Programiranje 2

Laboratorijske vježbe

LV4
Pokazivači i strukture

Fakultet elektrotehnike računarstva i informacijskih tehnologija Osijek
Kneza Trpimira 2b
www.ferit.unios.hr

Uvod

Za grupiranja različitih tipova podataka pod jednim imenom koristi se struktura. Struktura omogućava kreiranje novog korisnički kreiranog složenog tipa podatka s kojim se lakše može opisati neki objekt. Varijabla strukture predstavlja konkretnu realizaciju strukture koja zauzima memorijski prostor i s njom se radi u programskom kôd-u. Pokazivači su varijable u koje se može spremiti samo jedna memorijska adresa varijable određenog tipa. Pokazivači omogućavaju indirektni pristup sadržaju varijable na koju su usmjereni, odnosno omogućavaju pristup sadržaju koji se nalazi u memorijskoj adresi koja je pohranjena u sam pokazivač. Na ovi laboratorijskim vježbama, pokazat će se rad s pokazivačima i strukturama.

Pokazivač na strukturu

Kao što se može deklarirati pokazivač na bilo koji primitivni tip podatka, tako se može deklarirati pokazivač na korisnički kreirani složeni tip podatka kao što je struktura. Potrebno je obratiti pozornost na tip strukture, jer tip pokazivača mora odgovarati tipu određene varijable strukture na koju će pokazivač biti usmjereni. Pokazivač na strukturu sadrži memorijsku adresu varijable strukture na koju pokazuje. Prilikom deklariranja ako pokazivač nije inicijaliziran na određenu memorijsku adresu, poželjno je pokazivač inicijalizirati na *NULL* vrijednost.

Primjer 1: Kreiranje pokazivača na strukturu, te njegovo usmjeravanje na varijablu strukture.

```
typedef struct datum {
	unsigned short dan;
	unsigned short mjesec;
	unsigned short godina;
}DATUM;

typedef struct student {
	char ime[20];
	char prezime[20];
	char indeks[10];
	float prosjek;
	DATUM datumRodjenja;
}STUDENT;

int main(void) {

	STUDENT najStudent = {"Ivan", "Ivic", "10879", 4.56f, 5, 5, 1998};
	//STUDENT *pokS = NULL;
	//pokS = &najStudent;
	STUDENT *pokS = &najStudent;

	return 0;
}
```

U pokazivač ***pokS*** na strukturu tipa `STUDENT`, pridružena je memorijska adresa varijable strukture ***najStudent***. Pomoću adresnog operatora (&) dohvatila se memorijska adresa strukture ***najStudent*** i pomoću operatora pridruživanja (=) pridružila se memorijska adresa pokazivaču na strukturu ***pokS***.

Pristup članovima strukture pomoću pokazivača na strukturu

Pristup članovima strukture pomoću pokazivača na strukturu moguće je na dva načina:

1. Primjenom operatora dereferenciranja (*) i operatora točke (.).
2. Primjenom operatora strjelice (->).

Primjenom prvog načina pristupa potrebno je preko pokazivača na strukturu dohvatiti varijablu strukture na koju je pokazivač usmjeren i to upotrebom operator dereferenciranja (*), a kako bi se pristupilo pojedinom članu strukture, potrebno je koristi operator točka (.).

```
printf("Ime: %s\n", (*pokS).ime);
```

Kako operator točka ima veći prioritet od operatora dereferenciranja, potrebno je koristiti zagrade unutar kojih se prvo vrši pristup varijabli strukture preko pokazivača, a nakon toga se operatorom točka dohvaća član strukture.

Primjenom drugog načina olakšava se pristup varijabli strukture i njezinim članovima preko pokazivača uvođenjem novog operator, strelice (->). Operator strelica vrši dereferenciranje i dohvaćanje člana, drugim riječima zamjenjuje operator dereferenciranja (*) i operator (.). Koristi se između imena pokazivača na strukturu i određenog člana te strukture kojeg se treba dohvatiti. Operator strelica je uveden zbog preglednosti i semantički jasnijeg zapisa jer pristup elementima strukture preko pokazivača primjenom operatora dereferenciranja i operatora točke vrlo brzo može postati nepregledno.

```
printf("Ime: %s\n", pokS->ime);
```

