Polja

Ciljevi:

- upoznati se sa pojmom polja
- proučiti vrste/tipove polja
- shvatiti način rada s poljima
- naučiti raditi s poljima koristeći do sada obrađene programske konstrukte

Pregled lekcije

Definicija polja

Polje je mehanizam agregacije kojim se od jednostavnih tipova podataka tvori složeniji tip podataka. Ako uđemo malo dublje u samu strukturu, možemo reći da polje nije ništa drugo nego niz varijabli istog tipa koje su imenovane zajedničkim identifikatorom i nalaze se na uzastopnim memorijskim lokacijama. Svaka varijabla u polju naziva se element polja.

Sintaksa deklaracije polja:

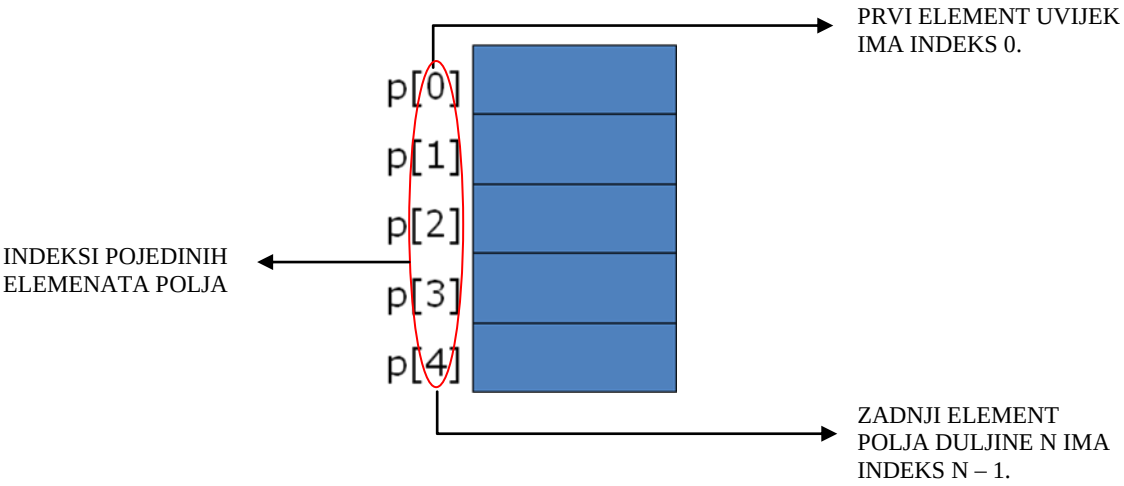
```
tip_podataka identifikator[broj_elemenata];
```

Napomena: Imajte uvijek na umu da su varijable u polju uvijek istog tipa podataka te da su smještene, kao što je i prethodno rečeno, na uzastopnim memorijskim lokacijama. Zbog toga je dovoljno pamtiti samo memorijsku lokaciju nultog elementa polja s obzirom da se memorijske lokacije ostalih elemenata mogu izračunati na sljedeći način:

```
lokacija 0-tog elementa + veličina elementa × index.
```

Zbog tog svojstva se svakom elementu polja može pristupiti u fiksnom vremenu, bez obzira na indeks elementa i veličinu polja.

Svaka varijabla (element) u polju ima svoj indeks preko kojega joj se pristupa. Za indeks možemo reći da predstavlja memorijski pomak u odnosu na adresu početka polja.



Slika 1: Elementi polja i indeksi

Primjer deklaracije polja:

`int a[10]` – deklarirano je polje cijelih brojeva koje se zove a i ima ukupno 10 elemenata. Indeksi elemenata polja idu od 0 do 9.

`float X[8]` – deklarirano je polje decimalnih brojeva koje se zove X i ima ukupno 8 elemenata. Indeksi elemenata polja idu od 0 do 7.

`char b[30]` – deklarirano je polje od 30 znakova. Indeksi elemenata polja idu od 0 do 29.

