjeru unesenih vrijednosti */
   tablica(&polje[0][0], m, n, MAXSTUP);
   printf("\nSlijedi ispis tablice mnozenja\n\n");
   for (i = 0; i < m; ++i) {
      for (j = 0; j < n; ++j)
         printf("%5d", polje[i][j]);
      printf("\n");
   }
   return 0;
}

void tablica(int *polje, int m, int n, int brclanstup) {
   int i, j;
   for (i = 0; i < m; ++i)
      for (j = 0; j < n; ++j)
         *(polje + i*brclanstup + j) = (i+1) * (j+1);
}
```

Važna napomena: česta pogreška koja se pronalazi u rješenjima ovakvih zadataka jest da se u funkciji umjesto formalnog argumenta `brclanstup` koristi simbolička konstanta MAXSTUP definirana na početku modula. Zašto to ne valja? Zato jer bi tada funkcija ispravno radila **samo s onim poljima** koja su definirana tako da im je broj stupaca jednak MAXSTUP. Npr, ako bi se u glavnom programu definiralo još jedno polje, drugačijih dimenzija, funkcija u tom polju ne bi mogla generirati ispravnu tablicu množenja.

```
   int polje2[10][10];
   ...
   tablica(&polje2[0][0], 5, 6, 10);
   ...
```

Također, pogrešno je koristiti globalnu ili statičku varijablu za "prijenos" informacije o broju stupaca u funkciju.

Rješenje 9. zadatka

```
#include <stdio.h>

#define MAXRED 20
#define MAXSTUP 20

void transp(int *polje, int *m, int *n, int brclanstup);

int main (void) {
   int m, n;
   int polje[MAXRED][MAXSTUP];
   int i, j;
   printf ("Upisite m i n koji su manji ili jednaki %d i %d: ", MAXRED, MAXSTUP);
   scanf("%d %d", &m, &n);
   printf ("Upisite elemente matrice po retcima:\n");
   for (i = 0; i < m; ++i)
      for (j = 0; j < n; ++j)
         scanf("%d", &polje[i][j]);

   printf("\nSlijedi ispis originalne matrice\n\n");
   for (i = 0; i < m; ++i) {
      for (j = 0; j < n; ++j)
         printf("%d  ", polje[i][j]);
      printf("\n");
   }
   transp(&polje[0][0], &m, &n, MAXSTUP);

   printf("\nSlijedi ispis transponirane matrice\n\n");
   for (i = 0; i < m; ++i) {
      for (j = 0; j < n; ++j)
         printf("%d  ", polje[i][j]);
      printf("\n");
   }

   return 0;
}



/*Dimenzije pomocne matrice su 100x100, kako bi se osiguralo da se
  i najveca moguca zadana matrica moze prepisati u pomocnu matricu. */
#define MAXDIMPOMOCNA 100

void transp(int *polje, int *pm, int *pn, int brclanstup) {
   int pom[MAXDIMPOMOCNA][MAXDIMPOMOCNA];
   int i, j;
   int pomocna;
   for (i = 0; i < *pm; ++i)
      for (j = 0; j < *pn; ++j)
         pom[j][i] = *(polje + i*brclanstup + j);
   /* zamijeni *pm i *pn */
   pomocna = *pm;
   *pm = *pn;
   *pn = pomocna;
   for (i = 0; i < *pm; ++i)
      for (j = 0; j < *pn; ++j)
         *(polje + i*brclanstup + j) = pom[i][j];

}
```

```
/* BOLJA funkcija za transponiranje (bez pomocnog polja) */
void transp(int *polje, int *pm, int *pn, int brclanstup) {
   int najvecaDim, i, j, pomClan, pomDim;
   najvecaDim = *pm > *pn ? *pm : *pn;
   for (i = 0; i < najvecaDim - 1; ++i) {
      for (j = i+1; j < najvecaDim; ++j ) {
         pomClan = *(polje + i*brclanstup + j);
         *(polje + i*brclanstup + j) = *(polje + j*brclanstup + i);
         *(polje + j*brclanstup + i) = pomClan;
      }
   }
   pomDim = *pm;
   *pm = *pn;
   *pn = pomDim;
}
```