Primjer 2: Pristup varijabli strukture i njezinim članovima preko pokazivača, primjenom operatora dereferenciranja i točke, te primjenom operatora strelice.

```c
#include<stdio.h>

typedef struct datum {
	unsigned short dan;
	unsigned short mjesec;
	unsigned short godina;
}DATUM;

typedef struct student {
	char ime[20];
	char prezime[20];
	char indeks[10];
	float prosjek;
	DATUM datumRodjenja;
}STUDENT;

int main() {

	STUDENT najStudent;
	STUDENT *pokS = &najStudent;

	printf("Unesite ime\n");
	scanf("%19s", (*pokS).ime);
	//scanf("%19s", pokS->ime);
	printf("Unesite prezime\n");
	scanf("%19s", (*pokS).prezime);
	//scanf("%19s", pokS->prezime);
	printf("Unesite indeks studenta\n");
	scanf("%9s", (*pokS).indeks);
	//scanf("%9s", pokS->indeks);
	printf("Unesite prosjek studenta\n");
	scanf("%f", &(*pokS).prosjek);
	//scanf("%f", &pokS->prosjek);
	printf("Unesite dan rodjendan studenta\n");
	scanf("%hu", &(*pokS).datumRodjenja.dan);
	//scanf("%hu", &pokS->datumRodjenja.dan);
	printf("Unesite mjesec rodjendan studenta\n");
	scanf("%hu", &(*pokS).datumRodjenja.mjesec);
	//scanf("%hu", &pokS->datumRodjenja.mjesec);
	printf("Unesite godinu rodjendan studenta\n");
	scanf("%hu", &(*pokS).datumRodjenja.godina);
	//scanf("%hu", &pokS->datumRodjenja.godina);

	printf("Ime: %s\nPrezime: %s\nIndeks: %s\nProsjek: %.2f\nDatum rodenja:\
	%02hu.%02hu.%4hu.\n", (*pokS).ime, (*pokS).prezime, (*pokS).indeks,
	(*pokS).prosjek, (*pokS).datumRodjenja.dan, (*pokS).datumRodjenja.mjesec,
	(*pokS).datumRodjenja.godina) /*pokS->ime, pokS->prezime, pokS->indeks,
	pokS->prosjek, pokS->datumRodjenja.dan, pokS->datumRodjenja.mjesec,
	pokS->datumRodjenja.godina)*/;

	return 0;
}
```

Pokazivač na strukturu i polje struktura

Notacija polja zajedno s pokazivačkom notacijom kod koje se može koristiti operator dereferenciranja i točka ili strelica, može se primijeniti nad poljem struktura, kao i nad pokazivačem na strukturu koji je usmjeren na prvi element polja struktura. Sljedećim primjerima pokazat će se primjena različitih notacija.

Primjer 3: Primjena notacije polja s poljem struktura i pokazivačem na prvi element polja struktura.

```
#include<stdio.h>

typedef struct datum {
	unsigned short dan;
	unsigned short mjesec;
	unsigned short godina;
}DATUM;

typedef struct student {
	char ime[20];
	char prezime[20];
	char indeks[10];
	float prosjek;
	DATUM datumRodjenja;
}STUDENT;

int main() {

	STUDENT studenti[3] = { 0 };
	STUDENT *pokS = studenti;

	for (int i = 0; i < 3; i++)
	{
		printf("Ime %d. studenta: %s\n", i + 1, studenti[i].ime);
		printf("Prezime %d. studenta: %s\n", i + 1, studenti[i].prezime);
		printf("Indeks %d. studenta: %s\n", i + 1, studenti[i].indeks);
		printf("Prosjek %d. studenta: %.2f\n", i + 1, studenti[i].prosjek);
		printf("Datum rodenja %d. studenta: %02hu.%02hu.%4hu.\n", i + 1,
		studenti[i].datumRodjenja.dan, studenti[i].datumRodjenja.mjesec,
		studenti[i].datumRodjenja.godina);
	}

	for (int i = 0; i < 3; i++)
	{
		printf("Ime %d. studenta: %s\n", i + 1, pokS[i].ime);
		printf("Prezime %d. studenta: %s\n", i + 1, pokS[i].prezime);
		printf("Indeks %d. studenta: %s\n", i + 1, pokS[i].indeks);
		printf("Prosjek %d. studenta: %.2f\n", i + 1, pokS[i].prosjek);
		printf("Datum rodenja %d. studenta: %02hu.%02hu.%4hu.\n", i + 1,
		pokS[i].datumRodjenja.dan, pokS[i].datumRodjenja.mjesec,
		pokS[i].datumRodjenja.godina);
	}

	return 0;
}
```

