Универзитет у Београду
Факултет организационих наука
Катедра за софтверско инжењерство
Лабораторија за софтверско инжењерство

Програмирање 1
Збирка задатака

Аутори:

Саша Лазаревић

Илија Антовић

Душан Савић

Милош Милић

Војислав Станојевић

Татјана Стојановић

Београд, 2019.

Садржај

1. Изрази, оператори, типови података, алгоритамске структуре

У овом одељку приказани су изрази, оператори, типови података и алгоритамске структуре. Дати су решени задаци и задаци за вежбање.

1.1. Решени задаци

1. Написати програм који на стандардном излазу у првом реду приказује поруку „Програмирање 1“, док у другом реду приказује име и презиме студента.

```
#include <stdio.h>
int main(void) {
	printf("\nProgramiranje\t1");
	printf("\nIme i prezime studenta\n");
	return 0;
}
```

2. Написати програм који рачуна и на стандардном излазу приказује површину круга полупречника 5.

```
#include <stdio.h>
#define PI 3.141592
int main(void) {
	double r = 5;
	double povrsina;
	povrsina = r * r * PI;
	printf("Povrsina kruga poluprecnika 5 je %lf\n", povrsina);
	return 0;
}
```

3. Написати програм који рачуна и на стандардном излазу приказује површину круга произвољног полупречника.

```
#include <stdio.h>
#define PI 3.141592
int main(void) {
	double r;
	double povrsina;
	printf("Unesite poluprecnik:\n");
	scanf("%lf", &r);
	povrsina = r * r * PI;
	printf("Povrsina kruga poluprecnika %lf je %lf\n", r, povrsina);
	return 0;
}
```

4. На стандардном улазу унети два броја *a* и *b* и заменити им вредности.

```
#include <stdio.h>
int main(void) {
	int a;
	int b;
	int c;
	printf("Unesite brojeve a i b \n");
	scanf("%d", &a);
	scanf("%d", &b);
	printf("a=%d, b=%d \t Pre smene\n", a, b);
	c = a;
	a = b;
	b = c;
	printf("a=%d, b=%d \t Posle smene\n", a, b);
	return 0;
}
```

5. Унети број *a*. Испитати парност броја *a*.

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

```
int main(void) {
	int a;
	printf("Unesite a: ");
	scanf("%d", &a);
	if (a > 0) {
		if (a % 2 == 0) {
			printf("Broj je paran\n");
		}
		else {
			printf("Broj je neparan\n");
		}
	}
	else {
		printf("Uneti broj je manji ili jednak nuli\n");
	}
	return 0;
}
```

6. Унети два броја *a* и *b*. Испитати релацију између бројева *a* и *b*.

```
#include <stdio.h>
int main(void) {
	int a;
	int b;
	printf("Unesite broj a: ");
	scanf("%d", &a);
	printf("Unesite broj b: ");
	scanf("%d", &b);
	if (a == b) {
		printf("Brojevi %d i %d su jednaki\n", a, b);
	}
	else {
		if (a > b) {
			printf("Broj %d je veci od broja %d\n", a, b);
		}
		else {
			printf("Broj %d je manji od broja %d\n", a, b);
		}
	}
	return 0;
}
```

7. Унети три броја *a*, *b* и *c*. Написати програм који проналази највећи број.

```
#include <stdio.h>
int main(void) {
	int a, b, c;
	int max;
	printf("Unesite broj a: ");
	scanf("%d", &a);
	printf("Unesite broj b: ");
	scanf("%d", &b);
	printf("Unesite broj c: ");
	scanf("%d", &c);
	max = a;
	if (b > max) {
		max = b;
	}
	if (c > max) {
		max = c;
	}
	printf("Max (%d, %d, %d) = %d\n", a, b, c, max);
```

```
	return 0;
}
```

8. Унети два броја *a* и *b*, и математичку операцију (+, -, * или /). Написати програм који рачуна вредност израза *a операција b*.

```
#include <stdio.h>
int main(void) {
	int a, b;
	char op;
	printf("Unesite a,b: ");
	scanf("%d,%d", &a, &b);
	getchar();
	printf("Unesite operaciju (+,-,* ili /): ");
	scanf("%c", &op);
	switch (op) {
	case '+':
		printf("Operacija: sabiranje\n");
		printf("Rezultat %d+%d=%d\n", a, b, a + b);
		break;
	case '-':
		printf("Operacija: oduzimanje\n");
		printf("Rezultat %d-%d=%d\n", a, b, a - b);
		break;
	case '*':
		printf("Operacija: mnozenje\n");
		printf("Rezultat %d*%d=%d\n", a, b, a * b);
		break;
	case '/':
		printf("Operacija: deljenje\n");
		printf("Rezultat %d/%d=%d\n", a, b, a / (double)b);
		break;
	default:
		printf("Niste uneli odgovarajucu operaciju!\n");
		break;
	}
	return 0;
}
```

9. Написати програм који рачуна суму једноцифрених бројева. Задатак решити коришћењем *for* наредбе.

```
#include <stdio.h>
int main(void) {
	int i;
	int suma;
	suma = 0;
	for (i = 1; i < 10; i++) {
		suma = suma + i;
	}
	printf("Suma jednocifrenih brojeva je: %d, a vrednost i je %d\n", suma, i);
	return 0;
}
```

10. Написати програм који рачуна суму једноцифрених бројева. Задатак решити коришћењем *while-do* наредбе.

```
#include <stdio.h>
int main(void) {
	int i;
	int suma;

	suma = 0;
	i = 1;
```

```
	while (i < 10) {
		suma = suma + i;
		i++;
	}
	printf("Suma jednocifrenih brojeva je: %d, a vrednost i je %d\n", suma, i);
	return 0;
}
```

11. Написати програм који рачуна суму једноцифрених бројева. Задатак решити коришћењем *do-while* наредбе.

```
#include <stdio.h>
int main(void) {
	int i;
	int suma;
	suma = 0;
	i = 1;
	do {
		suma = suma + i;
		i++;
	} while (i < 10);
	printf("Suma jednocifrenih brojeva je: %d, a vrednost i je %d\n", suma, i);
	return 0;
}
```

12. Унети два броја *a* и *n*. Написати програм који рачуна степен a^n.

```
#include <stdio.h>
int main(void) {
	int n;
	int a;
	int rez = 1;
	printf("Unesite a, n: ");
	scanf("%d,%d",&a, &n);
	printf("a = %d\n", a);
	printf("n = %d\n", n);
	for (int i = 1; i <= n; i++) {
		rez = a * rez;
	}
	printf("%d ^ %d = %d\n", a, n, rez);
	return 0;
}
```

13. Унети доњу и горњу границу интервала бројева [dg, gg]. Израчунати средњу вредност непарних бројева у задатом интервалу [dg, gg].

```
#include <stdio.h>
int main(void) {
	int dg;
	int gg;
	int s = 0;
	double sr;
	int brojac = 0;
	printf("\nUnesite interval a,b: ");
	scanf("%d,%d", &dg, &gg);
	for (int i = dg; i <= gg; i++) {
		if (i % 2 == 1) {
			s = s + i;
			brojac++;
		}
	}
	if (brojac > 0) {
		sr = s / brojac;
		printf("SrednjaVrednost je %.2lf\n", sr);
```

```
	}
	else {
		printf("U intervalu nema neparnih brojeva\n");
	}
	return 0;
}
```

14. Дефинисати промењиву *a* целобројног типа. Приказати њену вредност и адресу у меморији. Дефинисати промењиву *pint* показивачког типа на целобројну промењиву *a*. Приказати адресу на коју *pint* показује, вредност која се налази на тој адреси и адресу промењиве *pint.*

```
#include <stdio.h>
int main(void) {
	int a = 5;
	int * pint;
	printf("\n a=%d, &a=%p, &a=%X", a, &a, &a);
	pint = &a;
	printf("\n *pint=%d, pint=%p, &pint=%p", *pint, pint, &pint);
	return 0;
}
```

1.2. Задаци за вежбање

15. Унети границе интервала двоцифрених бројева [a, b]. Приказати све бројеве из интервала чији је збир цифара дељив са 3. Задатак решити коришћењем:
    а) *for* наредбе,
    б) *while-do* наредбе,
    в) *do-while* наредбе.
16. Унети границе интервала целих бројева [a, b]. У затвореном интервалу целих бројева наћи суму парних бројева. Задатак решити коришћењем:
    а) for наредбе,
    б) while-do наредбе,
    в) do-while наредбе.
17. Унети границе интервала целих бројева [a, b]. У затвореном интервалу целих бројева наћи колико има бројева који су дељиви са неким траженим бројем. Задатак решити коришћењем:
    а) for наредбе,
    б) while-do наредбе,
    в) do-while наредбе.
18. Омогућити кориснику унос *n* целих бројева све док унети цели бројеви представљају аритметички низ. Аритметички низ целих бројева је низ бројева код којег се сваки следећи члан добија из претходног додавањем једног истог броја *d*. Број *d* назива се разликом тог аритметичког низа.
19. Омогућити кориснику унос *n* целих бројева све док унети цели бројеви представљају геометријски низ. Геометријски низ целих бројева је низ бројева код којег се сваки члан низа, почевши од другог, добија из претходног множењем једним истим бројем *q* ($q \neq 0$). Број *q* је количник тог геометријског низа.

2. Потпрограми

У овом одељку приказани су потпрограми. Дати су решени задаци, задаци за вежбање, задаци са колоквијума и задаци са испитних рокова.

2.1. Решени задаци

20. Написати функцију која рачуна факторијел неког броја. Израчунати вредност израза *x! + (x+2)! - (2x)!*.

```
#include <stdio.h>
int faktorijel(int x) {
	int i;
	int f = 1;
	for (i = x; i > 1; i--) {
		f = f * i;
	}
	return f;
}
int main(void) {
	int x;
	double c;
	int izraz;
	printf("Unesite broj: ");
	scanf("%d", &x);
	izraz = faktorijel(x) + faktorijel(x + 2) - faktorijel(2 * x);
	printf("Vrednost izraza je %d\n", izraz);
	return 0;
}
```

21. Написати функцију која провера парност броја *a*.

```
#include <stdio.h>

enum Pripadnost_Broja { PARAN, NEPARAN, GRESKA };

enum Pripadnost_Broja provera_parnosti(int a) {
	if (a <= 0) {
		return GRESKA;
	}
	a % 2 == 0 ? PARAN : NEPARAN;
}
int main(void) {
	int a;
	enum Pripadnost_Broja pripadnost;
	printf("Unesite broj a: ");
	scanf("%d", &a);
	pripadnost = provera_parnosti(a);
	switch (pripadnost) {
	case PARAN:
		printf("Broj %d je paran\n", a);
		break;
	case NEPARAN:
		printf("Broj %d je neparan\n", a);
		break;
	case GRESKA:
		printf("Greska u unosu\n");
		break;
	}
}
```

22. Написати процедуру *zamena_mesta* која као параметре прихвата два броја и мења им вредности. У главном програму дефинисати два броја, приказати њихове вредности пре и након позива функције *zamena_mesta*.

```
#include <stdio.h>
void zamena_mesta(int* a, int* b) {
	int c;
	c = *b;
	*b = *a;
	*a = c;
	printf("\na=%d, b=%d", *a, *b);

}
int main(void) {
	int a = 5;
	int b = 3;
	printf("\na=%d, b=%d", a, b);
	zamena_mesta(&a, &b);
	printf("\na=%d, b=%d", a, b);
	return 0;
}
```

23. Приказати аргументе који су прослеђени *main* функцији.

```
#include <stdio.h>
int main(int argc, char* argv[]) {
	int i;
	printf("Broj argumenata je %d \n", argc);
	for (i = 0; i < argc; i++) {
		printf("argiment %d. je %s \n", i, argv[i]);
	}
}
```

24. Написати процедуру за унос два броја *a* и *n*. Написати функцију која рачуна степен a^n. У главном програму позвати претходно дефинисане потпрограме и израчунати вредности следећих израза: (1) a^n и (2) $a^n + a^{n+1}$.

```
#include <stdio.h>
int stepen_broja(int a, int n) {
	int rez = 1;
	for (int i = 1; i <= n; i++) {
		rez *= a;
	}
	return rez;
}
void unesi_stepen(int* a, int* n) {
	printf("\na = ");
	scanf("%d", a);
	printf("\nn = ");
	scanf("%d", n);
}
int main(void) {
	int x;
	int y;
	unesi_stepen(&x, &y);
	printf("\na = %d, n = %d", x, y);
	int r = stepen_broja(x, y);
	printf("\n%d^%d = %d", x, y, r);
	printf("\nIzraz: %d\n", stepen_broja(x, y) + stepen_broja(x, y + 1));
	return 0;
}
```

2.2. Задаци за вежбање

25. Написати потпрограм који рачуна да ли је унети број прост. Написати потпрограм који у неком интервалу бројева [a, b] рачуна колико има простих бројева.
26. Написати потпрограм који проверава да ли је цифра јединице неког броја 9. Написати потпрограм који у неком интервалу бројева [a, b] рачуна збир бројева чија је цифра јединице 9.
27. Написати процедуру која за унетих *n* унетих редова приказује фигуру као на слици (на слици је дат пример за *n* = 5).

```
    *
   * *
  * * *
 * * * *
* * * * *
```

Слика 1. Фигура за *n = 5*

28. Написати потпрограм који приказује све троцифрене бројеве код којих су цифре стотине, десетице и јединице узастопни бројеви (нпр. 456, 465, 546, 564, 645, 654 итд.).

2.3. Задаци са колоквијума

29. Дат је следећи програмски захтев:

а) Имплементирати непараметризовану процедуру **zadatak_1**. У оквиру процедуре са стандардног улаза прихватити два цела броја. Имплементирати потпрограм (процедуру или функцију) **kontrola_unosa** која проверава да ли су ова два цела броја лепо унета (оба броја морају бити већи од 0 и други број мора бити већи или једнак са првим бројем). Уколико корисник није лепо унео ова два броја, омогућити поновни унос. Поновни унос корисник може поновити максимално три пута. У случају да корисник лепо унесе ова два броја приказати поруку: *Корисник је унео интервал [a,b]*, где је *a* – први, а *b* – други број.Ако из три покушаја корисник не унесе лепо интервал, приказати поруку: *Корисник није унео интервал из три покушаја*.

б) Имплементирати потпрограм који у задатом интервалу налази средњу вредност непарних бројева. Ако је корисник лепо унео интервал на стандардном излазу у процедури **zadatak_1** приказати средњу вредност на 3 децимале.

в) Имплементирати потпрограм који у задатом интервалу налази колико има бројева чија је вредност већа од средње вредности непарних бројева. Ако је корисник лепо унео интервал на стандардном излазу у процедури **zadatak_1** приказати колико има бројева чија је вредност већа од средње вредности непарних бројева.

30. Дат је следећи програмски захтев:

а) Имплементирати непараметризовану процедуру **zadatak_1**. У оквиру процедуре са стандардног улаза прихватити два цела броја. Имплементирати потпрограм (процедуру или функцију) **kontrola_unosa** која проверава да ли су ова два цела броја лепо унета (оба броја морају бити троцифрени бројеви и други број мора бити већи од првог броја). Уколико корисник није лепо унео ова два броја, омогућити поновни унос. Поновни унос вршити све док корисник не унесе исправно ова два броја. Након успешног уноса приказати поруку: *Корисник је унео интервал троцифремих бројава [a,b],* где је *a* – први, а *b* – други број.

б) Имплементирати потпрограм који рачуна збир цифара неког броја.

в) Имплементирати потпрограм који у задатом интервалу налази све бројеве код којих је збир цифара дељив са 2. Искористити потпрограм из претходног дела задатка који рачуна збир цифара неког броја. У процедури **zadatak_1** приказати све бројеве код којих је збир цифара дељив са 2.

31. Имплементирати непараметризовану процедуру **zadatak_4**. У оквиру процедуре позвати процедуру meni која приказује кориснику конзолни кориснички интерфејс употребом менија.

```
========
STUDENT
========
0. Kraj rada
1. Ubaci     (Insert)
2. Izbaci    (Delete)
3. Promeni   (Update)
4. Prikazi   (Select)
4.0. Kraj (povratak u prethodni meni)
4.1. Prikazi sve  (1- opcija)
4.2. Prikazi po broju indeksa (BI) (2- opcija)
4.3. Prikazi po prezimenu (3- opcija)
```

32. Написати функцију која проверава да ли је број X савршен. Број је савршен ако је једнак суми својих делилаца, искључујући њега самог. На пример број 6=1+2+3, 28=1+2+4+7+14.
33. Написати потпрограм који исписује све бројеве од 1 до n за задати природан број n.
34. Написати функцију којом се проверава да ли су у броју N цифре сортиране у неопадајућем редоследу од цифре јединица ка цифрама веће тежине. На пример у броју N=7433 јесу, а у броју N=1322 нису.
35. Написати потпрограм који омогућава да корисник уноси бројеве све док не унесе нулу. Након уноса бројева одређује се највећи број од унетих бројева у коме су цифре сортиране у неопадајућем редоследу од цифре јединица ка цифрама веће тежине.