Deklaracijom polja pojedini elementi polja još uvijek nisu inicijalizirani, stoga su vrijednosti koje se nalaze u svakom elementu polja neki slučajan sadržaj, ovisno o tome što se nalazi u memoriji na tom mjestu. Ako inicijalizirati polje ili pojedine elemente polja prilikom njegove deklaracije možemo to učiniti na sljedeći način:

int A[] = {1000, 500, 200, 100, 50, 20, 10, 5, 2, 1};

A[0]	**A[1]**	**A[2]**	**A[3]**	**…**
1000	500	200	100	…

4 B

40 B

Prevoditelj će automatski znati da prethodno polje sadrži deset elemenata bez obzira što mu to nismo specificirali brojem u uglatim zagradama nakon naziva polja. To je tako jer se s obzirom na inicijalizirane vrijednosti elemenata jasno vidi koja je dužina polja. Ako pak želimo navesti dužinu polja uz inicijalizaciju, ona obavezno mora biti veća ili jednaka broju inicijaliziranih elemenata inače će doći do greške.

Imajte na umu da polje ne smije imati isti identifikator kao neka već deklarirana varijabla koja u istom dosegu kao polje. Isto tako pazite da kada radite s poljima ne koristite indekse izvan dosega jer se može dogoditi da slučajno zapišete određenu vrijednost na mjesto u memoriji koje koristi neka vaša varijabla ili možda operacijski sustav.

Operacije nad poljima

Sada kada smo se upoznali sa pojmom polja, razmotrimo neke operacije koje se mogu izvoditi nad njima. Recimo da želimo napraviti algoritam koji će primiti vrijednosti N elemenata polja (cijelobrojnih vrijednosti) i iz danog niza pronaći najmanji element polja.

Primjer 1:

ULAZ: Niz cijelih brojeva
IZLAZ: Najmanji broj u nizu

1. I = 0
2. Upiši N
3. Za svaki N upiši član u niz
4. Neka je prvi član niza najmanji
5. Sve dok je I < N radi
6. Ako je I-ti element niza manji od najmanjeg
   Neka je I-ti najmanji
7. Ispiši najmanji

Ovaj se problem može riješiti sa jednodimenzionalnim poljem (vrsta polja koju smo do sada spominjali) i dvije for petlje kako je prikazano u sljedećem primjeru.

Primjer 1.1

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main()
{
    int polje[100];
    int N,i,min;

    do {
    cout<<"Unesi N:";
    cin>>N;
    } while (N < 1 || N > 99);

    for(i = 0; i < N; i++){
    cout<< "\nUnesite clan u polje[" << i << "] = ";
    cin>> polje[i];
    }

    min=polje[0];
```

```
    // traženje najmanjeg clana u nizu
    for(i = 0; i < N; i++)
    if(polje [i] < min)
    min = polje[i];

    cout<< "\n Najmanji u nizu je:"<<min<<"\n\n";

    cout << "KRAJ!";
    system ("PAUSE");

    return 0;
}
```

U prethodnom primjeru deklarirali smo polje od 100 elemenata. Nakon toga unosimo za koliko ćemo elemenata tražiti najmanji element tj. koliko ćemo vrijednosti unijeti u polje prije traženja najmanje vrijednosti u nizu. Kada unesemo sve željene vrijednosti postavljamo vrijednost prvog elementa polja (sa indeksom 0) u vrijablu min. Nakon toga ulazimo u drugu for petlju i gledamo dali je možda neki drugi element niza manji od onoga koji je u varijabli min i ako je, smjestimo taj element u varijablu min. Ovaj postupak ponavljamo sve do kraja for petlje tj. do kraja svih elemenata polja. Na kraju se ispiše najmanji broj i program završava. Dakle, u ovom primjeru je napravljena operacija pretraživanja jednodimenzionalnog polja pri čemu smo tražili najmanji broj u nizu.

Do sada smo se već sreli sa tipom podataka string, međutim nije bilo rečeno da je string zapravo jedna specijalna vrsta polja odnosno string je polje tipa char. Stoga ekvivalent string varijabli možemo deklarirati koristeći polja na sljedeći način:

