U ovom dokumentu je kratak tutorial o poljima i pokazivačima i nekoliko zadataka sa prošlih mass instrukcija i jedan zadatak s prošlogodišnjeg međuispita.

Ako uočite grešku, slobodno javite.

Autor: Mario Magdić (Osim za zadatke je gdje je drugačije naznačeno).

Polje

Polje je tip podataka u kojemu se zapisuje više vrijednosti, tj, možemo reći da se unutar polja nalazi više varijabli istoga imena koje se razilikuju po indeksu. Podaci se unutar polja zapisuju redom po indeksu. Počinje s 0,1....,n.

Uzmimo za primjer polje tipa char. Već nam je poznato da varijabla tipa char zauzima 1 byte u memoriji što je jednako 8 bita. Ako definiramo polje char a[3] to znači da su unutar polja nalaze 3 podatatka tipa char, a[0], a[1] i a[2].

Vrijednost a[0] se sastoji od 8 bitova. Od bita 0*8+1 do bita 0*8 + 8. Poziciju nekog char a[n] podatka unutar polja možemo naći ovako: [n*8+1,n*8+8]. Kada bismo imali polje tipa int onda bi bilo [n*32+1,n*32+32].

Primjer: kako se u memoriju zapisuje slijedeće polje?
Napomena: pogledati ASCII tablicu.

```
Char a[3] = {'b','5',5};
```

```
   'b'     '5'    Broj 5
011000100011010100000101
```

Definiranje polja

Jednodimenzionalno polje možemo definirati na dva načina. Jedan od načina je da odmah na početku odredimo broj zapisa unutar polja (npr. int i[65];), a drugi je da definiramo polje neodređene veličine (npr. int i[];).

Vrijednosti članovima polja možemo pridijeliti odmah prilikom definicije ili kasnije. Ako ih pridjeljujemo prilikom definicije onda to radimo ovako:

```
int i[3] = {1,6,9};
```

pri čemu je i[0] = 1, i[1] = 6 i i[2] = 9

Ako unesemo više vrijednosti nego što smo definiriali veličinu polja compiler će prijaviti grešku. Unošenjem manje vrijednosti od veličine polja se sva ostala polja postavljaju na nulu.

Npr. `int i[3] = {4};`

i[0] = 4, i[1] = 0, i[2] = 0

Stringovi

U C programskom jeziku ne postoji string kao tip podatka već se string javlja kao polje tipa char. String možemo definirati na prethodno prikazan način:

```
char c[3+1] = {'F','E','R','\0'};
```

Zašto 3+1?

U string želimo upisati tri znaka ali na kraj stringa moramo staviti znak \0.

String ispisujemo na slijedeći način:

```
Printf("%s",c);
```

Rezultat: na ekran se ispisuje FER.

Što bi se dogodilo kada bismo pokušali ispisati ovakvo polje:

```
char c[3] = {'F','E','R'}; ?
```

Na ekran bi se ispisalo nešto tipa FER#/?ff`^$+.

Zašto?

Zato što funkcija printf ispisuje sve vrijednosti polja dok ne dođe do znaka \0. Pošto mi ne znamo što se u memoriji nalazi nakon naša tri znaka unutar polja ne znamo što će se točno ispisati. Printf će nastaviti kopati po memoriji sve dok ne

naiđe na znak \0 (uz pretpostavku da se znak \0 nalazi negdje u memoriji iza naša tri znaka).

Drugi način za definiranje stringa je:

```
char c[] = "PIPI";
```

Koliko bytea zauzima ovo polje?

Ovo polje je veličine 5 Bytea. Četiri bytea za riječ PIPI +1 za znak \0. Kada string definiramo na ovaj način na kraj se uvijek dodaje znak \n.

Pokazivači

Pokazivači (engl. Pointer) nam služe za pohranjivanje adresa varijabla koje koristimo.

Pointer je isto varijabla, ali nije int iako je slična varijabli tipa integer. Veličina pointera ovisi o operativnom sustavu/hardveru. Na 32-bitnom sustavu je to 32 bita, na 64-bitnom 64...