2.4. Задаци са испитних рокова

36. Имплементирати следеће потпрограме:
    а) Имплементирати функцију која рачуна збир цифара неког задатог броја.
    б) Написати главни програм који са стандардног улаза прихвата *n* бројева и приказује број чији је збир цифара највећи. Искористити претходно имплементирану функцију из задатака а).
    в) Написати функцију која рачуна колико има парних троцифрених бројева чији је збир цифара 13. Искористити претходно имплементирану функцију из задатака а). У главном програму позвати имплементирану функцију.
37. Имплементирати функцију која рачуна и враћа колико делилаца има задати број. У главном програму позвати имплементирану функцију, и исписати поруку у следећем формату: Број ___ има укупно ___ делилаца.
38. Имплементирати функцију која рачуна и враћа суму квадрата n природних бројева. У главном програму позвати имплементирану функцију, и исписати поруку у следећем формату: Сума квадрата за ___ природних бројева је ___.
    Тестни пример: Ако је n=4; функција треба да израчуна суму квадрата као: $1^2+2^2+3^2+4^2=30$.
39. Написати функцију која за унети арапски број приказује одговарајући римски број.
    Тестни пример:
    2736 = MMDCCXXXVI
40. Имплементирати следеће потпрограме:
    а) Написати потпрограм који за неки задати број испитује да ли је палиндром. Тест пример: 12321 (јесте), 2343 (није).
    б) Написати потпрограм и главни програм који са стандардног улаза прихвата n бројева (све док корисник не унесе 0) и проверава да ли су унесени бројеви палиндроми.
41. Близанци су два проста броја која се разликују за два (На пример: 3, 5 или 41, 43). Имплементирати функцију која у задатом отвореном интервалу целих бројева (a, b) приказује све парове близанаца у следећем формату:

```
[1.par 3,5]
```

```
[2.par 5,7]
....
{ukupno=2}
```

42. Имплементирати следеће потпрограме:
    а) Имплементирати функцију која на основу унете 4 цифре формира четвороцифрени број А на следећи начин:
    – Прва унета цифра представља цифру хиљаде, друга цифру стотине, трећа цифру десетице и четврта цифру јединице.
    – Цифре могу да се понављају.
    – Цифре морају бити у интервалу од 1 до 5. У случају да је корисник унео цифру која није у интервалу од 1 до 5 поновити унос. Максималан број покушаја за унос сваке цифре је 4. Уколико корисник не унесе исправно цифру из четири покушаја прекинути унос броја.
    б) Имплементирати функцију која прихвата два четвороцифрена броја (А и Б) и која проверава да ли се цифре броја Б поклапају по редоследу са цифрама броја А и приказује поруку у следећем формату: број погођених цифара које су на месту, број погођених цифара које нису на месту (слично игри „Скочко“ у „Слагалици“). Четвороцифрени бројеви А и Б садрже само цифре у интервалу од 1 до 5.
    Тестни пример 1:
    А = 1224
    Б = 2221
    Порука: 2 на месту, 1 није на месту
    Тестни пример 2:
    А = 1115
    Б = 2221
    Порука: 0 на месту, 1 није на месту
43. Палиндромски број је број који се исто чита са обе стране. Имплементирати функцију за унос бројева све док се не унесе нула и одредити највећи палиндромски број међу њима или приказати поруку да палиндромски број не постоји.
44. Палиндромски број је број који се исто чита са обе стране. Имплементирати функцију за унос бројева све док се не унесе нула и одредити најмањи палиндромски број међу њима или приказати поруку да палиндромски број не постоји.
45. Написати програм који за унети број налази њему најближи прост број. Ако је унети број прост, тражи се њему најближи прост број. Уколико постоје два најближа проста броја приказати оба броја.
    Тестни пример 1:
    N=15
    Најближи прости број: 17 и 13
    Тестни пример 2:
    N=19
    Најближи прости број: 17
46. Марко Марковић је студент 1. године ФОН-а и на свом рачуну у банци има X динара. Марко је имао договор са својим оцем да уколико положи Програмирање 1 у јунском испитном року, отац Марку почев од 1. септембра уплаћује на рачун у банци Y динара следећих D дана на следећи начин:
    – 1. септембра отац Марку треба да уплати Z динара.
    – Сваког наредног дана отац Марку треба да уплаћује V динара више него претходног дана.
    Имплементирати функцију која рачуна колико Марко има динара у банци након D дана и креира извештај на стандардном излазу у формату који је дат ниже.
    Формат извештаја на стандардном излазу за X = 100, Y = ..., V= 2, Z= 10, D=30 биће:

```
BANKOVNI IZVESTAJ
Marko Markovic, stanje = 100 dinara
```

```
1. dana: uplata = 10 dinara, saldo 110 dinara
2. dana: uplata = 12 dinara, saldo 122 dinara
3. dana: uplata = 14 dinara, saldo 136 dinara
...
30. dana: uplata = ... dinara, saldo ... dinara
Hvala tata.
```

3. Низови

У овом одељку приказани су низови. Дати су решени задаци, задаци за вежбање, задаци са колоквијума и задаци са испитних рокова.

3.1. Решени задаци

47. Дат је низ целих бројева. Написати процедуру за приказ елемената низа.

```
#include <stdio.h>
void prikazi_niz(int x[], int n) {
	printf("\n");
	for (int i = 0; i < n; i++) {
		printf("%d ", x[i]);
	}
}
int main(void) {
	int x[10] = { 8, 4, 2, 1, 6, 5, 3, 3 };
	prikazi_niz(x, 8);
	return 0;
}
```

48. Дат је низ целих бројева. Написати процедуру за приказ елемената низа коришћењем показивача.

```
#include <stdio.h>
void prikazi_niz(int *x, int n) {
	printf("\n");
	for (int i = 0; i < n; i++) {
		printf("%d ", *(x + i));
	}
}
int main(void) {
	int x[10] = { 8, 4, 2, 1, 6, 5, 3, 3 };
	prikazi_niz(x, 8);
	return 0;
}
```

49. Дат је низ целих бројева. Написати функцију која рачуна суму елемената низа.

```
#include <stdio.h>
int suma_niza_f(int niz[], int n) {
	int s = 0;
	for (int i = 0; i < n; i++) {
		s = s + niz[i];
	}
	return s;
}
int main(void) {
	int x[10] = { 8, 4, 2, 1, 6, 5, 3, 3 };
	int sx = suma_niza_f(x, 8);
	printf("\nSuma elemenata niza x=%d", sx);
	printf("\nSuma elemenata niza x=%d", suma_niza_f(x, 8));
	return 0;
}
```

50. Дат је низ целих бројева. Написати процедуру која рачуна суму елемената низа.

```
#include <stdio.h>
void suma_niza_p(int niz[], int n, int *suma) {
	int s = 0;
	for (int i = 0; i < n; i++) {
		s = s + niz[i];
	}
```

```
	*suma = s;
}
int main(void) {
	int x[10] = { 8, 4, 2, 1, 6, 5, 3, 3 };
	int sumax = 0;
	suma_niza_p(x, 8, &sumax);
	printf("\nSuma elemenata niza (procedura) x = %d", sumax);
	return 0;
}
```

51. Дат је низ целих бројева. Написати функцију проверава да ли одређени број постоји у низу.

```
#include <stdio.h>
int postoji_vrati_poziciju(int niz[], int n, int broj) {
	for (int i = 0; i < n; i++) {
		if (niz[i] == broj) {
			return i;
		}
	}
	return -1;
}
int main(void) {
	int broj;
	int x[] = { 8, 4, 2, 1, 6, 5, 3, 3 };
	int n = sizeof(x) / sizeof(int);
	printf("\nUnesite broj koji trazite: ");
	scanf("%d", &broj);
	int poz = postoji_vrati_poziciju(x, n, broj);
	if (poz == -1) {
		printf("\nU nizu x broj %d ne postoji.", broj);
	}
	else {
		printf("\nU nizu x broj %d postoji na poziciji %d", broj, poz + 1);
	}
	return 0;
}
```

52. Дат је низ целих бројева. Написати функцију која рачуна средњу вредност непарних елемената у низу.

```
#include <stdio.h>
double srv_neparnih(int x[], int n) {
	int suma = 0;
	int brEl = 0;
	for (int i = 0; i < n; i++) {
		if (x[i] % 2 == 1) {
			suma += x[i];
			brEl++;
		}
	}
	if (brEl > 0) {
		return (double)suma / brEl;
	}
	return 0;
}
int main(void) {
	double srv;
	int x[] = { 1, 8, 3, 13, 15, 8, 6, 4, 7 };
	int n = sizeof(x) / sizeof(int);
	srv = srv_neparnih(x, n);
	printf("\nSrednja vrednost neparnih elemenata je %.2lf", srv);
	return 0;
```

```
}
```

53. Дат је низ целих бројева. Написати процедуру која рачуна средњу вредност непарних елемената у низу.

```
#include <stdio.h>
void srv_neparnih(int x[], int n, double* srv) {
	int suma = 0;
	int brEl = 0;
	for (int i = 0; i < n; i++) {
		if (x[i] % 2 == 1) {
			suma += x[i];
			brEl++;
		}
	}
	if (brEl > 0) {
		*srv = (double)suma / brEl;
	}
	else {
		*srv = 0;
	}
}
int main(void) {
	double srv;
	int x[] = { 1, 8, 3, 13, 15, 8, 6, 4, 7 };
	int n = sizeof(x) / sizeof(int);
	srv_neparnih(x, n, &srv);
	printf("\nSrednja vrednost neparnih elemenata je %.2lf", srv);
	return 0;
}
```

54. Дат је низ целих бројева *x* у којем се вредности могу понављати. Написати процедуру за додавање новог елемента на крај низа. Написати процедуру која ће формирати нови низ *y* од елемената низа *x* при чему се вредности у низу *y* не смеју понављати. Приказати садржај низова *x* и *y*.

```
#include <stdio.h>
void ubaci_na_kraj_niza(int x[], int *n, int element) {
	x[*n] = element;
	*n = *n + 1;
}
void prikazi_niz(int niz[], int n) {
	for (int i = 0; i < n; i++) {
		printf("%d\t", niz[i]);
	}
}
int postoji(int x[], int n, int broj) {
	for (int i = 0; i < n; i++) {
		if (x[i] == broj) {
			return i;
		}
	}
	return -1;
}
void formiraj_niz_bez_ponavljanja(int x[], int n, int y[], int *m) {
	*m = 0; //Broj elemenata niza y je na pocetku 0
	for (int i = 0; i < n; i++) {
		//Da li se tekuci element niza x nalazi u nizu y
		int indeks = postoji(y, *m, x[i]); //Vrati -1 ako ne postoji ili indeks na
kom se nalazi trazeni broj
		if (indeks == -1) {
			ubaci_na_kraj_niza(y, m, x[i]);
```

```
			}
		}
	}
	int main(void) {
		int x[100] = { 1, 8, 3, 13, 4, 15, 8, 6, 4, 7 };
		int n;
		int y[100];
		int m;
		n = 10;
		printf("\nX: ");
		prikazi_niz(x, n);
		printf("\n");
		formiraj_niz_bez_ponavljanja(x, n, y, &m);
		printf("\nY: ");
		prikazi_niz(y, m);
		printf("\n");
		return 0;
	}
```

55. Написати процедуру која прихвата *n* целих бројева са стандардног улаза (тастатуре) и смешта их у низ. Написати процедуру за приказ елемената низа. За приступ елементима низа користити низовну промењиву.

```
#include <stdio.h>
void unos_niza(int y[], int m) {
	int i;
	for (i = 0; i < m; i++) {
		printf("y[%d]=", i);
		scanf("%d", &y[i]);
	}
}
void prikaz_niza(int y[], int m) {
	int i;
	for (i = 0; i < m; i++) {
		printf("%d\t", y[i]);
	}
	printf("\n");
}
int main(void) {
	int n;
	int x[10];
	printf("Koliko brojeva zelite da unesete (1-10)? ");
	scanf("%d", &n);
	unos_niza(x, n);
	prikaz_niza(x, n);
	return 0;
}
```

56. Написати процедуру која прихвата *n* целих бројева са стандардног улаза (тастатуре) и смешта их у низ. Написати процедуру за приказ елемената низа. За приступ елементима низа користити промењиву која представља показивач на низ.

```
#include <stdio.h>
void unos_niza(int *y, int m) {
	int i = 0;
	int broj;
	for (i = 0; i < m; i++) {
		printf("y[%d]=", i);
		scanf("%d", (y + i));
	}
}
void prikaz_niza(int *y, int m) {
```

```
	int i;
	for (i = 0; i < m; i++) {
		printf("%d\t", *y);
		y++;
	}
	printf("\n");
}
int main(void) {
	int n;
	int x[10];
	printf("Koliko brojeva zelite da unesete (1-10)? ");
	scanf("%d", &n);
	unos_niza(x, n);
	prikaz_niza(x, n);
	return 0;
}
```

57. Написати процедуру која омогућава кориснику да са стандардног улаза унесе произвољан број бројева, а само парне бројеве складишти у низу. Написати процедуру за приказ елемената низа.

```
#include <stdio.h>
void dodaj_element(int niz[], int *brEl, int broj) {
	niz[*brEl] = broj;
	*brEl = *brEl + 1;
}
void unos_niza(int x[], int *n) {
	int i = 0;
	int broj;
	int unos;
	do{
		printf("Unesite ceo broj: ");
		scanf("%d", &broj);
		if (broj % 2 == 0) {
			dodaj_element(x, n, broj);
		}
		printf("\Da li zelite da unesete novi broj (0-NE, 1-DA): ");
		scanf("%d", &unos);
	} while (unos != 0);
}
void prikaz_niza(int x[], int n) {
	int i;
	for (i = 0; i < n; i++) {
		printf("%d\t", x[i]);
	}
	printf("\n");
}
int main(void) {
	int n;
	int x[10000];
	n = 0;
	unos_niza(x, &n);
	printf("\n--- ELEMENTI NIZA ---\n");
	prikaz_niza(x, n);
	return 0;
}
```

58. Написати процедуру за додавање новог елемента на почетак низа. Написати процедуру за унос *n* елемената у низ (обавезно искористити претходну процедуру за додавање новог елемента на почетак низа). Приказати садржај низа.

```
#include <stdio.h>
void ubaci_na_pocetak_niza(int x[], int *n, int broj) {
```

```
	*n = *n + 1;
	//Pomeri sve elemente za jedno mesto udesno
	for (int i = *n - 1; i > 0; i--) {
		x[i] = x[i - 1];
	}
	x[0] = broj;
}
void napuni_niz(int niz[], int *n) {
	int broj;
	int br_el;
	printf("\nUnesite koliko zelite novih elemenata u nizu: ");
	scanf("%d", &br_el);
	for (int i = 1; i <= br_el; i++) {
		printf("\nUnesite %d. element niza: ", i);
		//Ubaci element na pocetak niza
		scanf("%d", &broj);
		ubaci_na_pocetak_niza(niz, n, broj);
	}
}
void prikazi_niz(int niz[], int n) {
	for (int i = 0; i < n; i++) {
		printf("%d\t", niz[i]);
	}
}
int main(void) {
	int x[100];
	int n;
	n = 0;
	napuni_niz(x, &n);
	printf("\nX: ");
	prikazi_niz(x, n);
	printf("\n");
	return 0;
}
```

3.2. Задаци за вежбање

59. Дат је низ целих бројева. Имплементирати потпрограм који у задатом низу целих бројева налази најмањи парни елемент низа.
60. Дат је низ целих бројева. Имплементирати потпрограм који у задатом низу целих бројева налази највећи парни елемент низа.
61. Дат је низ целих бројева. Имплементирати потпрограм који у задатом низу целих бројева налази најмањи непарни елемент низа.
62. Дат је низ целих бројева. Имплементирати потпрограм који у задатом низу целих бројева налази највећи непарни елемент низа.
63. Дат је низ целих бројева. Имплементирати потпрограм који у задатом низу целих бројева рачуна колико има елемената низа чија је вредност већа од средње вредности свих елемената низа.
64. Дат је низ целих бројева. Имплементирати потпрограм који у задатом низу целих бројева помера све елементе низа за једно место удесно.
65. Дат је низ целих бројева. Имплементирати потпрограм који у задатом низу целих бројева помера све елементе низа за једно место улево.
66. Дат је низ целих бројева. Имплементирати потпрограм који у задатом низу целих бројева помера све елементе низа за *n* места удесно.