```
char S[20];
```

Isto tako na kraju svakog stringa nalazi se oznaka '\0' koja označava završetak stringa. Zbog toga treba uvijek imati na umu da polje mora biti za jedan karakter veće od najveće dozvoljene duljine niza znakova (stringa) u njemu. Promotrimo prvo sljedeći primjer s obzirom da string za razliku od ostalih polja funkcionira nešto drugačije. Dakle, kada budemo govorili o stringu u sljedećem tekstu podražumijevamo zapravo polje tipa char.

Primjer 2:

ULAZ: Unijeti riječ S

IZLAZ: Provjeriti je li S palindrom (riječ koja se čita sprijeda i straga jednako).

1. Učitaj riječ S
2. Neka je L duljina riječi S
3. Za svako I=1..N/2
4. Ako je I-to slovo riječi S različito od (L-I)-tog slova, onda riječ nije palindrom

Potrebno je unijeti jednu riječ i vidjeti dali je ta riječ palindrom. S obzirom na zadatak dobivamo sljedeći programski kod.

Primjer 2.1

```
#include <iostream>
#include <cstring>
using namespace std;

int main ()
{
    char S[20];
    bool pal = true;

    cout << "S = ";
    cin >> S;

    int L = strlen(S);

    for (int i=0; i<L/2; i++)
        if (S[i] != S[L-i-1])
           pal = false;

    if (pal)
       cout << "Rijec je palindrom"
            << endl;
    else
       cout << "Rijec nije palindrom"
            << endl;

    cout << "\nKRAJ!";
    system ("PAUSE");

    return 0;
}
```

Deklarirali smo polje od 20 znakova odnosno string i varijablu tipa bool. Nakon toga unosimo podatke u deklarirani string što radimo koristeći ulazni tok cin. Iako općenito polja nije moguće učitavati kao jednu varijablu sa stringom je to moguće, dakle našim unosom se popunjava onoliko elemenata koliko smo unijeli. Nadalje, pomoću funkcije strlen gledamo koliko je taj string dugačak (ovo bismo mogli i ručno tražeći znak '\0') i potom u for petlji uspoređujemo redom znakove krečući se istovremeno od početka/kraja. Potom se ispisuje dali je palindrom pronađen ili nije s obzirom na to dali smo postavili varijablu pal na false ili nismo.

Još neke korisne funkcije za rad sa stringovima su:

cin.get() ⟶ učitava jedan znak sa tipkovnice (ako je uneseno više slova učitava se prvi znak a ostali znakovi ostaju u spremniku za sljedeće unose, dio je iostream biblioteke),

cin.getline(varijabla, max_duljina) ⟶ učitava znakove sve do kraja retka bez obzira na razmake kao jedan string (dio je iostream biblioteke),

cin.ignore() ⟶ briše sadržaj spremnika kako se ne bi unio neželjeni sadržaj koji je zaostao od prije (dio je iostream biblioteke).

Demonstrirajmo rad prethodno opisanih naredbi na primjeru. Izradimo algoritam u kojem ćemo pomoću Cezarove šifre kriptirati unesene podatke.

Primjer 3:
ULAZ: Tekst S i pozitivni cijeli broj N.
IZLAZ: Tekst S' koji je dobiven tako da su najprije mala slova pretvorena u velika, a nakon toga je svako slovo iz S pomaknuto za N mjesta u ASCII tablici

1. Učitaj N
2. Učitaj tekst S
3. Velika slova pretvoriti u mala
4. Svako slovo pomaknuti za N mjesta dalje u ASCII tablici, ali tako da ni jedno slovo ne premaši 'Z'.
5. Ispisati S'

Napomena: Specijalni slučaj ove kriptografske tehnike je kada je N = 13 i naziva se ROT13.

Primjer 3.1

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    int N;
    char S[1000];

    do {
       cout << "Kljuc: ";
       cin >> N;
    } while (N < 1 && N > 25);

    cout << "\nUnesi tekst za kriptiranje: ";
    cin.ignore();
    cin.getline(S,1000);

    int L = strlen(S);

    for (int i = 0; i < L; i++) {

        if (S[i] >= 'a' && S[i] <= 'z')
           S[i] -= 32;

        if (S[i] >= 'A' && S[i] <= 'Z')
           S[i] += N;

        if (S[i] > 'Z')
           S[i] -= 26;
```

```
    }
    cout << "\n\tKriptirano sa N = " << N << "\tSadrzaj = " <<
S << endl;

    system("pause");

    return 0;
}
```