Primjena notacije polja, uglate zagrada vrše dereferenciranje, a primjenom cjelobrojnog indeksa pristupa se pojedinoj strukturi unutar polja struktura. Elementima strukture pristupa se operatorom točka.

Primjer 4: Primjena pokazivačke notacije s poljem struktura i pokazivačem na prvi element polja struktura.

```c
#include<stdio.h>

typedef struct datum {
	unsigned short dan;
	unsigned short mjesec;
	unsigned short godina;
}DATUM;

typedef struct student {
	char ime[20];
	char prezime[20];
	char indeks[10];
	float prosjek;
	DATUM datumRodjenja;
}STUDENT;

int main() {

	STUDENT studenti[3] = { 0 };
	STUDENT *pokS = studenti;

	for (int i = 0; i < 3; i++)
	{
		printf("Ime %d. studenta: %s\n", i + 1, (*(studenti + i)).ime);
		printf("Prezime %d. studenta: %s\n", i + 1,
		(*(studenti + i)).prezime);
		printf("Indeks %d. studenta: %s\n", i + 1, (*(studenti + i)).indeks);
		printf("Prosjek %d. studenta: %.2f\n", i + 1,
		(*(studenti + i)).prosjek);
		printf("Datum rodenja %d. studenta: %02hu.%02hu.%4hu.\n", i + 1,
		(*(studenti + i)).datumRodjenja.dan,
		(*(studenti + i)).datumRodjenja.mjesec,
		(*(studenti + i)).datumRodjenja.godina);
	}

	for (int i = 0; i < 3; i++)
	{
		printf("Ime %d. studenta: %s\n", i + 1, (*(pokS + i)).ime);
		printf("Prezime %d. studenta: %s\n", i + 1, (*(pokS + i)).prezime);
		printf("Indeks %d. studenta: %s\n", i + 1, (*(pokS + i)).indeks);
		printf("Prosjek %d. studenta: %.2f\n", i + 1, (*(pokS + i)).prosjek);
		printf("Datum rodenja %d. studenta: %02hu.%02hu.%4hu.\n", i + 1,
		(*(pokS + i)).datumRodjenja.dan, (*(pokS + i)).datumRodjenja.mjesec,
		(*(pokS + i)).datumRodjenja.godina);
	}

	return 0;
}
```

Primjena pokazivačke notacije, u izrazu gdje se zbraja ime polja/pokazivača s cjelobrojnim indeksom postiže se pomak od početnog elementa, odnosno pristupa se memorijskoj adresi pojedine strukture. Operatorom zvjezdica vrši se dereferenciranje, odnosno pristupa se konkretnoj strukturi na određenom indeksu, a operatorom točka pristupa se pojedinom elementu strukture.

Primjer 5: Primjena pokazivačke notacije s poljem struktura i pokazivačem na prvi element polja struktura.

```c
#include<stdio.h>

typedef struct datum {
	unsigned short dan;
	unsigned short mjesec;
	unsigned short godina;
}DATUM;

typedef struct student {
	char ime[20];
	char prezime[20];
	char indeks[10];
	float prosjek;
	DATUM datumRodjenja;
}STUDENT;

int main() {

	STUDENT studenti[3] = { 0 };
	STUDENT *pokS = studenti;

	for (int i = 0; i < 3; i++)
	{
		printf("Ime %d. studenta: %s\n", i + 1, (studenti + i)->ime);
		printf("Prezime %d. studenta: %s\n", i + 1, (studenti + i)->prezime);
		printf("Indeks %d. studenta: %s\n", i + 1, (studenti + i)->indeks);
		printf("Prosjek %d. studenta: %.2f\n", i + 1,
		(studenti + i)->prosjek);
		printf("Datum rodenja %d. studenta: %02hu.%02hu.%4hu.\n", i + 1,
		(studenti + i)->datumRodjenja.dan,
		(studenti + i)->datumRodjenja.mjesec,
		(studenti + i)->datumRodjenja.godina);
	}

	for (int i = 0; i < 3; i++)
	{
		printf("Ime %d. studenta: %s\n", i + 1, (pokS + i)->ime);
		printf("Prezime %d. studenta: %s\n", i + 1, (pokS + i)->prezime);
		printf("Indeks %d. studenta: %s\n", i + 1, (pokS + i)->indeks);
		printf("Prosjek %d. studenta: %.2f\n", i + 1, (pokS + i)->prosjek);
		printf("Datum rodenja %d. studenta: %02hu.%02hu.%4hu.\n", i + 1,
		(pokS + i)->datumRodjenja.dan, (pokS + i)->datumRodjenja.mjesec,
		(pokS + i)->datumRodjenja.godina);
	}

	return 0;
}
```