67. Дат је низ целих бројева. Имплементирати потпрограм који у задатом низу целих бројева помера све елементе низа за *n* места улево.
68. Дат је низ целих бројева. Имплементирати потпрограм који у задатом низу целих бројева убацује нови елемент на задату позицију.
69. Дат је низ целих бројева. Имплементирати потпрограм за унос елемента низа тако да сви елементи низа буду међусобно различити. Приказати елементе низа.
70. Дат је низ целих бројева. Имплементирати потпрограм који избацује први елемент низа.
71. Дат је низ целих бројева. Имплементирати потпрограм који избацује последњи елемент низа.
72. Дат је низ целих бројева. Имплементирати потпрограм који избацује елемент низа са задате позиције.
73. Дат је низ целих бројева. Имплементирати потпрограм који избацује све елементе низа који имају задату вредност.
74. Дат је низ целих бројева. Имплементирати функцију која проверава да ли у низу целих бројева има дупликата (бројева који се појављују више пута у низу).
75. Дат је низ целих бројева у којем се елементи могу понављати. Имплементирати потпрограм која из низа избацује све елементе који су дупликати.
76. Дат је низ целих бројева. Имплементирати следеће потпрограме:
    а) Имплементирати функцију која рачуна фреквенцију појављивања неког броја у низу.
    б) Имплементирати функцију која приказује елементе низа који се у низу појављују тачно 2 пута. Сваки елемент приказати само једанпут.
77. Дат је низ целих бројева. Имплементирати следеће потпрограме:
    а) Имплементирати функцију која рачуна аритметичку средину елемената низа.
    б) Имплементирати функцију која налази елемент низа чија је вредност најближа аритметичкој средини елемената низа.
78. Дат је низ целих бројева. Имплементирати следеће потпрограме:
    а) Имплементирати функцију која проверава да ли је неки број прост.
    б) Имплементирати процедуру која приказује све елементе низа који су прости бројеви.
79. Дат је низ целих бројева. Имплементирати следеће потпрограме:
    а) Имплементирати функцију која рачуна збир цифара неког задатог броја.
    б) Имплементирати процедуру која приказује све елементе низа чији је збир цифара дељив са неким задатим бројем.
80. Дат је низ целих бројева. Имплементирати потпрограм који од парних елемената улазног низа *x* формира нови низ *y*. Приказати елементе низова *x* и *y*.
81. Дат је низ целих бројева. Имплементирати потпрограм који приказује све елементе низа у обрнутом редоследу.
82. Дат је низ целих бројева. Имплементирати потпрограм који приказује све елементе на парним позицијама у низу.
83. Дат је низ целих бројева. Имплементирати функцију која проверава да ли је низ целих бројева растући.
84. Дат је низ целих бројева. Имплементирати функцију која проверава да ли је низ целих бројева неопадајући.
85. Дат је низ целих бројева. Имплементирати функцију која проверава да ли је низ целих бројева опадајући.
86. Дат је низ целих бројева. Имплементирати функцију која проверава да ли је низ целих бројева нерастући.
87. Дат је низ целих бројева. Имплементирати функцију која проверава да ли је дати низ аритметички низ. Аритметички низ целих бројева је низ бројева код којег се сваки следећи члан добија из претходног додавањем једног истог броја *d*. Број *d* назива се разликом тог аритметичког низа.

88. Дат је низ целих бројева. Имплементирати функцију која проверава да ли је дати низ геометријски низ. Геометријски низ целих бројева је низ бројева код којег се сваки члан низа, почевши од другог, добија из претходног множењем једним истим бројем q ($q \neq 0$). Број q је количник тог геометријског низа.
89. Дат је низ целих бројева. Имплементирати функцију која у улазном низу А налази најдужи подниз парних бројева.
90. Дат је низ целих бројева. функцију која приказује најдужи подниз целих бројева кога чине исти бројеви.
91. Дат је низ целих бројева. Имплементирати функцију која све елементе на парним позицијама увећава за 1, а на непарним позицијама смањује за 2.
92. Дата су два низа целих бројева *A* и *B*. Имплементирати потпрограм који за два улазна низа *A* и *B* формира нови низ *C* као унију ова два низа. Низ *C* чине сви елементи низа *A* и сви елементи низа *B*. У низу *C* сви елементи морају бити међусобно различити.
93. Дата су два неопадајућа низа, низ А димензије 5 и низ Б димензије 8. Формирати нови низ Ц уређен у неопадајућем редоследу од елемената из низа А и низа Б.
    Тестни пример:

А:	2	3	3	6	7							
Б:	0	1	1	3	4	4	9	10				
Ц:	0	1	1	2	3	3	3	4	4	6	7	9	10

94. Дата су два низа целих бројева *A* и *B*. Имплементирати потпрограм који за два улазна низа *A* и *B* формира нови низ *C* као разлику ова два низа. Низ *C* чине сви елементи који се налазе у низу *A* а не налазе се у низу *B*. У низу *C* сви елементи морају бити међусобно различити.
95. Дата су два низа целих бројева *A* и *B*. Имплементирати потпрограм који за два улазна низа *A* и *B* формира нови низ *C* као пресек ова два низа. Низ *C* чине само они елементи који се налазе и у *A* и у низу *B*. У низу *C* сви елементи морају бити међусобно различити.

3.3. Задаци са колоквијума

96. Дат је низ целих бројева. Имплементирати следеће потпрограме:
    а) Имплементирати потпрограм **ubaci_na_pocetak** који убацује нови елемент на почетак низа.
    б) Имплементирати потпрограм **postoji_u_nizu** који проверава да ли неки број постоји у низу.
    в) Имплементирати потпрограм **pun_niz** која проверава да ли је низ пун.
    г) Имплементирати процедуру **suma** која рачуна суму елемената низа.
    д) Имплементирати непараметризовану процедуру **zadatak_2**. У оквиру ове процедуре прихватити n бројева и убацити их и низ. Приликом убацивања елемената у низ проверити да ли је низ пун и да ли у низу већ не постоји такав број. Уколико су ова два услова испуњена убацити број на почетак низа. Уколико је низ пун приказати поруку: *Niz je pun.*, а ако број постоји у низу приказати поруку: *Broj (koji broj) ce vec nalazi u nizu*. У процедури **zadatak_2** позвати процедуру за приказ елемената низа. Процедура **zadatak_2** мора бити имплементирана. Ово је предуслов да би се бодовале претходно имплементиране процедуре и функције.
    ђ) У процедури **zadatak_2** позвати процедуру **suma** (која рачуна суму свих елеменат низа) и приказати колика је сума парних елемената унетог низа (није дозвољено мењати функцију **suma**, али је дозвољено додати по потреби нове процедуре и функције).
97. Дат је низ целих бројева. Имплементирати следеће потпрограме:
    а) Имплементирати потпрограм **ubaci_na_poziciju** који убацује нови елемент на задату позицију у низу.
    б) Имплементирати потпрограм **postoji_u_nizu** који проверава да ли неки број постоји у низу.

в) Имплементирати потпрограм **proveri_poziciju** која проверава да ли нови број може да се убаци на задату позицију у низу (позиција може бити у интервалу од 0 до броја елемената у низу).

г) Имплементирати процедуру **suma** која рачуна суму елемената низа.

д) Имплементирати непараметризовану процедуру **zadatak_2**. У оквиру ове процедуре прихватити n бројева и убацити их у низ. Приликом убацивања елемената у низ унети број и позицију на коју се убацује нови број. Проверити да ли је позиција одговарајућа и да ли у низу већ не постоји такав број. Уколико су ова два услова испуњена убацити број на задату позицију у низу. У процедури **zadatak_2** позвати процедуру за приказ елемената низа. Процедура **zadatak_2** мора бити имплементирана. Ово је предуслов да би се бодовале претходно имплементиране процедуре и функције.

ђ) У процедури **zadatak_2** позвати процедуру **suma** (која рачуна суму свих елеменат низа) и приказати колика је сума парних елемената унетог низа (није дозвољено мењати функцију **suma**, али је дозвољено додати по потреби нове процедуре и функције).

98. Дат је низ целих бројева. Имплементирати следеће потпрограме:

а) Имплементирати потпрограм **izbaci_sa_pocetka** која избацује први елемент из низа.

б) Имплементирати потпрограм **postoji_parni_broj** која проверава да ли у низу постоји парни број.

в) Имплементирати потпрограм **min_parni_broj** који налази најмањи парни број у низу.

г) Имплементирати непараметризовану процедуру **zadatak_2**. У оквиру процедуре **zadatak_2** дат је низ целих бројева. У оквиру ове процедуре позвати процедуру **meni** која омогућава кориснику да позове претходно имплементирате потпрограме из задатка. Опцијом 1 брише се први елемент из низа, опцијом 2 проверава се да ли у низу постоји парни број, опцијом 3 приказује се најмањи парни елемент низа. Опцијом 0 излази се из процедуре **meni**. Процедура **zadatak_2** мора бити имплементирана. Ово је предуслов да би се бодовале претходно имплементиране процедуре и функције.

99. Дат је низ целих бројева. Имплементирати следеће потпрограме:

а) Имплементирати потпрограм **izbaci_sa_pocetka** која избацује први елемент из низа.

б) Имплементирати потпрограм **postoji_neparni_broj** која проверава да ли у низу постоји непарни број.

в) Имплементирати потпрограм **min_neparni_broj** који налази најмањи непарни број у низу.

г) Имплементирати непараметризовану процедуру **zadatak_2**. У оквиру процедуре **zadatak_2** дат је низ целих бројева. У оквиру ове процедуре позвати процедуру **meni** која омогућава кориснику да позове претходно имплементирате потпрограме из задатка. Опцијом 1 брише се први елемент из низа, опцијом 2 проверава се да ли у низу постоји непарни број, опцијом 3 приказује се најмањи непарни елемент низа. Опцијом 0 излази се из процедуре **meni.** Процедура **zadatak_2** мора бити имплементирана. Ово је предуслов да би се бодовале претходно имплементиране процедуре и функције.

100. Дат је низ целих бројева. Имплементирати следеће потпрограме:

а) Имплементирати потпрограм **izbaci_sa_pocetka** која избацује први елемент из низа.

б) Имплементирати потпрограм **postoji_parni_broj** која проверава да ли у низу постоји парни број.

в) Имплементирати потпрограм **min_parni_broj** који налази најмањи парни број у низу.

г) Имплементирати непараметризовану процедуру **zadatak_2**. У оквиру процедуре **zadatak_2** дат је низ целих бројева. У оквиру ове процедуре позвати процедуру **meni** која омогућава кориснику да позове претходно имплементирате потпрограме из задатка. Опцијом 1 брише се први елемент из низа, опцијом 2 проверава се да ли у низу постоји парни број, опцијом 3 приказује се најмањи парни елемент низа. Опцијом 0 излази се из процедуре **meni**. Процедура **zadatak_2** мора бити имплементирана. Ово је предуслов да би се бодовале претходно имплементиране процедуре и функције.

101. Дат је низ целих бројева А (димензија низа је 10) у коме елементи могу да се понављају. Дат је низ Б (димензија низа је 7) у коме су сви елементи међусобно различити. Написати функцију која на стандардном излазу приказује колико пута се сваки елемент низа Б појављује у низу А. Уколико се број из низа Б не налази у низу А приказати поруку у формат: „Број (који број) из низа Б се не појављује у низу А“.

А	6	0	8	7	8	2	8	**0**	**9**	**3**
Б	6	0	8	7	2	9	3			

Б(6) >> А(1)
Б(0) >> А(2)
Б(8) >> А(3)
Б(7) >> А(1)
Б(2) >> А(1)
Б(9) >> А(1)
Б(3) >> А(1)

102. Дата су два неопадајућа низа, низ А димензије 5 и низ Б димензије 8.Формирати нови низ Ц уређен у неопадајућем редоследу од елемената из низа А и низа Б. Није дозвољено користити функцију за сортирање низа.
Тестни пример:

А:	2	3	3	6	7								
Б:	0	1	1	3	4	4	9	10					
Ц:	0	1	1	2	3	3	3	4	4	6	7	9	10

103. Дат је низ карактера. Написати функцију која приказује најдужи подниз карактера кога чине исти карактери.
Тестни пример:
Улаз: (,=,+,*,(,*,*,*,*,2,2,a,a,a
Излаз: *,*,*,*

104. Дат је низ целих бројева А (димензија низа је 10) у коме елементи могу да се понављају. Сортирати низ целих бројева А у неопадајућем редоследу.

105. Дат је низ целих бројева А (димензија низа је 10) у коме елементи могу да се понављају. Испитати да ли је низ целих бројева А неопадајући. Низ је неопадајући ако је сваки елемент у низу већи или једнак свом претходнику.
Тестни пример:

А:	0	1	2	3	3	4	7	8	8	9

неопадајући: ДА

А:	0	1	2	6	3	4	7	8	8	9

неопадајући: НЕ

106. Написати функцију која омогућава кориснику унос n елемената у низ Х целих бројева максималне димензије 20, али тако да сви елементи низа буду међусобно различити. Приказати из колико покушаја је корисник унео n елемената у низ Х.
Тестни пример:
n:7
Улазни бројеви:3,3,2,0,1,4,2,6,5,3

Х:	3	4	0	1	2	6	5

Број покушаја: 10

107. Написати функцију која омогућава кориснику унос n елемената у низ Х целих бројева максималне димензије 20, али тако да се један елемент у низу може појавити максимално два пута. Тестни пример:

Улазни бројеви: 5,5,5,1,0,1,3,5,1

X:	5	5	1	0	1	3

108. Написати функцију која проверава да ли је улазни низ целих бројева палиндромски. Низ је палиндромски уколико су бројеви у низу исти ако се читају и са лева и са десна.
Тестни пример:

A:	1	3	4	3	1

палиндромски:ДА

A:	1	2	4	3	1

палиндромски:НЕ

109. Написати следеће потпрограме:
а)Функцију која трансформише природни број n тако што му се изостављају њихово прво појављивање гледано с десна на лево. На пример, број 12144 се трансформише у број 214.
б)Процедуру која бројеве од 1 до n трансформише и исписује.

110. Написати функцију која на основу улазног низа X формира два низа, низ парних и низ непарних бројева.
Тестни пример:

X:	1	2	4	3	1	4	5

P:	2	4	4	
N:	1	3	1	5

111. Написати функцију која на основу улазног низа X формира два низа. Први низ чине они елементи из низа X који се у низу X појављују само једанпут, док други низ чине они елементи из низа X који се у низу X појављују два или више пута**.**
Тестни пример:

X:	1	2	4	3	1	4	8	5	8

N1:	2	5	3
N2:	1	4	8

3.4. Задаци са испитних рокова

112. Дат је низ целих бројева. Написати потпрограм који на основу задатог улазног низа проналази и приказује најдужи подниз простих бројева.
113. Дат је низ целих бројева (сви елементи низа целих бројева су бројеви већи од 0). Имплементирати следеће потпрограме:
а) Функцију која у задатом низу целих бројева проналази позицију најмањег парног броја у низу. У случају да у низу нема парних бројева функција враћа вредност -1.
б) У главном програму позвати дефинисану функцију и приказати кориснику на екрану одговарајућу поруку у ниже наведеном формату.
Формат приказа поруке уколико у низу постоји бар један паран број је:
`Најмањи парни ел.низа је: ___, а налази се на позицији ___.`
Формат приказа уколико у низу нема парних бројева је:
`У низу нема парних бројева.`
114. Дат је низ целих бројева (сви елементи низа целих бројева су бројеви већи од 0). Имплементирати следеће потпрограме:

а) Функцију која у задатом низу целих бројева проналази позицију најмањег непарног броја у низу. У случају да у низу нема непарних бројева функција враћа вредност -1.

б) У главном програму позвати дефинисану функцију и приказати кориснику на екрану одговарајућу поруку у ниже наведеном формату.

Формат приказа поруке уколико у низу постоји бар један непаран број је:

```
Најмањи непарни ел.низа је: ___, а налази се на позицији ___.
```

Формат приказа уколико у низу нема непарних бројева је:

```
У низу нема непарних бројева.
```

115. Дат је низ целих бројева (сви елементи низа целих бројева су бројеви већи од 0). Имплементирати следеће потпрограме:

а) Функцију која у задатом низу целих бројева рачуна за колико је збир првих *к* бројева низа, већи или мањи од збира последњих *м* бројева низа.