Jedino što je ovdje možda nejasno je zašto unosimo pomak baš u danom rasponu te zašto oduzimamo baš ove vrijednosti. Kako bismo razjasnili odgovor na to pitanje pogledajmo ASCII tablicu na sljedećoj stranici.

Tabela 1: ASCII tablica kodova

Znak	*Dec*	*Oct*	*Hex*	*Znak*	*Dec*	*Oct*	*Hex*	*Znak*	*Dec*	*Oct*	*Hex*	*Znak*	*Dec*	*Oct*	*Hex*
(nul)	0	0000	0x00	(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
(soh)	1	0001	0x01	!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
(stx)	2	0002	0x02	"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
(etx)	3	0003	0x03	#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
(eot)	4	0004	0x04	$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
(enq)	5	0005	0x05	%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
(ack)	6	0006	0x06	&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
(bel)	7	0007	0x07	'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(bs)	8	0010	0x08	(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
(ht)	9	0011	0x09	)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
(nl)	10	0012	0x0a	*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
(vt)	11	0013	0x0b	+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
(np)	12	0014	0x0c	,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
(cr)	13	0015	0x0d	-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
(so)	14	0016	0x0e	.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
(si)	15	0017	0x0f	/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
(dle)	16	0020	0x10	0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
(dc1)	17	0021	0x11	1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
(dc2)	18	0022	0x12	2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
(dc3)	19	0023	0x13	3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
(dc4)	20	0024	0x14	4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
(nak)	21	0025	0x15	5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
(syn)	22	0026	0x16	6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
(etb)	23	0027	0x17	7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
(can)	24	0030	0x18	8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
(em)	25	0031	0x19	9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
(sub)	26	0032	0x1a	:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
(esc)	27	0033	0x1b	;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
(fs)	28	0034	0x1c	<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
(gs)	29	0035	0x1d	=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
(rs)	30	0036	0x1e	>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
(us)	31	0037	0x1f	?	63	0077	0x3f	_	95	0137	0x5f	(del)	127	0177	0x7f

Dakle, ako bismo unosili pomak u drugom rasponu ne bismo dobili željeni izlaz. Isto tako, s obzirom na poziciju velikih i malih slova u ASCII tablici moramo oduzimati 32 kod pretvaranja malih u velika slova, te na kraju ako se radi o nekom specijalnom znaku oduzmemo 26 tako da opet krenemo od prvog slova u engleskoj abecedi (ako je znak samo za jedan veći od Z). Ista logika slijedi i za ostale slučajeve.

ASCII (**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange) je standardni 7-bitni kod predložen od strane ANSI (American National Standards Institute) 1963. a dovršen 1968. ASCII tablica kodova je osmišljena s ciljem da se ostvari kompatibilnost u načinu prikaza podataka, tako da bi različita oprema za obradu podataka mogla skladnije komunicirati.