Definicija pointera:

Pointer deklariramo slično kao i ostale varijable.

```
tip *ime_pokazivaca;
```

npr, ako želimo definirati pointer na integer to radimo ovako:

```
int *p;
```

Pointeru pridružujemo adresu varijable na slijedeći način:

```
int i, *p;

p = &i;
```

Kada želimo pristupiti adresi neke varijable koristimo &prije imena. Dakle ako pointeru **p** želimo pridružiti adresu varijable **i** to radimo ovako: p = &i;

Za pristup vrijednosti varijable preko pointera koristimo znak * ispred imena pointera.

Dakle, ako varijabli i iz prethodnog primjera želimo dodijeliti vrijednost 5 korištenjem pokazivača to radimo ovako:

```
*p = 5;
```

Evo jednog primjera:

```
#include <stdio.h>

int main() {

    int varijabla, *pokazivac;

    varijabla = 5;

    pokazivac = &i;
    *pokazivac++;

  return 0;

}
```

Primjeri:

1. Autor: Kristijan Franković

Napišite funkciju koja prima n i potom kao rezultat preko imena funkcije vraća n! (n faktorijela). Napisati i glavni program koji učitava n, pomoću funkcije izračunava n! I potom rezultat ispisuje na ekran.

Rješenje:

```
#include <stdio.h>
int faktorijeli (int a)
{
            int rez = 1; // ako je broj 0 onda je rezultat 1
            while (a>0)  // pa se while petlja nece ni izvrsiti
            {
                        rez*=a;  //postupak obavljamo sve dok se ne
izmnoze svi brojevi
                        a--;
            }
            return rez;  //vrati rezultat
}
```

```
int main ()
{
		int n;
		scanf ("%d",&n);
		printf ("%d\n",faktorijeli(n));
		return 0;
}
```

1. Napišite funkciju koja zamijeni vrijednosti dviju varijabli.

Problem u ovom zadatku je to što funkcije ne mogu vraćati više od jedne varijable. Zato u ovom zadatku moramo koristiti pokazivače (pointere) jer pomoću njih možemo direktno pristupiti adresama u memoriji.

```
#include <stdio.h>

void zamijeni(int *a, int* b) { //funkcija ne vraca podatke pa je void

		int temp;	//temp je pomocna varijabla

		temp = *a;	 //spremi prvu varijablu u temp

		*a = *b;	//spremi drugu varijablu u drugu

		*b = temp;	//spremi temp u drugu varijablu

		return;		// vrati nista

}

int main() {

		int i,x;

		i = 5; x = 6445;

    zamijeni(&i,&x); //pozivamo funkciju i predajemo joj adrese varijabli i
i x

		return 0;
```

```
}
```

3. Napisati program koji pomoću `gets` naredbe učitava niz znakova. Možete pretpostaviti da učitani niz neće imati više od 80 znakova. Ispisati koje se veliko slovo abecede pojavilo najviše puta u učitanom nizu znakova, te ispisati broj pojavljivanja tog slova. Ukoliko su se različita slova pojavila jednak broj puta, ispisati ono slovo koje se

nalazi bliže kraju abecede. (Pretpostaviti da će se sigurno pojaviti barem jedno veliko slovo.)

Primjer.

za učitani niz: `LOTUS 123 je prvi Tablicni KALKULATOR` treba ispisati: `Slovo T pojavilo se 3 puta (prvi zadatam s 2. međuispita 2008/09.`

Da bi riješili ovaj zadatak moramo imati brojač za svako veliko slovo. To ćemo riješiti tako da u polje za svako veliko slovo koje se pojavi dodajemo +1 kada se pojavi. Sve elemente polja treba inicijalizirati na nulu prije početka brojanja jer bi u protivnom dodavali +1 nekim brojevima iz memorije.

Kada bi za indekse polja koristili redne brojeve iz ASCII tablice dobili bi polje sa puno neiskorištenih vrijednosti. Nama su potrebni samo znakovi od A-Z. Kako bi postigli da indeks slova A bude nula radimo r[d] – 'A', tj. oduzimamo ASCII vrijednost znaka u polju r na poziciji d od vrijednosti znaka 'A'.