б) У главном програму позвати дефинисану функцију и приказати кориснику на екрану одговарајућу поруку у ниже наведеном формату.

```
Збир првих ___ бројева је: ___, а последњих ___ бројева је ___.
Збир првих ___ бројева је за ___ већи/мањи од збира последих ___ бројева.
```

116. Дат је низ целих бројева. Написати функцију која рачуна средњу вредност елемената низа и има следећи потпис:

```
void srednjaVrednost(int niz[], int brojElemenataNiza, double *srv);
```

117. Дат је низ целих бројева. Написати функцију која враћа број елемената низа који су већи од задате вредности и има следећи потпис:

```
int brojElVeciOdZadateVrednosti(int *p, int n, double zadatiBroj);
```

118. Дата су два низа природних бројева A и B, чије су димензије n односно m. Написати функцију и главни програм тако да на стандардном излазу буду приказани елементи из низа A који пребацивањем у низ B повећавају просечне вредности оба низа.

Тестни пример:

A = [12, 10, 14, 13], n = 4, SRV = 12.25

B =[2, 4, 4, 6, 6], m = 5, SRV = 4.4

Корак 1: Ако из низа A број 12 пређе у низ B тада је:

A = [10, 14, 13], SRV A је= 12.33, што је веће од 12.25

B =[2, 4, 4, 6, 6, 12], SRV B је= 5,67, што је веће од 4.4

У овом случају треба приказати број 12.

Поновити поступак за све елементе из низа A.

119. Компанија Тојота Јапан има развијен систем производње (енгл. *Toyota Production System*) у циљу оптимизације производње и елиминисања губитака. У том смислу је потребно направити програм који ће извршити оцену учинка запосленог на производној линији. **Укупан учинак запосленог се рачуна као сума броја бодова остварених за производњу ауто-дела (нпр. шасије, кочионог система, електро-инсталација итд.) и просечног броја бодова остварених за квалитет израде**. Бодовање се врши према следећем критеријуму:

– Бодови за производњу ауто-дела се обрачунавају на следећи начин: као гранично време за производњу изабрано је 20 мин за које запослени добија 120 бодова. За време које је испод 20 мин, на сваких пола минута испод границе од 20 минута запослени добија додатних 1.4 бода (*увећање за учинак*). За време које је изнад 20 мин, на сваки 0.5 минута изнад границе од 20 минута запосленом се одузима 1.2 бода (*умањење за учинак*).

– Бодове за квалитет даје 7 инжењера за управљање квалитетом, оценама од 10 до 20 (оцене могу бити бројеви са две децимале нпр. 11.34 или 15.65). Број бодова за квалитет се добија када се најмања и највећа оцена одбаце, а на основу осталих оцена израчуна аритметичка средина.

Имплементирати следеће потпрограме:

а) Имплементирати функцију рачуна број бодова које је остварио запослени на производној линији за произведени ауто-део чије је време (м минута) и чији су квалитет инжењери оценили оценама од 10 до 20.
б) Имплементирати процедуру која на стандардном излазу приказује извештај о учинку запосленог у следећем формату.

```
Toyota Japan
Zaposleni: <ime i prezime zaposlenog>
Vreme izrade: ______ min, broj bodova: =  ______
Ocene za kvalitet: {10.5; 12.5; 15.5; 12.25; 17.5; 15.5; 15.5}, prosecno bodova =
______
Ukupno bodova: ______
```

Тестни пример:
Време израде: 21.4 мин, број бодова = 117.6
Оцене за квалитет: {**10.5**; 12.5; 15.5; 12.25; **17.5**; 15.5; 15.5}, просечно бодова = 14.25
Укупно бодова: 117.6 + 14.25 = 131.85

4. Матрице

У овом одељку приказане су матрице. Дати су решени задаци, задаци за вежбање, задаци са колоквијума и задаци са испитних рокова.

4.1. Решени задаци

120. Дата је матрица целих бројева. Написати процедуру која приказује елементе матрице.

```
#include <stdio.h>
void prikazi_matricu(int n, int m, int mat[][10]) {
	for (int i = 0; i < n; i++) {
		for (int j = 0; j < m; j++) {
			printf("%5d", mat[i][j]);
		}
		printf("\n");
	}
}
int main(void) {
	int mat[][4] = {
		{ 1, 2, 3, 4 },
		{ 11, 22, 33, 44 }
	};
	int n = 2; //Broj redova matrice
	int m = 4; //Broj kolona matrice
	prikazi_matricu(n, m, mat);
	return 0;
}
```

121. Дата је матрица целих бројева. Написати процедуру која приказује елементе матрице коришћењем показивача.

```
#include <stdio.h>
void prikazi_matricu_v1(int n, int m, int *p) {
	int i;
	int total = n * m;
	for (i = 0; i < total; i++) {
		printf("%5d", *(p + i));
	}
	printf("\n");
}
void prikazi_matricu_v2(int n, int m, int *p) {
	int i, j;
	for (i = 0; i < n; i++) {
		for (j = 0; j < m; j++) {
			printf("%5d", *(p + i * m + j));
		}
		printf("\n");
	}
	printf("\n");
}
int main(void) {
	int mat[][4] = {
		{ 1, 2, 3, 4 },
		{ 11, 22, 33, 44 }
	};
	int n = 2; //Broj redova matrice
	int m = 4; //Broj kolona matrice
	prikazi_matricu_v1(n, m, mat);
	prikazi_matricu_v2(n, m, mat);
	return 0;
```

```
}
```

122. Дата је матрица целих бројева. Написати функцију која рачуна суму елемената задатог реда матрице.

```
#include <stdio.h>
int suma_reda(int zadati_red, int m, int mat[][10]) {
	int suma = 0;
	for (int j = 0; j < m; j++) {
		suma = suma + mat[zadati_red][j];
	}
	return suma;
}
int main(void) {
	int mat[10][10] = {
		{ 1, 2, 3, 4 },
		{ 11, 22, 33, 44 }
	};
	int zadati_red;
	int suma;
	int n = 2; //Broj redova matrice
	int m = 4; //Broj kolona matrice
	printf("Unesite zadati red: ");
	scanf("%d", &zadati_red);
	suma = suma_reda(zadati_red - 1, m, mat);
	printf("Suma elemenata %d. reda je %d\n", zadati_red, suma);
	return 0;
}
```

123. Дата је матрица целих бројева. Написати процедуру која приказује све елементе матрице и суму елемената сваког реда (искористити функцију која рачуна суму елемената задатог реда матрице).

```
#include <stdio.h>
typedef int TMATRICA[10][10]; //Definicija tipa
int suma_reda(int zadati_red, int m, TMATRICA mat) {
	int suma = 0;
	for (int j = 0; j < m; j++) {
		suma = suma + mat[zadati_red][j];
	}
	return suma;
}
void prikazi_red(int zadati_red, int m, TMATRICA mat) {
	printf("%d : ", zadati_red + 1);
	for (int j = 0; j < m; j++) {
		printf("%5d ", mat[zadati_red][j]);
	}
	printf(" -> %d\n", suma_reda(zadati_red, m, mat));
}
void prikazi_matricu_sa_sumom_reda(int n, int m, TMATRICA mat) {
	for (int i = 0; i < n; i++) {
		prikazi_red(i, m, mat);
	}
}
int main(void) {
	TMATRICA mat = {
		{ 1, 2, 3, 4 },
		{ 11, 22, 33, 44 }
	};
	int n = 2; //Broj redova matrice
	int m = 4; //Broj kolona matrice
	prikazi_matricu_sa_sumom_reda(n, m, mat);
```

```
	return 0;
}
```

124. Дата је матрица целих бројева. Написати функцију која рачуна суму задате колоне матрице.

```
#include <stdio.h>
#define MAX 10
typedef int TMATRICA[MAX][MAX];
int suma_kol(int n, TMATRICA mat, int zad_kol) {
	int i;
	int suma = 0;
	for (i = 0; i < n; i++) {
		suma += mat[i][zad_kol];
	}
	return suma;
}
int main(void) {
	TMATRICA mat = {
		{ 1, 2, 3, 4 },
		{ 5, 7, 7, 8 },
		{ 9, 10, 11, 12 }
	};
	// Broj redova
	int n = 3;
	// Broj kolona
	int m = 4;
	int zad_kol;
	printf("Unesite zadatu kolonu (1-%d): ", m);
	scanf("%d", &zad_kol);
	printf("Suma elemenata %d kolone: %d\n", zad_kol, suma_kol(n, mat, zad_kol - 1));
}
```

125. Дата је матрица целих бројева. Имплементирати следеће потпрограме:

а) Написати процедуру за унос матрице димензије nXm.

б) Написати процедуру за приказ матрице димензије nXm.

в) Написати функцију која рачуна суму задате колоне матрице.

г) Написати функцију која проналази и приказује највећу суму колоне матрице. Обавезно искористити функцију која рачуна суму задате колоне матрице.

```
#include <stdio.h>
#define MAX 10
typedef int TMATRICA[MAX][MAX];
void unos_matrice(int *n, int *m, TMATRICA mat) {
	int i, j;
	printf("Unesite broj (r)edova i (k)olona u formatu rxk: ");
	scanf("%dx%d", n, m);
	for (i = 0; i < *n; i++) {
		for (j = 0; j < *m; j++) {
			printf("Unesite [%d][%d]: ", i, j);
			scanf("%d", &mat[i][j]);
		}
		printf("\n");
	}
}
void prikaz_matrice(int n, int m, TMATRICA mat) {
	int i, j;
	for (i = 0; i < n; i++) {
		for (j = 0; j < m; j++) {
			printf("%5d", mat[i][j]);
		}
		printf("\n");
	}
```

```
}
int suma_kol(int n, TMATRICA mat, int zad_kol) {
	int i;
	int suma = 0;
	for (i = 0; i < n; i++) {
		suma += mat[i][zad_kol];
	}
	return suma;
}
int max_suma_kol(int n, int m, TMATRICA mat) {
	// F-ja vraca indeks kolone sa najvecom sumom
	int suma;
	int max_suma = suma_kol(n, mat, 0);
	int max_index = 0;
	int j = 1;
	while (j < m) {
		suma = suma_kol(n, mat, j);
		if (suma > max_suma) {
			max_suma = suma;
			max_index = j;
		}
		j++;
	}
	return max_index;
}
int main(void) {
	TMATRICA mat;
	// Broj redova
	int n = 0;
	// Broj kolona
	int m = 0;
	// Indeks kolone sa najvecom sumom
	int max_index;
	unos_matrice(&n, &m, mat);
	prikaz_matrice(n, m, mat);
	max_index = max_suma_kol(n, m, mat);
	printf("Kolona sa najvecom sumom je %d i ona iznosi %d.\n", (max_index + 1),
							suma_kol(n, mat, max_index));
}
```

126. Написати процедуру за унос елемената квадратне матрице целих бројева. Написати процедуру која приказује све елементе квадратне матрице целих бројева.

```
#include <stdio.h>
typedef int TMATRICA[10][10];
void unesi_elemente_matrice(int *n, TMATRICA M) {
	printf("\nUnesite dimenziju kvadratne matrice: ");
	scanf("%d", n);
	for (int i = 0; i < *n; i++) {
		for (int j = 0; j < *n; j++) {
			printf("Unesi [%d][%d] = ", i, j);
			scanf("%d", &M[i][j]);
		}
	}
}
void prikazi_matricu(int n, TMATRICA M) {
	for (int i = 0; i < n; i++) {
		for (int j = 0; j < n; j++) {
			printf("%5d", M[i][j]);
		}
```

```
			printf("\n");
		}
	}
	int main(void) {
		TMATRICA M;
		int n; //Dimenzija kvadratne matrice (nxn)
		unesi_elemente_matrice(&n, M);
		prikazi_matricu(n, M);
		return 0;
	}
```

127. Написати процедуру за унос елемената квадратне матрице целих бројева. Написати процедуру која приказује све елементе квадратне матрице целих бројева, при чему се на позицији елемената изнад главне дијагонале уместо вредности приказује симбол „*“.

```
#include <stdio.h>
typedef int TMATRICA[10][10];
void unesi_elemente_matrice(int *n, TMATRICA M) {
	printf("\nUnesite dimenziju kvadratne matrice: ");
	scanf("%d", n);

	for (int i = 0; i < *n; i++) {
		for (int j = 0; j < *n; j++) {
			printf("Unesi [%d][%d] = ", i, j);
			scanf("%d", &M[i][j]);
		}
	}
}
void prikazi_matricu_zvezdica_iznad_gd(int n, TMATRICA M) {
	for (int i = 0; i < n; i++) {
		for (int j = 0; j < n; j++) {
			if (j > i) {
				printf("%5s", "*");
			}
			else {
				printf("%5d", M[i][j]);
			}
		}
		printf("\n");
	}
}
int main(void) {
	TMATRICA M;
	int n; //Dimenzija kvadratne matrice (nxn)
	unesi_elemente_matrice(&n, M);
	prikazi_matricu_zvezdica_iznad_gd(n, M);
	return 0;
}
```

128. Дата је квадратна матрица целих бројева.
(а) Написати функцију која додаје нови елемент у низ али тако да низ остане сортиран у неопадајућем редоследу,
(б) Написати процедуру која формира неопадајући низ од елемената испод главне дијагонале матрице.

```
#include <stdio.h>
int vrati_poziciju(int x[], int n, int broj) {
	int i;
	for (i = 0; i<n; i++) {
		if (x[i]>broj) {
			return i;
```

```
			}
		}
		return n;
	}
	void dodaj_element(int x[], int *n, int broj) {
		int pozicija = vrati_poziciju(x, *n, broj);
		int i;
		for (i = *n; i >= (pozicija + 1); i--) {
			x[i] = x[i - 1];
		}
		x[pozicija] = broj;
		*n = *n + 1;
	}
	void prikazi_niz(int *niz, int n) {
		int i;
		printf("x:");
		for (i = 0; i < n; i++) {
			printf("%d\t", *(niz + i));
		}
	}
	void formiraj_niz(int m[3][3], int dimm, int x[], int *n) {
		int i, j;
		*n = 0;
		for (i = 0; i < dimm; i++) {
			for (j = 0; j<dimm; j++) {
				if (i > j) {
					dodaj_element(x, n, m[i][j]);
				}
			}
		}
	}
	int main(void) {
		int m[3][3] = {
			1, 2, 3,
			41, 5, 6,
			35, 23, 9
		};
		int x[10];
		int n = 0;
		formiraj_niz(m, 3, x, &n);
		prikazi_niz(x, n);
		return 0;
	}
```


4.2. Задаци за вежбање

129. Дата је матрица целих бројева *M* димензије *nXm*. Написати функцију која рачуна суму елемената задате колоне матрице.
130. Дата је матрица целих бројева *M* димензије *nXm*. Написати процедуру која приказује све елементе матрице и суму елемената сваке колоне (искористити функцију која рачуна суму елемената задате колоне матрице).
131. Дата је матрица целих бројева *M* димензије *nXm*. Написати функцију која проналази и приказује највећу суму реда матрице.
132. Дата је матрица целих бројева *M* димензије *nXm*. Имплементирати функцију која налази најмањи елемент матрице.
133. Дата је матрица целих бројева *M* димензије *nXm*. Имплементирати функцију која налази највећи елемент матрице.

134. Дата је матрица целих бројева *M* димензије *nXm*. Имплементирати функцију која налази најмањи елемент задате колоне матрице.
135. Дата је квадратна матрица целих бројева *M* димензије *nXn*. Имплементирати функцију која приказује све елементе изнад споредне дијагонале матрице.
136. Дата је квадратна матрица целих бројева *M* димензије *nXn*. Имплементирати функцију која приказује све елементе испод споредне дијагонале матрице.
137. Дата је квадратна матрица целих бројева *M* димензије *nXn*. Имплементирати функцију која формира низ од елемената споредне дијагонале тако да у низу не буде понављања.
138. Дата је квадратна матрица целих бројева *M* димензије *nXn*. Имплементирати функцију која формира низ од елемената изнад споредне дијагонале тако да у низу не буде понављања.
139. Дата је квадратна матрица целих бројева *M* димензије *nXn*. Имплементирати функцију која формира низ од елемената испод споредне дијагонале тако да у низу не буде понављања.
140. Дата је квадратна матрица целих бројева *M* димензије *nXn*. Имплементирати функцију која приказује све елементе изнад главне дијагонале матрице.
141. Дата је квадратна матрица целих бројева *M* димензије *nXn*. Имплементирати функцију која приказује све елементе испод главне дијагонале матрице.
142. Дата је квадратна матрица целих бројева *M* димензије *nXn*. Имплементирати функцију која формира низ од елемената главне дијагонале тако да у низу не буде понављања.
143. Дата је квадратна матрица целих бројева *M* димензије *nXn*. Имплементирати функцију која формира низ од елемената изнад главне дијагонале тако да у низу не буде понављања.
144. Дата је квадратна матрица целих бројева *M* димензије *nXn*. Имплементирати функцију која формира низ од елемената испод главне дијагонале тако да у низу не буде понављања.