Standardni ASCII kod se sastoji od 128 cjelobrojnih vrijednosti, u intervalu od 0 do 127, i svakoj vrijednosti je dodijeljen jedan znak. Uz standardne znakove abecede tu se nalaze i ostali specijalni često korišteni znakovi. Pošto računala znaju raditi jedino s brojevima bilo je potrebno svakom znaku dati ekvivalenti broj. S obzirom na evoluciju današnje tehnologije s vremenom se pojavila potreba za dodatnim znakovima pa je tako nastala i proširena ASCII tablica kodova u kojoj se nalaze dodatni znakovi kojih nema u standardnoj ASCII tablici.

Napomena: osim ASCII tablice kodova postoje i drugi formati kodiranja znakova međutim to su u pravilu interni standardi određenih organizacija kao što je IBM-ov **EBCDIC** (**E**xtended **B**inary **C**oded **D**ecimal **I**nterchange **C**ode) koji je vjerojatno jedan od poznatijih načina kodiranja znakova odmah poslije ACSII kodova. IBM ga koristi u svojim mainframe računalima.

Dvodimenzionalna polja

Do sada smo radili sa jednodimenzionalnim poljima, međutim u praksi se iznimno često javlja potreba za rad s dvodimenzionalnim poljima odnosno matricama. U jednodimenzionalnim poljima se članovi polja dohvaćaju preko jednog indeksa dok se kod dvodimenzionalnih polja to radi s dva indeksa, pri čemu je prvi indeks red matrice dok drugi predstavlja stupac. Recimo da imamo sljedeći primjer:

```
float mat[2][4];
```

U prethodnom retku smo deklarirali polje tipa float identifikatora mat, dimanzija 2 reda i 4 stupca. Članovi nisu inicijalizirani. Prikažimo ovu matricu na slici:

mat[0][0]	mat[0][1]	mat[0][2]	mat[0][3]
mat[1][0]	mat[1][1]	mat[1][2]	mat[1][3]

Ako bolje promotrimo ovo polje vidjet ćemo da je svaki redak ovog dvodimenzionalnog polja zapravo zasebno jednodimenzionalno polje. Svakom se elementu pristupa preko oba indeksa, dakle prvo ide red a zatim stupac i naravno kao i prije indeksi počinju od nule. Kao i kod jednodimenzionalnog polja većina operacija se svodi na prolazak kroz petlju međutim ovdje su nam u većini slučajeva potrebne dvije s obzirom da s jednom prolazimo kroz redove a s drugom kroz stupce (radi se o tzv. ugnježđenim petljama).

Ako želimo inicijalizirati dvodimenzionalno polje pravila su ista kao i za jednodimenzionalno međutim redovi se odvajaju zarezom, izostavimo li vitičaste zagrade inicijalizacija počinje od prvog reda pa sve do zadnjeg člana u tom redu i tek tada se preskače u drugi red.

```
char DvoDim[3][3] = {{'x', 'x', 'o'},
                     {'o', 'o', 'x'},
                     {'o', 'x', ' '}};
```

Što se tiče polja od više dimenzija, moguće ih je koristiti, međutim u praksi im je uporaba iznimno rijetka. Primjer jednog takvog polja bio bi:

```
int TriDim [x][y][z];
```

pri čemu su x, y, z indeksi elemenata kao i prije.

Isto tako kako se povećava broj dimenzija tako raste zauzeće memorije, npr. jedno relativno malo trodimenzionalno polje '`TriDim [5][5][5]`' ima $5 \times 5 \times 5 = 125$ elemenata.

Pokažimo uporabu dvodimenzionalnih polja na sljedećem primjeru. Recimo da želimo pronaći minimum i maksimum u polju stringova.