Primjena pokazivačke notacije, u izrazu gdje se zbraja ime polja/pokazivača s cjelobrojnim indeksom postiže se pomak od početnog elementa, odnosno pristupa se memorijskoj adresi pojedine strukture. Operatorom strelica vrši se dereferenciranje, odnosno pristupa se konkretnoj strukturi na određenom indeksu i ujedno se dohvaća element strukture.

Pokazivač kao član strukture

Vrlo je praktično imati pokazivač kao član strukture u situacijama gdje je nepoznata duljina polja, tada se može dinamički zauzeti memorijski prostor i usmjeriti pokazivač na blok memorije.

Primjer 6: Dinamičko zauzimanje memorije pomoću funkcije *unos()* za string egzaktne duljine, te oslobađanje memorije pomoću funkcije *oslobadjanje()*.

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

void unos(char **);
void oslobadjanje(char **);

typedef struct datum {
	unsigned short dan;
	unsigned short mjesec;
	unsigned short godina;
}DATUM;
typedef struct student {
	char *ime;
	char *prezime;
	char *indeks;
	float prosjek;
	DATUM datumRodjenja;
}STUDENT;

int main() {

	STUDENT najStudent = { 0 };

	printf("Unesite ime\n");
	unos(&najStudent.ime);
	if(najStudent.ime == NULL){
		return 1;
	}
	printf("Unesite prezime\n");
	unos(&najStudent.prezime);
	if(najStudent.prezime == NULL){
		return 1;
	}
	printf("Unesite indeks studenta\n");
	unos(&najStudent.indeks);
	if(najStudent.indeks == NULL){
		return 1;
	}
	printf("Unesite prosjek studenta\n");
	scanf("%f", &najStudent.prosjek);
	printf("Unesite dan rodjendan studenta\n");
	scanf("%hu", &najStudent.datumRodjenja.dan);
	printf("Unesite mjesec rodjendan studenta\n");
	scanf("%hu", &najStudent.datumRodjenja.mjesec);
	printf("Unesite godinu rodjendan studenta\n");
	scanf("%hu", &najStudent.datumRodjenja.godina);
```

```
        printf("Ime: %s\nPrezime: %s\nIndeks: %s\nProsjek: %.2f\nDatum rodenja:\
        %02hu.%02hu.%4hu.\n", najStudent.ime, najStudent.prezime, najStudent.indeks,
        najStudent.prosjek, najStudent.datumRodjenja.dan,
        najStudent.datumRodjenja.mjesec, najStudent.datumRodjenja.godina);

        oslobadjanje(&najStudent.ime);
        oslobadjanje(&najStudent.prezime);
        oslobadjanje(&najStudent.indeks);

        return 0;
}

void unos(char **pok) {

        char pomocnoPolje[50] = { 0 };
        int duljina = 0;

        scanf("%49[^\n]", pomocnoPolje);
        duljina = strlen(pomocnoPolje);

        *pok = (char*)calloc(duljina + 1, sizeof(char));
        if(*pok == NULL){
                return;
        }