4.3. Задаци са колоквијума

145. Дата је матрица целих бројева *M*. Имплементирати следеће потпрограме:
а) Имплементирати потпрограм који проверава да ли је задати ред матрице палиндром.
б) Имплементирати потпрограм који налази разлику (као скуповну операцију) два реда матрице и формира низ од тих елемената. У ново-креираном низу елементи не смеју да се понављају. Написати процедуру за приказ елемената низа и позвати је у оквиру овог потпрограма за приказ елемената ново-креираног низа.
в) Имплементирати непараметризовану процедуру **zadatak_3**. У процедури је дата матрица целих бројеваа произвољне димензије не веће од 10x10. У оквиру процедуре **zadatak_3** омогућити кориснику да унесе редни број реда матрице и провери да ли је задати ред матрице палиндром (позвати претходно имплементирани потпрограм). У оквиру процедуре **zadatak_3** омогућити кориснику да унесе два броја која представају задате редове матрице. Наћи пресек ова два реда (позвати претходно имплементирани потпрограм). Процедура **zadatak_3** мора бити имплементирана. Ово је предуслов да би се бодовале претходно имплементиране процедуре и функције.
146. Дата је матрица целих бројева *M*. Имплементирати следеће потпрограме:
а) Имплементирати потпрограм који проверава да ли је задата колона матрице палиндром.
б) Имплементирати потпрограм који налази разлику (као скуповну операцију) две колоне матрице и формира низ од тих елемената. У ново-креираном низу елементи не смеју да се понављају. Написати процедуру за приказ елемената низа и позвати је у оквиру овог потпрограма за приказ елемената ново-креираног низа.
в) Имплементирати непараметризовану процедуру **zadatak_3**. У процедури је дата матрица целих бројеваа произвољне димензије не веће од 10x10. У оквиру процедуре **zadatak_3** омогућити кориснику да унесе редни број колоне матрице и провери да ли је задатака колона матрице палиндром (позвати претходно имплементирани потпрограм). У оквиру процедуре **zadatak_3** омогућити кориснику да унесе два броја која представају задате колоне матрице.

Наћи пресек ове две колоне (позвати претходно имплементирани потпрограм). Процедура **zadatak_3** мора бити имплементирана. Ово је предуслов да би се бодовале претходно имплементиране процедуре и функције.

147. Дата је матрица целих бројева *M*. Имплементирати следеће потпрограме:

а) Имплементирати потпрограм **je_rastuci** који проверава да ли је задати ред матрице растући.

б) Имплементирати потпрограм **ima_podniz_parnih** који проверава да ли у низу целих бројева постоји подниз парних бројева чија је дужина већа од 2.

в) Имплементирати потпрограм **prebaci_obod_u_niz** који све елементе који се налазе по ободу матрице пребацује у низ. Написати процедуру за приказ елемената низа и позвати је у оквиру овог потпрограма за приказ елемената ново-креираног низа. У оквиру овог потпрограма позвати процедуру која проверава да ли у ново-креираном низу постоји подниз парних бројева чија је дужина већа од 2 (искористити претходно имплементирани потпрограм **ima_podniz_parnih**).

г) Имплементирати непараметризовану процедуру **zadatak_3**. У процедури је дата матрица целих бројева произвољне димензије не веће од 10x10. У оквиру процедуре **zadatak_3** омогућити кориснику да унесе редни број реда матрице и провери да ли је задати ред матрице растући (позвати претходно имплементирани потпрограм). У оквиру процедуре **zadatak_3** позвати претходно имплемнтирани потпрограм **prebaci_obod_u_niz**. Процедура **zadatak_3** мора бити имплементирана. Ово је предуслов да би се бодовале претходно имплементиране процедуре и функције.

148. Дата је матрица целих бројева *M*. Имплементирати следеће потпрограме:

а) Имплементирати потпрограм **je_rastuci** који проверава да ли је задати ред матрице растући.

б) Имплементирати потпрограм **ima_podniz_neparnih** који проверава да ли у низу целих бројева постоји подниз непарних бројева чија је дужина већа од 2.

в) Имплементирати потпрограм **prebaci_obod_u_niz** који све елементе који се налазе по ободу матрице пребацује у низ. Написати процедуру за приказ елемената низа и позвати је у оквиру овог потпрограма за приказ елемената ново-креираног низа. У оквиру ово потпрограма позвати процедуру која проверава да ли у ново-креираном низу постоји подниз непарних бројева чија је дужина већа од 2 (искористити претходно имплементирани потпрограм **ima_podniz_neparnih**).

г) Имплементирати непараметризовану процедуру **zadatak_3**. У процедури је дата матрица целих бројева произвољне димензије не веће од 10x10. У оквиру процедуре **zadatak_3** омогућити кориснику да унесе редни број реда матрице и провери да ли је задати ред матрице растући (позвати претходно имплементирани потпрограм). У оквиру процедуре **zadatak_3** позвати претходно имплемнтирани потпрограм **prebaci_obod_u_niz**. Процедура **zadatak_3** мора бити имплементирана. Ово је предуслов да би се бодовале претходно имплементиране процедуре и функције.

149. Дата је матрица целих бројева *M*. Имплементирати следеће потпрограме:

а) Имплементирати потпрограм **je_rastuci** који проверава да ли је задати ред матрице растући.

б) Имплементирати функцију **je_prost** која проверава да ли је број прост.

г) Имплементирати потпрограм **ima_podniz_prostih** који проверава да ли у низу целих бројева постоји подниз простих бројева чија је дужина већа од 2.

д) Имплементирати потпрограм **prebaci_obod_u_niz** који све елементе који се налазе по ободу матрице, а који су прости, пребацује у низ. Написати процедуру за приказ елемената низа и позвати је у оквиру овог потпрограма за приказ елемената ново-креираног низа. У оквиру овог потпрограма позвати процедуру која проверава да ли у ново-креираном низу постоји подниз простих бројева чија је дужина већа од 2 (искористити претходно имплементирани потпрограм **ima_podniz_prostih**).

ђ) Имплементирати непараметризовану процедуру **zadatak_3**. У процедури је дата матрица целих бројева произвољне димензије не веће од 10x10. У оквиру процедуре **zadatak_3**

омогућити кориснику да унесе редни број реда матрице и провери да ли је задати ред матрице растући (позвати претходно имплементирани потпрограм). У оквиру процедуре **zadatak_3** позвати претходно имплемнтирани потпрограм **prebaci_obod_u_niz**. Процедура **zadatak_3** мора бити имплементирана. Ово је предуслов да би се бодовале претходно имплементиране процедуре и функције.

150. Дата је улазна квадратна матрица целих бројева димензије 5 x 5. Написати функцију која проверава да ли се сви елементи на споредној дијагонали матрице налазе и на главној дијагонали матрице. На споредној и главној дијагонали елементи могу да се понављају.
Тестни пример:
улазна матрица

1	2	5	-1	1
4	8	4	1	6
2	4	6	7	8
3	4	7	1	2
8	8	11	9	3

Излаз: ДА

151. Написати функцију која омогућава кориснику унос елемената у квадратну матрицу целих бројева димензије N x N, али тако да сви елементи по колонама матрице буду међусобно различити (осим броја 0 који може да се понавља).
Корисник са стандардног улаза уноси бројеве. Пре уписа броја у матрицу треба проверити да ли се број већ налази у колони за коју се уносе подаци. Уколико елемент постоји у колони матрице, у текући елемент матрице уписати 0, у супротном уписати број који је корисник унео са стандардног улаза.
Тестни пример:
Крајњи изглед матрице:

1	2	5	3	5
0	8	0	0	6
2	4	0	0	8
3	0	7	8	2
4	18	0	9	3

152. Написати функцију која мења садржај улазне матрице на следећи начин: Свака 0 у колони се мења са највећим непарним бројем у тој колони. Уколико у колони не постоје непарни бројеви, садржај те колоне се не мења.
Тестни пример:

Улазна матрица

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

Излазна матрица

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

макс. непарни за 3 - 7 9 5 колону.

153. Написати функцију која омогућава кориснику унос елемената у квадратну матрицу целих бројева димензије N x N, али тако да сви елементи по редовима матрице буду међусобно различити (осим броја -1 који може да се понавља). Корисник са стандардног улаза уноси бројеве. Пре уписа броја у матрицу треба проверити да ли се број већ налази у колони за коју се уносе подаци. Уколико елемент постоји у колони матрице, у текући елемент матрице уписати -1, у супротном уписати број који је корисник унео са стандардног улаза.

154. Написати функцију која мења садржај улазне матрице на следећи начин: Сваки -1 број у матрици се мења са бројем који се налази на пресеку споредне и главне дијагонале матрице. Уколико се у пресеку главне и споредне дијагонале налази број -1, сви -1 бројеви у матрици се мењају бројем 0, осим елемента на пресеку споредне и главне дијагонале.
155. Дата је улазна квадратна матрица целих бројева димензије 5 x 5. Написати функцију која све елементе који се налазе испод споредне дијагонале квадратне матрице пребацује у низ.
Тестни пример:

Улазна матрица

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

Низ:

6	0	8	7	8	2	8	0	9	3

156. Дата је улазна квадратна матрица целих бројева димензије 5 x 5. Написати функцију која све елементе који се налазе у другој половини матрице пребацује у низ.
Тестни пример:

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

Низ:

3	4	7	8	2	1	8	0	9	3

157. Дата је улазна квадратна матрица целих бројева димензије 5 x 5. Написати функцију која упоређује суму елемената у првој дијагонали изнад главне дијагонале матрице и првој дијагонали испод главне дијагонале матрице и приказује поруку у следећем формату

```
Сума ел. у првој дијагонали изнад ГД је: _____
Сума ел. у првој дијагонали испод ГД је: _____
__< или >__
```

Тестни пример:

улазна матрица

1	2	5	-1	7
4	8	4	11	6
2	4	6	7	8
3	4	7	8	2
1	8	11	9	3

```
Сума ел. у првој дијагонали изнад ГД је: 15
Сума ел. у првој дијагонали испод ГД је: 24
15<24
```

158. Дата је улазна квадратна матрица целих бројева димензије 5 x 5. Написати функцију која упоређује суму елемената у првој дијагонали изнад споредне дијагонале матрице и првој дијагонали испод споредне дијагонале матрице и приказује поруку у следећем формату

```
Сума ел. у првој дијагонали изнад СД је:
Сума ел. у првој дијагонали испод СД је:
__< или >__
```

Тестни пример:

улазна матрица

1	2	5	-1	7
4	8	4	11	6
2	4	6	7	8
3	4	7	8	2
1	8	11	9	3

```
Сума ел. у првој дијагонали изнад СД је: 10
Сума ел. у првој дијагонали испод СД је: 28
10<28
```

159. Дата је улазна квадратна матрица целих бројева димензије 5 x 5. Написати функцију која све елементе које се налазе на првој дијагонали испод главне дијагонале матрице пребацује у низ X.
Тестни пример:

улазна матрица

1	2	5	-1	7
4	8	4	11	6
2	4	6	7	8
3	4	7	8	2
1	8	11	9	3

X:

6	7	7	8

160. Дата је улазна квадратна матрица целих бројева димензије 5 x 5. Написати функцију која за другу половину матрице (означена на слици) рачуна суму елемената по колонама и приказује у формату који је ниже дат на слици.
Тестни пример:

улазна матрица

1	2	5	-1	7
4	8	4	11	6
2	4	6	7	8
3	4	7	8	2
1	8	11	9	3

излаз на екрану

1	2	5	-1	7
4	8	4	11	6
2	4	6	7	8
3	4	7	8	2
1	8	11	9	3

S= 34 26

4.4. Задаци са испитних рокова

161. На стандардном улазу се уноси низ целих бројева који се завршава 0. Написати програм који од задатог низа формира матрицу тако да први ред матрице одговара унетом низу, а сваки наредни ред се добија цикличним померањем елемената низа за једно место улево.
Тестни пример:
Улаз: 4 5 6 9 0
Излаз:
4 5 6 9
5 6 9 4

6 9 4 5
9 4 5 6

162. Дата је улазна квадратна матрица целих бројева димензије 5 x 5. Написати потпрограм која рачуна унију елемената две задате колоне матрице и елементе уније смешта у нови низ. Унију елемената чине сви елементи који се налазе у ове две задате колоне, при чему није дозвољено понављање елемената.
163. Дата је улазна квадратна матрица целих бројева димензије 5 x 5. Написати потпрограм која рачуна пресек елемената две задате колоне матрице и елементе пресека смешта у нови низ. Пресек елемената чине они елементи који се налазе у првој колони и другој колони, при чему није дозвољено понављање елемената.
164. Дата је улазна квадратна матрица целих бројева димензије 5 x 5. Написати потпрограм која рачуна разлику елемената две задате колоне матрице и елементе разлике смешта у нови низ. Разлику елемената чине сви елементи који се налазе у првој колони а не налазе се у другој колони, при чему није дозвољено понављање елемената.
165. Дата је матрица целих бројева димензије nXm. Написати потпрограм која рачуна унију елемената два задата реда матрице и елементе уније смешта у нови низ. Унију елемената чине сви елементи који се налазе у ова два задата реда, при чему није дозвољено понављање елемената.
166. Дата је матрица целих бројева димензије nXm. Написати потпрограм која рачуна пресек елемената два задата реда матрице и елементе пресека смешта у нови низ. Пресек елемената чине они елементи који се налазе у првом реду и другом реду, при чему није дозвољено понављање елемената.
167. Дата је матрица целих бројева димензије nXm. Написати потпрограм која рачуна разлику елемената два задата реда матрице и елементе разлике смешта у нови низ. Разлику елемената чине сви елементи који се налазе у првом реду а не налазе се у другом реду, при чему није дозвољено понављање елемената.
168. Дата је матрица целих бројева димензије nXm. Написати потпрограм која рачуна унију елемената задатог реда и задате колоне матрице и елементе уније смешта у нови низ. Унију елемената чине сви елементи задатог реда и задате колоне, при чему није дозвољено понављање елемената.
169. Дата је матрица целих бројева димензије nXm. Написати потпрограм која рачуна пресек елемената задатог реда и задате колоне матрице и елементе пресека смешта у нови низ. Пресек елемената чине они елементи који налазе у задатом реду и задатој колони, при чему није дозвољено понављање елемената.
170. Дата је матрица целих бројева димензије nXm. Написати потпрограм која рачуна разлику елемената задатог реда и задате колоне матрице и елементе разлике смешта у нови низ. Разлику елемената чине сви елементи који се налазе у задатом реду а не налазе се задатој колони, при чему није дозвољено понављање елемената.
171. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна унију елемената главне и споредне дијагонале матрице и елементе уније смешта у нови низ. Унију елемената чине сви елементи главне и споредне дијагонале, при чему није дозвољено понављање елемената.
172. Дата је матрица целих бројева димензије n. Написати потпрограм која рачуна пресек елемената главне и споредне дијагонале матрице и елементе пресека смешта у нови низ. Пресек елемената чине они елементи који се налазе на главној и споредној дијагонали, при чему није дозвољено понављање елемената.
173. Дата је матрица целих бројева димензије n. Написати потпрограм која рачуна разлику елемената главне и споредне дијагонале матрице и елементе разлике смешта у нови низ. Разлику елемената чине сви елементи који се налазе на главној дијагонали а не налазе се на споредној дијагонали, при чему није дозвољено понављање елемената.