Primjer 4:

ULAZ: Unijeti niz riječi S[0]..S[I-1] . Niz S1 može imati najviše 20 elemenata. Ako se unese prazan string "" unos niza se prekida.
IZLAZ: Pronaći najmanju i najveću riječ u nizu

1. I = 0
2. Radi
3. Unesi S1[i]
4. I = I + 1
5. sve dok je I<20 i S1[i] != ""
6. Ako je (S[I] = "") I = I -1
7. Max = Min = S[0]
8. Za J = 1..I-1 radi
9. Ako je (S[j] > Max) onda Max = S[j]
10. Ako je (S[j] < Min) onda Min = S[j]
11. Ispiši Min, Max

Primjer 4.1:

```
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
    char S[100][20];
    int I = 0;

    do {
```

```
        cout << "Upisite tekst:" << endl;
        cin.getline(S[I],100);
    } while (I < 19 && strcmp(S[I++],""));

    char Max[100], Min[100];

    strcpy(Max, S[0]);
    strcpy(Min, S[0]);

    for (int J = 1; J < I; J++) {
        if (strcmp(S[J], Max) == 1) strcpy(Max, S[J]);
        if (strcmp(S[J], Min) == -1) strcpy(Min, S[J]);
    }

    cout << "Max = " << Max << endl;
    cout << "Min = " << Min << endl;

    system("pause");

    return 0;
}
```

Iako su tokovi cout i cin napravljeni tako da se stringovi mogu učitavati i ispisivati kao jednostavne varijable, oni to u jeziku C++ jeziku nisu. Stoga im ne možemo pomoću operatora = pridruživati vrijednosti niti ih možemo uspoređivati pomoću operatora uspoređivanja. Kako ne bismo ove operacije ručno morali implementirati (prolaziti kroz svaki znak zasebno) koristimo naredbe strcpy(*u_string*, *iz_stringa*) te strcmp(*string1*, *string1*), obje funkcije se nalaze u cstring biblioteci.

S time da strcmp radi na sljedeći način:

- Ako je *string1* < *string2*, vraća -1
- Ako je *string1* == *string2*, vraća 0
- Ako je *string1* > *string2*, vraća 1

Zadaci za vježbu

1. Odgovorite na pitanje što je to polje.

2. Objasnite razliku između jednodimenzionalnih i dvodimenzionalnih polja.

3. Koje vrijednosti vraća funkcija strcmp, što te vrijednosti znače i čemu služi ta funkcija.

4. Izradite algoritam koji će prirodne brojeve N <= 1000 pretvarati u različite brojevne sustave s time da korisnik unosi bazu B.

Programski primjeri za laboratorijske vježbe

Zadatak 1.

Izradite program u kojem:

1. Se unosi 5 brojeva (brojevi mogu biti cijeli i decimalni) u jednodimenzionalno polje tipa float i potom ispišite unesene brojeve obrnutim redoslijedom.

2. Pronađite najmanji i najveći broj u polju iz prethodne točke i ispišite ih na zaslon, pretvorite te min i max brojeve u cjelobrojne vrijednosti, te cjelobrojne vrijednosti modularno podijelite (max/min) te rezultat ispišite. Nakon toga deklarirajte i inicijalizirajte novo polje koje će biti tipa char s time da to novo polje mora biti duljine konstante vrijednosti 4 te ispišite vrijednosti svih članova polja do kraja stringa.

3. Izradite algoritam koji će izračunati prvih M brojeva Fibonaccijevog niza F_n za n = 0, 1. Fibonaccijev niz definiran je sljedećom rekurzivnom relacijom:

$$F(n) := \begin{cases} 0 & n = 0; \\ 1 & n = 1; \\ F_{n-1} + F_{n-2} & n > 1. \end{cases}$$

4. U svaki element polja iz prve točke upišite slučajno izgeneriran broj u intervalu [0, 127] i sortirajte polje uzlazno. Nakon toga za svaki cjelobrojni dio elemenata polja odredite ekvivalentni znak u ASCII tablici.

5. Napravite algoritam koji će pronaći magični kvadrat 5 x 5 i ispisati ga na zaslon. Magični kvadrat n-tog reda je kvadratna shema n×n brojeva od 1 do n^2 u kojoj je zbroj u svakom stupcu, retku i dijagonali jednak i iznosi:

$$Sn = \frac{n(1+n^2)}{2}$$