        //*pok = (char*)malloc((duljina + 1) * sizeof(char));
        //if(*pok == NULL){
        //      return;
        //}
        //*(*pok + duljina) = '\0';

        strcpy(*pok, pomocnoPolje);
}

void oslobadjanje(char **pok) {

        free(*pok);
        *pok = NULL;
}
```

Funkcija *unos()* prima memorijsku adresu jednostrukog pokazivača koji je član strukture STUDENT iz razloga kako bi mogla promijeniti, odnosno postaviti sadržaj pokazivača na memorijsku adresu prvog elementa polja za koji se dinamički zauzela memorija. Unutar funkcije *unos()* koristi se pomoćno polje određene duljine u koje se privremeno sprema string određenog sadržaja. Nakon toga se određuje stvarna duljina stringa funkcijom *strlen()* iz standardne biblioteke, koja živi u zaglavlju *<string.h>.* Zatim se dinamički zauzima memorijski prostor za pravu duljini stringa, te se pomoću funkcije *strcpy()* iz standardne biblioteke koja živi u zaglavlju *<string.h>* kopira sadržaj stringa iz pomoćnog polja u dio memorije koji se dinamički zauzeo te odgovara duljini stringa. Potrebno je napomenuti kako se dvostruki pokazivač mora jedanput dereferencirati da bi se dohvatio jednostruki pokazivač na koji pokazuje, te na taj način ažurirao sadržaj u jednostrukom pokazivaču.

Funkcija *oslobadjanje()* prima memorijsku adresu jednostrukog pokazivača, istom namjerom kao i funkcija *unos()* kako bi se unutar funkcije *oslobadjanje()* promijenio sadržaj jednostrukog pokazivača koji je predan toj funkciji. Unutar funkcije se poziva funkcija *free()* te joj se predaje jednostruki pokazivač i u konačnici oslobodila memorija. Nakon toga se jednostrukom pokazivaču postavlja sadržaj na *NULL* memorijsku adresu.

Pokazivač na strukturu kao član strukture

Primjer 7: Primjena pokazivača na strukturu kao člana strukture.

```c
#include <stdio.h>

int odabirIndeksa(int);

typedef struct tocka {
	float x;
	float y;
}TOCKA;

typedef struct trokut {
	TOCKA *v1;
	TOCKA *v2;
	TOCKA *v3;
}TROKUT;

int main(void) {

	TOCKA tocke[9];
	TROKUT trokuti[3];

	for (int i = 0; i < 9; i++)
	{
		printf("Unesite koordinate x, y za tocku %d\n", i + 1);
		scanf("%f%f", &tocke[i].x, &tocke[i].y);
	}
	for (int i = 0; i < 3; i++)
	{
		trokuti[i].v1 = &tocke[odabirIndeksa(i)];
		trokuti[i].v2 = &tocke[odabirIndeksa(i)];
		trokuti[i].v3 = &tocke[odabirIndeksa(i)];
	}
	for (int i = 0; i < 3; i++)
	{
		printf("Trokut %d ima sljedce koordinate\n", i + 1);
		printf("\nVrh 1:\nx: %f\ny: %f\n",
		trokuti[i].v1->x, trokuti[i].v1->y);
		printf("\nVrh 2:\nx: %f\ny: %f\n",
		trokuti[i].v2->x, trokuti[i].v2->y);
		printf("\nVrh 3:\nx: %f\ny: %f\n",
		trokuti[i].v3->x, trokuti[i].v3->y);
	}

	return 0;
}
```

```
int odabirIndeksa(int i) {

	int indeks;
	do {
		printf("Odaberite koordinate od 1 do 9 za vrh trokuta %d\n", (i + 1));
		scanf("%d", &indeks);
		if (indeks < 1 || indeks > 9) {
			printf("Odabrali ste krivu koordinatu, pokusajte ponovno\n");
		}
	} while (indeks < 1 || indeks > 9);