174. Дата је улазна квадратна матрица целих бројева димензије 5 x 5. Написати функцију која за квадратну матрицу произвољне димензије проверава да ли су дијагонале једнаке. Дијагонале су једнаке ако се на дијагоналама матрице налазе исти бројеви. Један исти елемент се може понављати више пута на дијагонали.
175. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна унију елемената изнад главне и изнад споредне дијагонале матрице и елементе уније смешта у нови низ. Унију елемената чине сви елементи изнад главне и изнад споредне дијагонале, при чему није дозвољено понављање елемената.
176. Дата је матрица целих бројева димензије n. Написати потпрограм која рачуна пресек елемената изнад главне и изнад споредне дијагонале матрице и елементе пресека смешта у нови низ. Пресек елемената чине они елементи који се налазе изнад главне и изнад споредне дијагонале, при чему није дозвољено понављање елемената.
177. Дата је матрица целих бројева димензије n. Написати потпрограм која рачуна разлику елемената изнад главне и изнад споредне дијагонале матрице и елементе разлике смешта у нови низ. Разлику елемената чине сви елементи који се налазе изнад главне дијагонале а не налазе се изнад споредне дијагонале, при чему није дозвољено понављање елемената.
178. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна унију елемената испод главне и испод споредне дијагонале матрице и елементе уније смешта у нови низ. Унију елемената чине сви елементи испод главне и испод споредне дијагонале, при чему није дозвољено понављање елемената.
179. Дата је матрица целих бројева димензије n. Написати потпрограм која рачуна пресек елемената испод главне и испод споредне дијагонале матрице и елементе пресека смешта у нови низ. Пресек елемената чине они елементи који се налазе испод главне и испод споредне дијагонале, при чему није дозвољено понављање елемената.
180. Дата је матрица целих бројева димензије n. Написати потпрограм која рачуна разлику елемената испод главне и испод споредне дијагонале матрице и елементе разлике смешта у нови низ. Разлику елемената чине сви елементи који се налазе испод главне дијагонале а не налазе се испод споредне дијагонале, при чему није дозвољено понављање елемената.
181. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна унију елемената изнад главне и испод споредне дијагонале матрице и елементе уније смешта у нови низ. Унију елемената чине сви елементи изнад главне и испод споредне дијагонале, при чему није дозвољено понављање елемената.
182. Дата је матрица целих бројева димензије n. Написати потпрограм која рачуна пресек елемената изнад главне и испод споредне дијагонале матрице и елементе пресека смешта у нови низ. Пресек елемената чине они елементи који се налазе изнад главне и испод споредне дијагонале, при чему није дозвољено понављање елемената.
183. Дата је матрица целих бројева димензије n. Написати потпрограм која рачуна разлику елемената изнад главне и испод споредне дијагонале матрице и елементе разлике смешта у нови низ. Разлику елемената чине сви елементи који се налазе изнад главне дијагонале а не налазе се испод споредне дијагонале, при чему није дозвољено понављање елемената.
184. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна унију елемената испод главне и изнад споредне дијагонале матрице и елементе уније смешта у нови низ. Унију елемената чине сви елементи испод главне и изнад споредне дијагонале, при чему није дозвољено понављање елемената.
185. Дата је матрица целих бројева димензије n. Написати потпрограм која рачуна пресек елемената испод главне и изнад споредне дијагонале матрице и елементе пресека смешта у нови низ. Пресек елемената чине они елементи који се налазе испод главне и изнад споредне дијагонале, при чему није дозвољено понављање елемената.
186. Дата је матрица целих бројева димензије n. Написати потпрограм која рачуна разлику елемената испод главне и изнад споредне дијагонале матрице и елементе разлике смешта у

нови низ. Разлику елемената чине сви елементи који се налазе испод главне дијагонале а не налазе се изнад споредне дијагонале, при чему није дозвољено понављање елемената.

187. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна унију елемената главне дијагонале матрице и елемената задате колоне и елементе уније смешта у нови низ. Унију елемената чине сви елементи главне дијагонале и елементи задате колоне, при чему није дозвољено понављање елемената.
188. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна пресек елемената главне дијагонале матрице и елемената задате колоне и елементе пресека смешта у нови низ. Пресек елемената чине они елементи који се налазе на главној дијагонали и задатој колони, при чему није дозвољено понављање елемената.
189. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна разлику елемената главне дијагонале матрице и елемената задате колоне и елементе разлике смешта у нови низ. Разлику елемената чине они елементи који се налазе на главној дијагонали а не налазе се у задатој колони, при чему није дозвољено понављање елемената.
190. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна унију елемената изнад главне дијагонале матрице и елемената задате колоне и елементе уније смешта у нови низ. Унију елемената чине сви елементи изнад главне дијагонале и елементи задате колоне, при чему није дозвољено понављање елемената.
191. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна пресек елемената изнад главне дијагонале матрице и елемената задате колоне и елементе пресека смешта у нови низ. Пресек елемената чине они елементи који се налазе изнад главне диагонале и у задатој колони, при чему није дозвољено понављање елемената.
192. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна разлику елемената изнад главне дијагонале матрице и елемената задате колоне и елементе разлике смешта у нови низ. Разлику елемената чине они елементи који се налазе изнад главне дијагонале а не налазе се у задатој колони, при чему није дозвољено понављање елемената.
193. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна унију елемената испод главне дијагонале матрице и елемената задате колоне и елементе уније смешта у нови низ. Унију елемената чине сви елементи испод главне дијагонале и елементи задате колоне, при чему није дозвољено понављање елемената.
194. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна пресек елемената испод главне дијагонале матрице и елемената задате колоне и елементе пресека смешта у нови низ. Пресек елемената чине они елементи који се налазе испод главне диагонале и у задатој колони, при чему није дозвољено понављање елемената.
195. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна разлику елемената испод главне дијагонале матрице и елемената задате колоне и елементе разлике смешта у нови низ. Разлику елемената чине они елементи који се налазе испод главне дијагонале а не налазе се у задатој колони, при чему није дозвољено понављање елемената.
196. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна унију елемената споредне дијагонале матрице и елемената задатог реда и елементе уније смешта у нови низ. Унију елемената чине сви елементи споредне дијагонале и елементи задатог реда, при чему није дозвољено понављање елемената.
197. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна пресек елемената споредне дијагонале матрице и елемената задатог реда и елементе пресека смешта у нови низ. Пресек елемената чине они елементи који се налазе на споредној дијагонали и задатом реду, при чему није дозвољено понављање елемената.
198. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна разлику елемената споредне дијагонале матрице и елемената задатог реда и елементе разлике смешта у нови низ. Разлику елемената чине они елементи који се налазе на споредној дијагонали а не налазе се у задатом реду, при чему није дозвољено понављање елемената.

199. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна унију елемената изнад споредне дијагонале матрице и елемената задатог реда и елементе уније смешта у нови низ. Унију елемената чине сви елементи изнад споредне дијагонале и елементи задатог реда, при чему није дозвољено понављање елемената.
200. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна пресек елемената изнад споредне дијагонале матрице и елемената задатог реда и елементе пресека смешта у нови низ. Пресек елемената чине они елементи који се налазе изнад споредне диагонале и у задатом реду, при чему није дозвољено понављање елемената.
201. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна разлику елемената изнад споредне дијагонале матрице и елемената задатог реда и елементе разлике смешта у нови низ. Разлику елемената чине они елементи који се налазе изнад споредне дијагонале а не налазе се у задатом реду, при чему није дозвољено понављање елемената.
202. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна унију елемената испод споредне дијагонале матрице и елемената задатог реда и елементе уније смешта у нови низ. Унију елемената чине сви елементи испод споредне дијагонале и елементи задатог реда, при чему није дозвољено понављање елемената.
203. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна пресек елемената испод споредне дијагонале матрице и елемената задатог реда и елементе пресека смешта у нови низ. Пресек елемената чине они елементи који се налазе испод споредне диагонале и у задатом реду, при чему није дозвољено понављање елемената.
204. Дата је квадратна матрица целих бројева димензије n. Написати потпрограм која рачуна разлику елемената испод споредне дијагонале матрице и елемената задатог реда и елементе разлике смешта у нови низ. Разлику елемената чине они елементи који се налазе испод споредне дијагонале а не налазе се у задатом реду, при чему није дозвољено понављање елемената.

5. Стрингови

У овом одељку приказани су стрингови. Дати су решени задаци, задаци за вежбање, задаци са колоквијума и задаци са испитних рокова.

5.1. Решени задаци

205. Дат је стринг самогласника. Написати функцију која рачуна дужину стринга. Приказати стринг самогласника и његову дужину.

```
#include <stdio.h>
#include <string.h>
int vrati_duzinu_stringa(char str[]) {
	char ch = str[0];
	int brojac = 0;
	while (ch != '\0') {
		brojac++;
		ch = str[brojac];
	}
	return brojac;
}
int main(void) {
	char samoglasnici[] = { 'A', 'E', 'I', 'O', 'U', '\0' };
	printf("\nSamoglasnici su: %s\n", samoglasnici);
	int duzina = vrati_duzinu_stringa(samoglasnici);
	printf("String %s ima %d karaktera\n", samoglasnici, duzina);
	duzina = strlen(samoglasnici);
	printf("String %s ima %d karaktera\n", samoglasnici, duzina);
	return 0;
}
```

206. Написати функцију која испитује да ли је одређени знак самогласник. Унети неки стринг и приказати самогласнике.

```
#include <string.h>
#include <stdio.h>
int samoglasnik(char c) {
	if (c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U' || c == 'a' || c == 
'e' || c == 'i' || c == 'o' || c == 'u') {
		return 1;
	}
	return 0;
}
void prikazi_samoglasnike(char s[]) {
	int i;
	int duzinaString = strlen(s);
	for (i = 0; i < duzinaString; i++) {
		if (samoglasnik(s[i]) == 1) {
			printf("Samoglasnik na poziciji %d je %c\n", i, s[i]);
		}
	}
}
int main(void) {
	char s[100];
	printf("\nUnesite string:");
	gets(s);
	printf("Korisnik je uneo: %s\n", s);
	prikazi_samoglasnike(s);
	return 0;
}
```

207. Написати потпрограм који омогућава прихватање знакова са стандардног улаза који представљају име и презиме, као и њихово чување у оквиру стринга. Приказати унесено име и презиме.

```
#include <stdio.h>
#include <string.h>
void prihvati_string(char str[]) {
	char ch;
	int brEl = 0;
	do {
		ch = getchar();
		if (ch != '\n') {
			str[brEl++] = ch;
		}
	} while (ch != '\n');
	str[brEl] = '\0';
}
int main(void) {
	char ime_prezime[30];
	printf("\nUnesite ime i prezime: ");
	prihvati_string(ime_prezime);
	printf("Vase ime i prezime je %s\n", ime_prezime);
	return 0;
}
```

208. Написати функцију која проверава да ли је дати стринг палиндром.

```
#include <stdio.h>
#include<string.h>
int palindrom(char x[]) {
	int i;
	int n = strlen(x);
	int broj_poredjenja = n / 2;
	for (i = 0; i < broj_poredjenja; i++) {
		if (x[i] != x[n - i - 1]) {
			// String nije palindrom
			return 0;
		}
	}
	// Jeste palindrom
	return 1;
}
int main(void) {
	char x[100] = "anavolimilovana";
	if (palindrom(x) == 1) {
		printf("%s je palindrom.\n");
	}
	else {
		printf("%s nije palindrom.\n");
	}
	return 0;
}
```

5.2. Задаци за вежбање

209. Имплементирати функцију која од на основу задатог стринга креира нови стринг који садржи само цифре.
210. Имплементирати функцију која проверава да ли у задатом стрингу постоје цифре.
211. Имплементирати функцију која на основу задатог стринга креира нови стринг који садржи само самогласнике.

212. Имплементирати функцију која на основу задатог стринга креира нови стринг који садржи само сугласнике.
213. Имплементирати функцију која на основу задатог стринга креира нови стринг који садржи најдужи подстринг самогласника.
Улазни стринг: 12 фг ааа бвцф ссд ае
Излазни стринг: ааа
214. Имплементирати функцију која на основу задатог стринга креира нови стринг који садржи најкраћи подстринг самогласника.
Улазни стринг: 12 фг ааа бвцф ссд ае
Излазни стринг: ае
215. Имплементирати функцију која на основу задатог стринга креира нови стринг који садржи најдужи подстринг сугласника.
Улазни стринг: 12 фг ааа бвцф ссд ае
Излазни стринг: бвцф
216. Имплементирати функцију која на основу задатог стринга креира нови стринг који садржи најкраћи подстринг сугласника.
Улазни стринг: 12 фг ааа бвцф ссд ае
Излазни стринг: фг

5.3. Задаци са колоквијума

217. Имплементирати следеће потпрограме:
а) Функцију која проверава да ли је задати карактер слово. Уколико вам је потребно можете да искористите чињеницу да карактер *A* има ASCI вредност 65, а карактер *a* има ASCI вредност 97.
б) Функцију која на основу улазног стринга креира најдужи подниз кога чине слова (A-Z и a-z). Тестни пример: улазни стринг s23d f e4asdd44f.ff//, излазни стринг asdd.
218. Имплементирати следеће потпрограме:
а) Функцију која проверава да ли је задати карактер цифра.
б) Функцију која на основу улазног стринга креира најдужи подниз кога чине цифре. Тестни пример: улазни стринг 12 23 3553 =((=)&, излазни стринг 3553.

5.4. Задаци са испитних рокова

219. Имплементирати следеће потпрограме:
а) Имплементирати функцију која прихвата са стандардног улаза карактер по карактер све док корисник не унесе знак минус (-) и формира стринг кога чине само цифре (максималан број карактера у стрингу цифара је 100).
б) Имплементирати функцију која у стрингу кога чине цифре налази најдужи подниз кога чине парне цифре.
220. Имплементирати програм који компресује низ карактера тако што сваку серију узастопно истих симбола дужине веће од три приказује као знак који чини ту серију за којом у малој загради следи број узастопно поновљених симбола.
Тестни пример:
Низ карактера:
ASSSSDDDDhjkkkCCCCC
Компресија:
AS(4)DDDhjkkC(5)
221. Написати функцију која за два задата низа карактера проверава да ли су анаграми. Два низа су анаграми уколико се од карактера првог низа може формирати други низ.

Тест пример 1:

Ако је дат први низ А: {1,a,b,4,g,t,t} , а други низ Б: {t,a,b,t,g,1,4} тада ова два низа ЈЕСУ анаграми

Тест пример 2:

Ако је дат први низ А: {1,a,b,4,g,t,t }, а други низ Б:{ 1,a,b,4,t,t,t } тада ова два низа НИСУ анаграми

222. Имплементирати функцију која компресује низ карактера тако што сваку серију узастопно истих симбола дужине веће од три приказује као знак који чини ту серију за којом у малој загради следи број узастопно поновљених симбола.

Тестни пример:

Низ карактера:

ASSSSDDDDhjkkkCCCCC

Компресија:

AS(4)DDDhjkkC(5)

223. Имплеметирати функцију која проверава да ли је у изразу (дат као низ карактера) број и редослед заграда коректан (нема затворене пре отворене заграде).

Тесни пример:

Коректно: a/(a1* (5+b1)-3*c)

Коректно: ((a(h(88)aa)a)a)

Коректно: ((a(h(88)()aa)a)a)

Некоректно: a/(a1))* (5+b1-3*c)

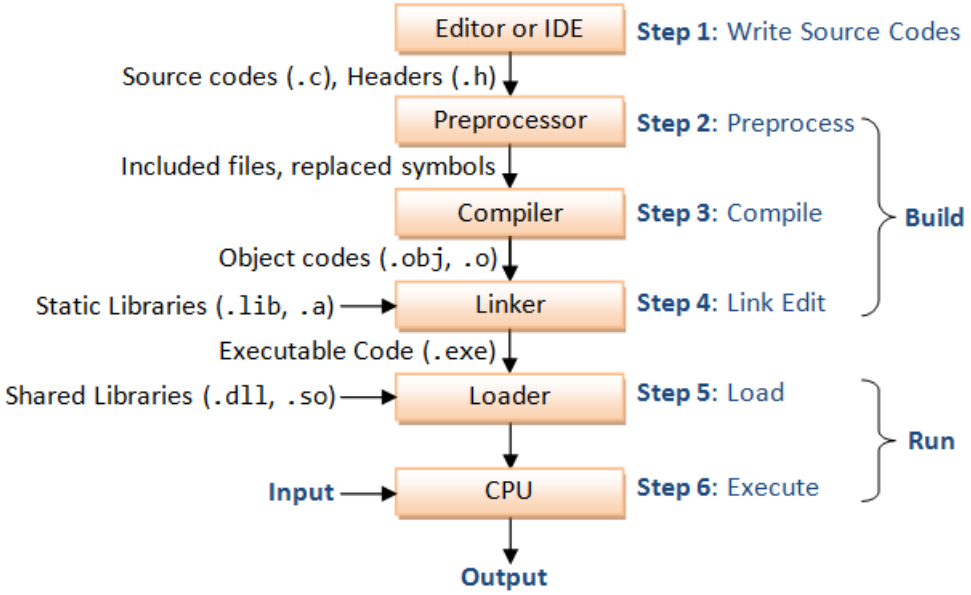
Некоректно: ss)(aa)(fff

Некоректно: ())(

6. Прилози

У овом одељку налазе се следећи прилози:
– Фазе процеса израде програма
– Коришћење интегрисаног развојног окружења *Visual Studio 2010*
– Структура заглавља програма
– Процес израде програма – од монолитног до структурираног програма
– Процес израде програма – израда корисничког менија

6.1. Фазе процеса израде програма



Слика 2. Фазе процеса израде програма

6.2. Коришћење интегрисаног развојног окружења *Visual Studio 2010*

Погледати датотеку Креирање C програма у VS2010.