U riječi KALKULATOR u r[0] zapisano je slovo 'K'. Njegov redni broj 75 oduzimamo od rednog broja slova 'A' (65) i tako dobivamo 75 – 65 = 10. U polje brVelika na indeksu 10 dodajemo +1. Isti postupak se izvršava sve dok ne dođemo do znaka '\0' koji označava kraj stringa.

```
#include <stdio.h>
#define MAXZNAK 80 //pretpostavka je da broj znakova nece biti veci od 80

int main(){

   char r[MAXZNAK+1];  //definirajmo polje u koje stane 80 znakova + \0
znak
   int brVelika['Z'-'A' +1] = {0}; //definirajmo brojac za sva velika slova
   int i, d, index;
   int maxVelikoSlovo = 0;
   printf("Unesite recenicu:\n");
   gets(r);  //funkcija gets ucitava string s tipkovnice i upisuje ga u
polje r
   d = 0;

   while (r[d]){  //postupak se ponavlja sve dok se dodje do znaka \0 koji
vrijedi isto kao i logicka nula

     if(r[d] >= 'A' && r[d] <= 'Z'){ //ako je znak veliko slovo
```

```
                        index = r[d] - 'A'; //oduzmi redni broj znaka
od rednog broja slova 'A'
          brVelika[index]++; //dodaj jedan u counter za taj znak
      }
     d++; //prijedji na iduce slovo u stringu
    }

    //dalje koristimo poznati algoritam za odredjivanje varijable s najvecom
vrijednoscu
    for (i ='Z'; i >= 'A'; i--){

                if (brVelika[i-'A'] > brVelika[maxVelikoSlovo])
maxVelikoSlovo = i-'A';
    }

    printf ("Slovo %c, pojavilo se %d puta.\n",
    maxVelikoSlovo + 'A', brVelika[maxVelikoSlovo]);

    return 0;
}
```

4.Napisati funkciju kojoj predajemo tri varijable tipa float redom a, b, c tako da na mjesto jedne varijable unosimo 0, a funkcija potom pitagorinim poučkom izračunava vrijednost varijable koja je nula.

Ako unesemo a = 3, b = 5, c = 0 funkcija treba računati c = sqrt (a*a + b*b), a ako unesemo a=0, b = 5, c = 8 funkcija treba računati a = sqrt (c*c – b*b). Pretpostavimo da korisnik nije unio dvije vrijednosti 0 za varijable.

```
#include <math.h> //math.h sadrzi funkciju za racunanje drugog korijena

float pitagorin_poucak(float a, float b, float c) {

            if(!a) {

                        return sqrt( c*c - b*b );

            }

            if(!b) {
```

```
                return sqrt( c*c - a*a );

        }

        if(!c) {

                return sqrt( a*a + b*b);

        }

}
```

5. Autor: Siniša Matetić

Učitavati broj redaka i stupaca matrice (neće biti veći od 20) sve dok korisnik ne unese valjane brojeve, a zatim učitati elemente matrice. Ispisati najmanji element desne polovice matrice i najveci element na donjoj polovice matrice!

```
#include <stdio.h>
#define MAXR 20
#define MAXS 20

int main()
{
        int i, j, redaka, stupaca;
        int min, max;
        int matrica[MAXR][MAXS];

    //ucitavaj sve dok se ne korisnik ne unese valjane brojeve

        do
        {
                scanf("%d %d", &redaka, &stupaca);
        } while (redaka < 1 || redaka > MAXR || stupaca < 1 ||
stupaca > MAXS);

    //ucitavanje elemenata matrice

        for (i = 0; i < redaka; ++i)
                for (j = 0; j < stupaca; ++j)
                        scanf("%d", &matrica[i][j]);

        max = matrica[redaka-1][stupaca-1];
        min = matrica[redaka-1][stupaca-1];
```

```
            for(i=0 ; i<redaka ; i++)
                        for(j=stupaca/2 ; j< stupaca;j++)  //prvo
trazimo najmanji element desne polovice
                                    if(matrica[i][j]<min)
//u desnoj polovici su svi stupci za koje je j veci od broja stupaca/2
                                                min=matrica[i][j];


            for(i=redaka/2;i<redaka;i++)          //isti postupak za
najveci element donje polovice matrice
                        for(j=0;j<stupaca;j++)            //to su svi
redovi za koje je i veci od broja redova/2
                                    if(matrica[i][j]>max)
                                                max=matrica[i][j];
            printf("%d %d", min, max);
            return 0;
}
```