	return indeks - 1;
}
```

U ovom primjeru, važno je primijetiti kako su članovi strukture TROKUT tri pokazivača na strukturu TOCKA, a članovi strukture TOCKA su dva realna člana koji opisuju tocku u dvodimenzionalnom sustavu s dvije koordinate. U glavnom dijelu programa, funkciji *main()*, kreirano je polje trokutova i tocaka. Pristupa se svakom trokutu individualno, kao i članovima pojedine strukture TROKUT te se pojedini pokazivač na TOCKU usmjerava na jednu strukturu TOCKA iz polja tocaka. Usmjeravanje, odnosno odabir pojedine strukture TOCKA je omogućen pomoću funkcije *odabirIndeksa()*.

Ovim načinom se omogućilo da u jednom dijelu memorije budu sve koordinate točaka, odnosno strukture TOCKA koje opisuju točku, dok se trokut tvori pomoću tri pokazivača na strukturu TOCKA koji su usmjereni na pojedine strukture TOCKA. Ako bi se promijenila jedna koordinata ili njih nekoliko koje opisuju točku pomoću strukture TOCKA, automatski bi se izgled trokuta promijenio jer su članovi strukture TROKUT pokazivači na strukturu TOCKA. Potrebno je naglasiti kako se stvarni podaci, a to je polje točaka, nalaze na jednom mjestu u memoriji. Ovim načinom se dobilo ekonomično upravljanje s memorijom i uklonjeno je dupliciranje podataka koje bi nastalo da se umjesto pokazivača na strukturu TOCKA koriste članovi koji su varijable strukture TOCKA.

Dinamičko zauzimanje memorije za strukturu

Vrlo često je potreba za racionalnim upravljanjem memorijom, stoga se pristupa dinamičkom zauzimanju memorije. Kao i za bilo koji tip podatka, moguće je dinamički zauzeti memorijski prostor i za strukturu.

Primjer 8: Dinamičko zauzimanje memorije za jednu strukturnu varijablu.

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

typedef struct datum {
	unsigned short dan;
	unsigned short mjesec;
	unsigned short godina;
}DATUM;

typedef struct student {
	char ime[20];
	char prezime[20];
	char indeks[10];
	float prosjek;
	DATUM datumRodjenja;
}STUDENT;

int main() {

	STUDENT *pokNaStudenta = NULL;

	pokNaStudenta = (STUDENT*)calloc(1, sizeof(STUDENT));
	if(pokNaStudenta == NULL){
		return 1;
	}
	printf("Unesite ime\n");
	scanf("%19s", pokNaStudenta->ime);
	printf("Unesite prezime\n");
	scanf("%19s", pokNaStudenta->prezime);
	printf("Unesite indeks studenta\n");
	scanf("%9s", pokNaStudenta->indeks);
	printf("Unesite prosjek studenta\n");
	scanf("%f", &pokNaStudenta->prosjek);
	printf("Unesite dan rodjendan studenta\n");
	scanf("%hu", &pokNaStudenta->datumRodjenja.dan);
	printf("Unesite mjesec rodjendan studenta\n");
	scanf("%hu", &pokNaStudenta->datumRodjenja.mjesec);
	printf("Unesite godinu rodjendan studenta\n");
	scanf("%hu", &pokNaStudenta->datumRodjenja.godina);

	printf("Ime: %s\nPrezime: %s\nIndeks: %s\nProsjek: %.2f\nDatum rodenja:\
	%02hu.%02hu.%4hu.\n", pokNaStudenta->ime, pokNaStudenta->prezime,
	pokNaStudenta->indeks, pokNaStudenta->prosjek,
	pokNaStudenta->datumRodjenja.dan, pokNaStudenta->datumRodjenja.mjesec,
	pokNaStudenta->datumRodjenja.godina);

	free(pokNaStudenta);

	return 0;
}
```

Primjer 9: Dinamičko zauzimanje memorije za polje struktura.

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

typedef struct datum {
	unsigned short dan;
	unsigned short mjesec;
	unsigned short godina;
}DATUM;

typedef struct student {
	char ime[20];
	char prezime[20];
	char indeks[10];
	float prosjek;
	DATUM datumRodjenja;
}STUDENT;

int main() {

	STUDENT *pokNaStudente = NULL;

	pokNaStudente = (STUDENT*)calloc(3, sizeof(STUDENT));
	if(pokNaStudente == NULL){
		return 1;
	}