6.3. Структура заглавља програма

Датотека: *Pr_3. ZaglavljePrograma.c*

```
/*
** PROJEKAT : <Naziv projekta. Npr: Osnove programiranja u programskom jeziku C>
** DATOTEKA : <Naziv datoteke. Npr. P1_promenljive.c ili P1_promenljive.h>
** OPIS     : <Opis sadržaja datoteke ili verbalna formulacija zahteva.
**            Npr: Napisati program u programskom jeziku C koji
**            prikazuje globalne i lokalne promenljive.>
** DATUM    : <Datum kreiranja datoteke. Npr: 24.03.2016.>
** AUTOR    : <Ime i prezime autora programskog koda; može i e-mail adresa.>
** PROMENE  :
**   21.07.2017. - promenjen je prototip f-je za prikaz rezultata (SDL)
**   xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/
```

6.4. Процес израде програма – од монолитног до структурираног програма

Верзија 1

Датотека: *P02_01_krug.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : P02_01_krug.c
** OPIS     : Napisati program u programskom jeziku C koji na standardnom
**            izlaznom uredjaju
**            prikazuje površine i obime krugova ciji su poluprecnici zadati:
**            11.1, 12.2, 13.3, 14.4, 15.5 .
** DATUM    : 26.02.2016.
** AUTOR    : S.D.L.
** PROMENE  :
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/

#include <stdio.h>

#define      PI   3.1415926536

const double pi = 3.1415926536;

double pp1, pp2 = 12.2, pp3, pp4, pp5;
double pv1, pv2, pv3, pv4, pv5,
       ob1, ob2, ob3, ob4, ob5;

int  main(void)
{
       printf(">>> P02_01_krug.c  <<<\n\n");
       // Dodeljivanje vrednosti poluprecnicima: 11.1, 12.2, 13.3, 14.4, 15.5
       pp1 = 11.10000;
       pp5 = 1.1 + (pp4 = (1.1 + (pp3 = 13.3)));
       // Izracunavanje površine
       pv1 = pp1 * pp1 * pi;
       pv2 = pp2 * pp2 * pi;
       pv3 = pp3 * pp3 * pi;
       pv4 = pp4 * pp4 * pi;
       pv5 = pp5 * pp5 * pi;
       // Izracunavanje obima
       ob1 = 2 * pp1 * PI;
       ob2 = 2 * pp2 * PI;
       ob3 = 2 * pp3 * PI;
       ob4 = 2 * pp4 * PI;
       ob5 = 2 * pp5 * PI;
       // Prikaz dobijenih rezultata
       printf(" POVRSINE I OBIMI KRUGOVA\n\n");
       printf(" Poluprecnik     Povrsina     Obim\n"
              "==================================\n");
       printf("%12.1lf %12.3lf %8.3lf\n", pp1, pv1, ob1);
       printf("%12.1lf %12.3lf %8.3lf\n", pp2, pv2, ob2);
       printf("%12.1lf %12.3lf %8.3lf\n", pp3, pv3, ob3);
       printf("%12.1lf %12.3lf %8.3lf\n", pp4, pv4, ob4);
       printf("%12.1lf %12.3lf %8.3lf\n", pp5, pv5, ob5);
       return 0;
}
```

Верзија 2

Датотека: *P02_02_krug.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : P02_02_krug.c
** OPIS     : Napisati program u programskom jeziku C koji na standardnom
**            izlaznom uredjaju
**            prikazuje površine i obime krugova ciji su poluprecnici zadati
**            (11.1, 12.2, 13.3, 14.4, 15.5)
**            upotrebom funkcija.
** DATUM    : 26.02.2016.
** AUTOR    : S.D.L.
** PROMENE  :
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/

#include <stdio.h>

#define      PI   3.1415926536

const double pi = 3.1415926536;

double pp1, pp2 = 12.2, pp3, pp4, pp5;
double pv1, pv2, pv3, pv4, pv5,
       ob1, ob2, ob3, ob4, ob5;

void IzracunajPovrsinu(void) {
      pv1 = pp1 * pp1 * pi;
      pv2 = pp2 * pp2 * pi;
      pv3 = pp3 * pp3 * pi;
      pv4 = pp4 * pp4 * pi;
      pv5 = pp5 * pp5 * pi;
}

void IzracunajObim(void) {
      ob1 = 2 * pp1 * PI;
      ob2 = 2 * pp2 * PI;
      ob3 = 2 * pp3 * PI;
      ob4 = 2 * pp4 * PI;
      ob5 = 2 * pp5 * PI;
}

void PrikazZaglavlja(void) {
     printf(" POVRSINE I OBIMI KRUGOVA\n\n");
     printf(" Poluprecnik     Povrsina     Obim\n"
            "===================================\n");
}

void PrikazRezultata(void) {
      printf("%12.1lf %12.3lf %8.3lf\n", pp1, pv1, ob1);
      printf("%12.1lf %12.3lf %8.3lf\n", pp2, pv2, ob2);
      printf("%12.1lf %12.3lf %8.3lf\n", pp3, pv3, ob3);
      printf("%12.1lf %12.3lf %8.3lf\n", pp4, pv4, ob4);
      printf("%12.1lf %12.3lf %8.3lf\n", pp5, pv5, ob5);
}
```

```
int  main(void)
{
       printf(">>> P02_02_krug.c  <<<\n\n");
       // Dodeljivanje vrednosti poluprecnicima: 11.1, 12.2, 13.3, 14.4, 15.5
       pp1 = 11.10000;
       pp5 = 1.1 + (pp4 = (1.1 + (pp3 = 13.3)));
       // Izracunavanje povrsine
       IzracunajPovrsinu();
       // Izracunavanje obima
       IzracunajObim();
       // Prikaz dobijenih rezultata
       PrikazZaglavlja();
       PrikazRezultata();

       return 0;
}
```

Верзија 3

Датотека: *P02_03_krug.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : P02_03_krug.c
** OPIS     : Napisati program u programskom jeziku C koji na standardnom
**            izlaznom uredjaju
**            prikazuje površine i obime krugova ciji su poluprecnici zadati
**            (11.1, 12.2, 13.3, 14.4, 15.5)
**            upotrebom funkcija i funkcijskih prototipova.
** DATUM    : 26.02.2016.
** AUTOR    : S.D.L.
** PROMENE  :
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/

#include <stdio.h>

#define      PI   3.1415926536

const double pi = 3.1415926536;

double pp1 = 11.1, pp2 = 12.2, pp3 = 13.3, pp4 = 14.4, pp5 = 15.5;
double pv1, pv2, pv3, pv4, pv5,
       ob1, ob2, ob3, ob4, ob5;

void IzracunajPovrsinu(void);
void IzracunajObim(void);
void PrikazZaglavlja(void);
void PrikazRezultata(void);

int  main(void)
{
       printf(">>> P02_03_krug.c  <<<\n\n");
       IzracunajPovrsinu();
       IzracunajObim();
       PrikazZaglavlja();
       PrikazRezultata();
```

```
	return 0;
}

void IzracunajPovrsinu(void) {
	pv1 = pp1 * pp1 * pi;
	pv2 = pp2 * pp2 * pi;
	pv3 = pp3 * pp3 * pi;
	pv4 = pp4 * pp4 * pi;
	pv5 = pp5 * pp5 * pi;
}

void IzracunajObim(void) {
	ob1 = 2 * pp1 * PI;
	ob2 = 2 * pp2 * PI;
	ob3 = 2 * pp3 * PI;
	ob4 = 2 * pp4 * PI;
	ob5 = 2 * pp5 * PI;
}

void PrikazZaglavlja(void) {
	printf(" POVRSINE I OBIMI KRUGOVA\n\n");
	printf(" Poluprecnik     Povrsina     Obim\n"
		"==================================\n");
}

void PrikazRezultata(void) {
	printf("%12.1lf %12.3lf %8.3lf\n", pp1, pv1, ob1);
	printf("%12.1lf %12.3lf %8.3lf\n", pp2, pv2, ob2);
	printf("%12.1lf %12.3lf %8.3lf\n", pp3, pv3, ob3);
	printf("%12.1lf %12.3lf %8.3lf\n", pp4, pv4, ob4);
	printf("%12.1lf %12.3lf %8.3lf\n", pp5, pv5, ob5);
}
```

Верзија 4

Датотека: *P02_04_krug.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : P02_04_krug.c
** OPIS     : Napisati program u programskom jeziku C koji na standardnom
**            izlaznom uredjaju
**            prikazuje površine i obime krugova ciji su poluprecnici zadati
**            (11.1, 12.2, 13.3, 14.4, 15.5)
**            upotrebom funkcija, funkcijskih prototipova i pomocnih funkcija
**            za izracunavanje povrsine i obima.
** DATUM    : 26.02.2016.
** AUTOR    : S.D.L.
** PROMENE  :
**   xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/

#include <stdio.h>

#define PI    3.1415926536

double  pp1 = 11.1, pp2 = 12.2, pp3 = 13.3, pp4 = 14.4, pp5 = 15.5;
double  pv1, pv2, pv3, pv4, pv5,
```

```
        ob1, ob2, ob3, ob4, ob5;

void IzracunajPovrsinu(void);
void IzracunajObim(void);
void PrikazZaglavlja(void);
void PrikazRezultata(void);

int  main(void)
{
       printf(">>> P02_04_krug.c  <<<\n\n");
       IzracunajPovrsinu();
       IzracunajObim();
       PrikazZaglavlja();
       PrikazRezultata();
       return 0;
}

////////////////////////////////////////////////////////
double PovKruga(double pp) {
       double pov;
       pov = pp * pp * PI;
       return pov;
}

void IzracunajPovrsinu(void) {
       pv1 = PovKruga(pp1);
       pv2 = PovKruga(pp2);
       pv3 = PovKruga(pp3);
       pv4 = PovKruga(pp4);
       pv5 = PovKruga(pp5);
}
////////////////////////////////////////////////////////

double ObKruga(double pp) {
       return 2 * pp * PI;
}

void IzracunajObim(void) {
       ob1 = ObKruga(pp1);
       ob2 = ObKruga(pp2);
       ob3 = ObKruga(pp3);
       ob4 = ObKruga(pp4);
       ob5 = ObKruga(pp5);
}
////////////////////////////////////////////////////////

void PrikazZaglavlja(void) {
       printf(" POVRSINE I OBIMI KRUGOVA\n\n");
       printf(" Poluprecnik     Povrsina     Obim\n"
              "==================================\n");
}

void PrikazRezultata(void) {
       printf("%12.1lf %12.3lf %8.3lf\n", pp1, pv1, ob1);
       printf("%12.1lf %12.3lf %8.3lf\n", pp2, pv2, ob2);
       printf("%12.1lf %12.3lf %8.3lf\n", pp3, pv3, ob3);
       printf("%12.1lf %12.3lf %8.3lf\n", pp4, pv4, ob4);
       printf("%12.1lf %12.3lf %8.3lf\n", pp5, pv5, ob5);
}
```

Верзија 5

Датотека: *P02_05_krug.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : P02_05_krug.c
** OPIS     : Napisati program u programskom jeziku C koji na standardnom
**            izlaznom uredjaju
**            prikazuje povrsšine i obime krugova ciji su poluprecnici zadati
**            (11.1, 12.2, 13.3, 14.4, 15.5)
**            upotrebom funkcija, funkcijskih prototipova i pomocnih funkcija
**            za izracunavanje povrsine i obima.
**            Umesto skalara (promenljivih skalarnog tipa) koristiti
**            nizove skalara (promenljive tipa niz skalara).
** DATUM    : 26.02.2016.
** AUTOR    : S.D.L.
** PROMENE  :
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/

#include <stdio.h>

#define PI    3.1415926536

double  pp[5] = {11.1, 12.2, 13.3, 14.4, 15.5};
double  pv[5], ob[5];

void IzracunajPovrsinu(void);
void IzracunajObim(void);
void PrikazZaglavlja(void);
void PrikazRezultata(void);

int  main(void)
{
       printf(">>> P02_05_krug.c  <<<\n\n");
       IzracunajPovrsinu();
       IzracunajObim();
       PrikazZaglavlja();
       PrikazRezultata();
       return 0;
}

////////////////////////////////////////////////////////
double PovKruga(double pp) {
      double pov;
      pov = pp * pp * PI;
      return pov;
}

void IzracunajPovrsinu(void) {
       int i;
       for (i = 0; i<=4; i = i+1)
              pv[i] = PovKruga(pp[i]);
}
////////////////////////////////////////////////////////
```

```
double ObKruga(double pp) {
	return 2 * pp * PI;
}

void IzracunajObim(void) {
	for (int i = 0; i<=4; i++)
		ob[i] = ObKruga(pp[i]);
}
//////////////////////////////////////////////////////

void PrikazZaglavlja(void) {
	printf(" POVRSINE I OBIMI KRUGOVA\n\n");
	printf(" Poluprecnik	Povrsina	Obim\n"
		"==================================\n");
}

void PrikazRezultata(void) {
	for (int i = 0; i<=4; i++)
	printf("%12.1lf %12.3lf %8.3lf\n", pp[i], pv[i], ob[i]);
}
```

Верзија 6

Датотека: *P02_06_krug.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : P02_06_krug.c
** OPIS     : Napisati program u programskom jeziku C koji na standardnom
**            izlaznom uredjaju
**            prikazuje povrsšine i obime krugova ciji su poluprecnici zadati
**            (11.1, 12.2, 13.3, 14.4, 15.5)
**            upotrebom funkcija, funkcijskih prototipova i pomocnih funkcija
**            za izracunavanje povrsine i obima.
**            Umesto skalara (promenljivih skalarnog tipa) koristiti
**            nizove skalara (promenljive tipa niz skalara).
**            Program podeliti u tri datoteke:
**              1) P02_06_krug.c - sadrzi glavnu f-ju main()
**              2) P02_krug.h –
**                 sadrzi specifikaciju operacije koje se pozivaju iz glavne f-je
**              3) P02_krug.c –
**                 sadrzi implementaciju operacija koje su specificirane u *.h datoteci
** DATUM    : 26.02.2016.
** AUTOR    : S.D.L.
** PROMENE  :
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/

//#include <stdio.h>
#include "P02_06_krug_spec.h"

double pp[5] = {11.1, 12.2, 13.3, 14.4, 15.5};
char * poruka = "P02_06_krug.c";

int main(void) {
	double pv[5], ob[5];
	Info(poruka);
	IzracunajPovrsinu(pp, pv);
```

```
	IzracunajObim(pp, ob);
	PrikazZaglavlja();
	PrikazRezultata(pp, pv, ob);
	return 0;
}
```

Датотека: *P02_06_krug_spec.h*

```
// P02_06_krug_spec.h  :: Specifikacija operacija

#ifndef _P02_06_KRUG_SPEC_H
#define _P02_06_KRUG_SPEC_H

void Info(char *);
void IzracunajPovrsinu(const double[], double[]);
void IzracunajObim(const double[], double[]);
void PrikazZaglavlja(void);
void PrikazRezultata(const double[], const double[], const double[]);

#endif /* _P02_06_KRUG_SPEC_H */
```