6. Autor: Siniša Matetić

Učitati broj redaka i stupaca matrice (neće biti veći od 20), a zatim elemente matrice. Konstruirati novu matricu na temelju prve tako da je element (r, s) koji se nalazi u retku **r** i stupcu **s** jednak umnošku zbroja elemenata r-tog retka i zbroja elemenata s-tog stupca koji ne uključuju element na poziciji r,s (prve matrice).

```
#include <stdio.h>
#define MAXR 20
#define MAXS 20

int main()
{
            int i, j, k, redaka, stupaca,zi=0,zj=0;
            int prva[MAXR][MAXS], druga[MAXR][MAXS];

            scanf("%d %d", &redaka, &stupaca); //ucitaj dimenzije matrice
            for (i = 0; i < redaka; ++i)       //....
                        for (j = 0; j < stupaca; ++j)
                                    scanf("%d", &prva[i][j]);

   //postupak stvaranja druge matrice je slican onome za ucitavanje
elemenata samo sto vrijednosti dobivamo racunajuci s vrijednostima prve

            for (i = 0; i < redaka; ++i)
                        for (j = 0; j < stupaca; ++j)
```

```
                        {
                                zi=0;  //zbroj elemenata reda
stavimo na 0
                                zj=0;  //.... stupaca na 0
                                for (k=0;k<stupaca;++k) //ova
petlja se krece kroz trenutni redak prve matrice

zi+=prva[i][k];     //i racuna sumu svih elemenata tog retka
                                zi-=prva[i][j];        //na kraju
oduzmemo element na poziciji i,j od sume

                                for(k=0;k<redaka;++k) //identican
postupak je za sumu elemenata stupca
                                        zj+=prva[k][j];
                                zj-=prva[i][j];      //oduzimanje
elementa na poziciji i,j

                                druga[i][j]=zi*zj;   //umnozak
suma
                        }

                        //ostatak koda sluzi za ispis druge matrice
        for (i = 0; i < redaka; ++i)
        {
                        for (j = 0; j < stupaca; ++j)
                                printf("%d ", druga[i][j]);
        printf("\n");
        }
        return 0;
}
```

7. Autor: Siniša Matetić

Učitati veličinu kvadratne matrice (neće biti veća od 20), a zatim elemente matrice. Ispisati sumu elemenata koji tvore pješčani sat kada se povuku glavna i sporedna dijagonala!

```
#include <stdio.h>
#define MAXD 20
int main()
{
        int i, j, n;
        int  suma=0;
        int matrica[MAXD][MAXD];

        scanf("%d", &n);
```

```
            for (i = 0; i < n; ++i)
                        for (j = 0; j < n; ++j)
                                    scanf("%d", &matrica[i][j]);

            for(i=0 ; i<n/2;i++)
                        for(j=i ; j<n-1 ; j++)
                                    suma+=matrica[i][j];

            for(i=n/2 ; i<n ;i++)
                        for(j=i ; j>=n-i-1 ;j--)
                                    suma+=matrica[i][j];

            printf("Suma: %d", suma);
            return 0;
}
```

Primjer:

3 5 7 0
2 **8 4** 4
6 **7 8** 1
9 4 5 2

Pjescani sat se sastoji od dva zrcalno simetricna trokuta koji se nalaze na svojim polovicama matrice. Prvi se nalazi na gornjoj i u prvom retku sadrzi najvise elemenata matrice, a u svakom iducem retku sve manje dok se ne dođe do pola.

Za drugi trokut vrijedi suprotno. Više elemenata matrice što se više udaljavamo od pola.

```
            for(i=0 ; i<n/2;i++) //citaj redove do pola
                        for(j=i ; j<n-1 ; j++) // gledaj elemente reda pocevsi od stupca koji ima isti redni broj kao i red. (i == j)
                                    suma+=matrica[i][j];
```