	for (int i = 0; i < 3; i++)
	{
		printf("Unesite ime\n");
		scanf("%19s", (pokNaStudente + i)->ime);
		printf("Unesite prezime\n");
		scanf("%19s", (pokNaStudente + i)->prezime);
		printf("Unesite indeks studenta\n");
		scanf("%9s", (pokNaStudente + i)->indeks);
		printf("Unesite prosjek studenta\n");
		scanf("%f", &(pokNaStudente + i)->prosjek);
		printf("Unesite dan rodjendan studenta\n");
		scanf("%hu", &(pokNaStudente + i)->datumRodjenja.dan);
		printf("Unesite mjesec rodjendan studenta\n");
		scanf("%hu", &(pokNaStudente + i)->datumRodjenja.mjesec);
		printf("Unesite godinu rodjendan studenta\n");
		scanf("%hu", &(pokNaStudente + i)->datumRodjenja.godina);

		printf("Ime: %s\nPrezime: %s\nIndeks: %s\nProsjek: %.2f\nDatum\
		rodenja: %02hu.%02hu.%4hu.\n", (pokNaStudente + i)->ime,
		(pokNaStudente + i)->prezime, (pokNaStudente + i)->indeks,
		(pokNaStudente + i)->prosjek, (pokNaStudente + i)->datumRodjenja.dan,
		(pokNaStudente + i)->datumRodjenja.mjesec,
		(pokNaStudente + i)->datumRodjenja.godina);
	}

	free(pokNaStudente);

	return 0;
}
```

Zadaci

1 Napisati C program koji omogućuje unos dva broja *m* i *n*, gdje je *m* $(3 \leq m \leq 30)$, a *n* $(1 \leq n < 11)$. Omogućiti unos *m* točaka i *n* trokuta. Dinamički zauzeti memoriju za polje točaka i polje trokuta (u potpunosti rukovati memorijom). Pronaći i ispisati trokut koji ima najveći opseg, zajedno s njegovim indeksom. Opseg najvećeg trokuta ispisati u formatu ("Trokut %d ima opseg %.2f s koordinatama %f %f %f\n"). Potrebno je izračunati udaljenosti između dvije susjedne točke kako bi se dobila jedna stranica trokuta, napraviti izračun za preostale dvije stranice i to ponoviti za svaki trokut iz polja trokuta. Koristiti isključivo pokazivačku notaciju. Koristiti *typedef*.

$$d(t1, t2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$O_T = a + b + c$$

2 Napisati C program koji će u polje struktura artikl omogućiti unos *n* artikala (struktura ima članove ime, cijena i kolicina), gdje je *n* $(1 \leq n \leq 50)$. Dinamički zauzeti memoriju za polje artikala (u potpunosti rukovati memorijom). U pokazivač na strukturu artikl *max* spremiti memorijsku adresu najskupljeg artikla. Ispisati podatke o najskupljem artiklu. Koristiti isključivo pokazivačku notaciju. Koristiti *typedef*.

3 Napisati C program koji će ponuditi učitavanje 10 točaka u trodimenzionalnom prostoru. Kreirati strukturu tocka s tri realna člana *x*, *y*, *z*. Dinamički zauzeti memoriju za polje točaka (u potpunosti rukovati memorijom). Program treba pronaći i ispisati indeks i koordinate najviše točke. Najviša točka je onaj koja ima najveću z koordinatu. Koristiti isključivo pokazivačku notaciju. Koristiti *typedef*.

4 Napisati C program koji omogućuje unos dva broja *m* i *n*, gdje je *m* $(3 \leq m \leq 30)$, a *n* $(1 \leq n < 11)$. Omogućiti unos *m* točaka i *n* trokuta. Dinamički zauzeti memoriju za polje točaka i polje trokuta (u potpunosti rukovati memorijom). Pronaći i ispisati trokut koji ima najveću i najmanju udaljenost od ishodišta, zajedno s njihovim indeksima. Udaljenost trokuta ispisati u formatu ("Trokut %d ima udaljenost %.2f s koordinatama %f %f %f\n"). Potrebno je izračunati udaljenosti između pojedine koordinate trokuta i ishodišta kako bi se dobila udaljenost trokuta, napraviti izračun za svaki trokut iz polja trokuta. Koristiti isključivo pokazivačku notaciju. Koristiti *typedef*.

$d(t1, t2) = \sqrt{(x_2)^2 + (y_2)^2}$, gdje je t1 ishodište s koordinatama (0, 0)

$\bar{d} = \frac{(d1+d2)}{2}$, predstavlja prosječnu udaljenost točaka trokuta