Датотека: *P02_06_krug_impl.c*

```
// P02_krug.c  ::  Implementacija operacija

#include <stdio.h>
#include "P02_06_krug_spec.h"

#define PI  3.1415926536

//== Specifikacija internih funkcija ================================================

double PovKruga(double pp);
double ObKruga(double pp);

//================================================================================ Kraj ==

//== Implementacija eksternih funkcija ==============================================

inline void Info(char * txt) {
	printf("\n>>>  %s  <<<\n\n", txt);
}

void IzracunajPovrsinu(const double pp[], double pv[]) {
	unsigned short int i;
	// pp[0] = 10;            // Prijavljuje se greska, jer je: const double pp[]
	for (i = 0; i <= 4; i = i + 1)
		pv[i] = PovKruga(pp[i]);
}

void IzracunajObim(const double pp[], double ob[]) {
	for (size_t i = 0; i <= 4; i++)
		ob[i] = ObKruga(pp[i]);
}

void PrikazZaglavlja(void) {
	printf(" POVRSINE I OBIMI KRUGOVA\n\n");
	printf(" Poluprecnik     Povrsina     Obim\n"
		"==================================\n");
}
```

```
void PrikazRezultata(const double pp[], const double pv[], const double ob[]) {
	for (size_t i = 0; i <= 4; i++)
	printf("%12.1lf %12.3lf %8.3lf\n", pp[i], pv[i], ob[i]);
}
//======================================================================== Kraj ==

//== Implementacija internih funkcija ============================================

inline double PovKruga(double pp) {
	double pov;
	pov = pp * pp * PI;
	return pov;
}

inline double ObKruga(double pp) {
	return 2 * pp * PI;
}
//======================================================================== Kraj ==
```

6.5. Процес израде програма – израда корисничког менија

Верзија 1

Датотека: *meni_01.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : meni_01.c
** OPIS     : Napisati program u programskom jeziku C koji
**            implementira konzolni korisnički interfejs
**            upotrebom menija.
**            V.1
** DATUM    : 26.02.2016.
** AUTOR    : S.D.L.
** PROMENE  :
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2016.
*/

/*

========
STUDENT
========

0. Kraj rada

1. Ubaci     (Insert)
2. Izbaci    (Delete)
3. Promeni   (Update)
4. Prikazi   (Select)

	4.0. Kraj (povratak u prethodni meni)

	4.1. Prikazi sve
	4.2. Prikazi po broju indeksa (BI)
	4.3. Prikazi po prezimenu
```

```
*/

#include <stdio.h>
#include <conio.h>

#define KRAJ 0

int  GlavniMeni(void);
void Izvrsi(const int);
void Odjava(void);
void UbaciStud(void);
void IzbaciStud(void);
void PromeniStud(void);

int  PrikaziMeni(void);
void IzvrsiPrikaz(const int);
void PrikaziStud(void);
void PrikaziSve(void);
void PrikaziPoBI(void);
void PrikaziPoPrez(void);

int main(void){
	int izbor;

	do {
		izbor = GlavniMeni();
		Izvrsi(izbor);
	} while (izbor != KRAJ);

	return 0;
}

int GlavniMeni(void) {
	clrscr();
	printf("\n=============\n STUDENT\n=============");
	printf("\n GLAVNI MENI\n");
	printf("\n 0. Kraj rada\n");
	printf("\n 1. Ubaci");
	printf("\n 2. Izbaci");
	printf("\n 3. Promeni");
	printf("\n 4. Prikazi");
	int iz;
	do {
		printf("\n\nVas izbor (0 - 4): ");
		scanf("%d", &iz);
	} while ((iz < 0) || (iz > 4));
	return iz;
}

void Izvrsi (const int iz) {
	switch (iz) {
		case 0:	Odjava();	break;
		case 1:	UbaciStud();	break;
		case 2:	IzbaciStud();	break;
		case 3:	PromeniStud();	break;
		case 4:	PrikaziStud();	break;
		default: printf("\nUneli ste neodgovarajuci broj.");
	}
}
```

```
void Odjava(void) {
	printf("\nKRAJ RADA\n\n");
}

void UbaciStud(void) {
	printf("\n\nRadi se... UBACI\n");
	getch();
}

void IzbaciStud(void) {
	printf("\n\nRadi se... IZBACI\n");
	getch();
}

void PromeniStud(void) {
	printf("\n\nRadi se... PROMENI\n");
	getch();
}

void PrikaziStud(void) {
	int izborPrikaza;
	do {
		izborPrikaza = PrikaziMeni();
		IzvrsiPrikaz(izborPrikaza);
	} while (izborPrikaza != KRAJ);
}

int PrikaziMeni(void) {
	clrscr();
	printf("\n======================\nSTUDENT => PRIKAZI\n======================");
	printf("\n M E N I\n");
	printf("\n 0. Kraj\n");
	printf("\n 1. Prikazi sve");
	printf("\n 2. Prikazi po broju indeksa");
	printf("\n 3. Prikazi po prezimenu");
	int iz;
	do {
		printf("\n\nVas izbor (0 - 3): ");
		scanf("%d", &iz);
	} while ((iz < 0) || (iz > 3));
	return iz;
}

void IzvrsiPrikaz (const int iz) {
	switch (iz) {
		case 0:	break;
		case 1:	PrikaziSve();  break;
		case 2:	PrikaziPoBI(); break;
		case 3:	PrikaziPoPrez();break;
		default: printf("\nUneli ste neodgovarajuci broj.");
	}
}

void PrikaziSve(void) {
	printf("\n\nRadi se... PRIKAZI SVE\n");
	getch();
}
```

```
void PrikaziPoBI(void) {
	printf("\n\nRadi se... PRIKAZI PO BROJU INDEKSA\n");
	getch();
}

void PrikaziPoPrez(void) {
	printf("\n\nRadi se... PRIKAZI PO PREZIMENU\n");
	getch();
}
```

Верзија 2

Датотека: *meni_02.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : meni_02.c
** OPIS     : Napisati program u programskom jeziku C koji implementira
**            konzolni korisnički interfejs upotrebom menija.
**            V.2
** DATUM    : 11.03.2019.
** AUTOR    : S.D.L.
** PROMENE  :
**  12.03.2019. - dodat podmeni PRIKAZI (SDL)
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2019.
*/

/*

Implementirati sledeći meni:

===============
 S T U D E N T
===============

1. Ubaci     (Insert)
2. Izbaci    (Delete)
3. Promeni   (Update)
4. Prikazi   (Select)
	1. Prikazi sve
	2. Prikazi po broju indeksa
	3. Prikazi po prezimenu
	4. Kraj (tj. povratak u nadmeni)
5. Kraj rada

*/

#include <stdio.h>
#include <stdlib.h>

#define KRAJ  0
#define RADI  1

void glavni_meni(char* stavke, int br_stavki);
void prikazi_meni(char* stavke, int br_stavki);
int  get_int(char* prompt, int donja_granica, int gornja_granica);
int  izvrsi_gm(int izbor);
int  izvrsi_pr(int izbor);
```

```
char* stavke_gl =                       // Stavke glavnog menija
   "\n===============\n "
   "S T U D E N T"
   "\n===============\n\n"
   " GLAVNI MENI\n\n"
   " 1. Ubaci\n"
   " 2. Izbaci\n"
   " 3. Promeni\n"
   " 4. Prikazi...\n"
   " 5. Kraj rada\n";

char* stavke_pr =                       // Stavke menija Prikazi
   "\n===============\n "
   "S T U D E N T"
   "\n===============\n\n"
   " PRIKAZI\n\n"
   " 1. Prikazi sve\n"
   " 2. Prikazi po broju indeksa\n"
   " 3. Prikazi po prezimenu\n"
   " 4. Kraj\n";

int main(void) {
       glavni_meni(stavke_gl, 5);
       return 0;
}

void glavni_meni(char* stavke, int br_stavki) {
       int izbor = -1;
       char prompt[20];
       sprintf(prompt, "\nVas izbor [1..%d]: ", br_stavki);
       do {
              fputs(stavke, stdout);
              izbor = get_int(prompt, 1, br_stavki);
       } while (izvrsi_gm(izbor) != KRAJ);
}
void prikazi_meni(char* stavke, int br_stavki) {
       int izbor = -1;
       char prompt[20];
       sprintf(prompt, "\nVas izbor [1..%d]: ", br_stavki);
       do {
              fputs(stavke, stdout);
              izbor = get_int(prompt, 1, br_stavki);  //
       } while (izvrsi_pr(izbor) != KRAJ);
}

int izvrsi_gm(int izbor) {
      int signal = RADI;
      system("cls");
      switch (izbor) {
              case 1:  puts("\nIzvrsavam UBACI"); system("pause"); break;
              case 2:  puts("\nIzvrsavam IZBACI"); system("pause"); break;
              case 3:  puts("\nIzvrsavam PROMENI"); system("pause"); break;
              case 4:  prikazi_meni(stavke_pr, 4); break;
              case 5:  puts("\n\nKraj rada"); signal = KRAJ; break;
              default: puts("\n\nGRESKA");
       }
       return signal;
}
```

```
int izvrsi_pr(int izbor) {
	int signal = RADI;
	system("cls");
	switch (izbor) {
		case 1:  puts("\nIzvrsavam PRIKAZI SVE"); system("pause"); break;
		case 2:  puts("\nIzvrsavam PRIKAZI PO BROJU INDEKSA"); system("pause");
break;
		case 3:  puts("\nIzvrsavam PRIKAZI PO PREZIMENU"); system("pause"); break;
		case 4:  signal = KRAJ; break;
		default: puts("\n\nGRESKA");
	}
	return signal;
}

int get_int(char* prompt, int dg, int gg) {
	char string[99];
	int number;
	do {
		do {
			fputs(prompt, stdout);
			fgets(string, 99, stdin);
		} while(!(number = atoi(string)));
	} while (number < dg || number > gg);
	return number;
}
```

Верзија 3

Датотека: *meni_03.c*

```
/*
** PROJEKAT : Osnove programiranja u programskom jeziku C
** DATOTEKA : meni_03.c
** OPIS     : Napisati program u programskom jeziku C koji implementira
**            konzolni korisnički interfejs upotrebom menija.
**            V.3
** DATUM    : 17.03.2019.
** AUTOR    : S.D.L.
** PROMENE  :
**  18.03.2019. - xxx (SDL)
**  xx.xx.xxxx. - <opis promene> (<programer>)
**
** Copyright (C) FON-LSI, 2019.
*/

/*****************************************

Имплементирати следећи мени:

===============
 S T U D E N T
===============

1. Ubaci     (Insert)
2. Izbaci    (Delete)
3. Promeni   (Update)
4. Prikazi   (Select)
	1. Prikazi sve
	2. Prikazi po broju indeksa
	3. Prikazi po prezimenu
```

```
      4. Kraj (tj. povratak u nadmeni)
5. Kraj rada

*******************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MBS 5// максималан број ставки у менију

typedef enum { ERR = -2, WRN, KRAJ, OK } signal_t;  // сигнал успешности извршења
наредбе
typedef signal_t func_t(void);         // функција која извршава наредбу; без
параметара, враћа сигнал
typedef func_t* naredba_t;
typedef struct {
      int       br_stavki;      // број ставки у менију
      char *    naslov;         // наслов менија
      char *    tekst;          // текст који садржи ставке менија
      naredba_t naredba[MBS];   // наредбе
      } meni_t;

// Функције за реализацију корисничког интерфејса
signal_t izvrsi(meni_t meni);
int izaberi_naredbu(meni_t meni);
int get_int(char* prompt, int dg, int gg);  // dg = доња граница, gg = горња граница

// Функција за обраду сигнал успешности извршења наредбе
void obradi_signal(signal_t signal);

// Тип и функција за обраду упозорења и грешака
typedef enum { GR_IZZ, GR_NULL, PRAZNO_PREZ, NEMA_PREZ, PRAZAN_BI, NEMA_BI }
izuzetak_t;
void obradi_izuzetak(izuzetak_t izuzetak);

// Наредбе Главног менија
signal_t ubaci(void);
signal_t izbaci(void);
signal_t promeni(void);
signal_t prikazi(void);
signal_t kraj(void);

// Наредбе менија Прикази
signal_t prikazi_sve(void);
signal_t prikazi_po_broju_indeksa(void);
signal_t prikazi_po_prezimenu(void);
signal_t kraj_prikazi(void);

meni_t glavni_meni = {
      .br_stavki = 5,
      .naslov = "\n===============\n S T U D E N T\n===============\n\n G L A V N I   M
E N I\n",
      .tekst = " 1. Ubaci\n 2. Izbaci\n 3. Promeni\n 4. Prikazi...\n 5. Kraj rada\n",
      .naredba = { ubaci, izbaci, promeni, prikazi, kraj }
};

meni_t prikazi_meni = {
      .br_stavki = 4,
```

```
	.naslov = "\n===============\n S T U D E N T\n===============\n\n P R I K A Z
I\n",
	.tekst = " 1. Prikazi sve\n 2. Prikazi po broju indeksa\n 3. Prikazi po
prezimenu\n 4. Kraj\n",
	.naredba = { prikazi_sve, prikazi_po_broju_indeksa, prikazi_po_prezimenu,
kraj_prikazi }
};

int main(void) {
	while (izvrsi(glavni_meni) != KRAJ);
	return EXIT_SUCCESS;
}

signal_t izvrsi(meni_t meni) {
	int izbor = izaberi_naredbu(meni);
	signal_t sig = meni.naredba[izbor]();  // изврши наредбу
	obradi_signal(sig);
	return sig;
}

int izaberi_naredbu(meni_t meni) {
	system("cls");
	puts(meni.naslov);
	puts(meni.tekst);
	char prompt[22] = " Vas izbor [1..%d] -> ";
	snprintf(prompt, 22, prompt, meni.br_stavki);  // прилагоди промпт за приказ
	int i = get_int(prompt, 1, meni.br_stavki);    // изабери наредбу из менија
	return (--i);
}

void obradi_signal(signal_t signal) {
	switch (signal) {
	    case ERR:  fputs("\n>>> GRESKA: program ce biti prekinut.\n", stderr);
exit(EXIT_FAILURE);
		case WRN:
		case OK:   system("pause"); break;
		case KRAJ: break;
		default:   fputs("ERROR: izvrsi()", stderr);  // Ово не би требало да се
изврши. Никада.
	}
}

void obradi_izuzetak(izuzetak_t izuzetak) {
	// Изузетак = упозорење или грешка
	// GR_IZZ, GR_NULL, PRAZNO_PREZ, NEMA_PREZ, PRAZAN_BI, NEMA_BI
	char * poruka[] = {
		"\n>>> GRESKA: nepoznat izuzetak. ",
		"\n>>> GRESKA: vracena je NULL vrednost. ",
		"\n::: Nedostaje prezime. ",
		"\n::: Nema studenata sa navedenim prezimenom. ",
		"\n::: Nedostaje broj indeksa. ",
		"\n::: Nema studenta sa navedenim BI. "
	};
	unsigned int broj_poruka = sizeof(poruka) / sizeof(char *);
	if (izuzetak >= (izuzetak_t)broj_poruka) izuzetak = 0;
	fputs(poruka[izuzetak], stdout);
}

int get_int(char* prompt, int dg, int gg) {
```

```
	char string[99];
	int number;
	do {
		do {
			fputs(prompt, stdout);
			fgets(string, 99, stdin);
		} while(!(number = atoi(string)));
	} while (number < dg || number > gg);
	return number;
}

signal_t ubaci(void)   { puts("\nIzvrsavam ubaci()");  return OK; }
signal_t izbaci(void)  { puts("\nIzvrsavam izbaci()"); return OK; }
signal_t promeni(void) { puts("\nIzvrsavam promeni()"); return OK; }
signal_t prikazi(void) { while (izvrsi(prikazi_meni) != KRAJ); return OK; };
signal_t kraj(void)    { fputs("\nKRAJ RADA\nCopyright (C) FON-LSI, 2019.\n", stdout);
return KRAJ; };

signal_t prikazi_sve(void)              { puts("\nIzvrsavam prikazi_sve()"); return OK;
}
signal_t prikazi_po_broju_indeksa(void) { puts("\nIzvrsavam
prikazi_po_broju_indeksa()"); return OK; }
signal_t kraj_prikazi(void) { return KRAJ; }

signal_t prikazi_po_prezimenu(void) {  // пример имплементације наредбе из менија
	signal_t signal = OK;
	char prezime[99];
	system("cls");
	fputs("\nPRIKAZI PO PREZIMENU\n\nUnesite prezime: ", stdout);
	if (gets_s(prezime, 98) == NULL) {
		obradi_izuzetak(GR_NULL);
		signal = ERR;
	}
	else if (strlen(prezime) == 0) {
		obradi_izuzetak(PRAZNO_PREZ);
		signal = WRN;
	}
	else if (strncmp(prezime, "Jovanovic", strlen(prezime)) == 0) {
		fputs("\n\tPREZIME\t\tIME\n\t------------------------\n", stdout);
		fputs("\tJovanovic\tAna\n\tJovanovic\tEma\n\tJovanovic\tMiodrag\n\n",
stdout);
	}
	else {
		obradi_izuzetak(NEMA_PREZ);
		signal = WRN;
	}
	return signal;
}
```