ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И ИНФОРМАЦИСКИ ТЕХНОЛОГИИ

ПОДАТОЧНИ СТРУКТУРИ И ПРОГРАМИРАЊЕ 2023/2024

АУДИТОРИСКИ ВЕЖБИ 1

ВОВЕД ВО СТРУКТУРИ. БИТСКИ ОПЕРАЦИИ. ЕНУМЕРАЦИЈА

1. Вовед во структури (во C)

Дефинирање на структура

Структурите претставуваат множество од логички поврзани податоци, од ист, но најчесто од различен податочен тип, во единствена целина (под исто име). Тие претставуваат нов податочен тип и се користат заради поедноставна претстава на податоците и манипулација со нив.

Која е разликата со низи?

Структурите заедно со покажувачите се основа за формирање на нови податочни структури, како магацини, редови, поврзани листи, дрва, итн.

```
struct imeNaStruktura  {
        tip1  chlen1;
        tip2  chlen2;
        ...
  } Promenliva1, Promenliva2;  /* декларации на променливи од тип imeNaStruktura */
```

Членовите во рамки на структурата (`chlen1`, `chlen2`) може да бидат променливи од примитивен податочен тип (`int`, `float`, `char`, итн.), агрегати како низи или променливи од други структури.

Структурата **не смее** да содржи променлива која е инстанца од самата структура, но **може** да содржи променлива која е покажувач кон иста структура.

Пример: Се чуваат податоци за студенти во следниов формат:

Име: *Mike*
Презиме: *Ross*
Возраст: *23*
Индекс: *1002009*
Пол: *M*

Податоците ги групираме во единствена структура на следниов начин:

```
struct Student {
       char ime[15];
       char prezime[20];
       int vozrast;
       int indeks;
       char pol;
};
```

Декларирање на променливи од тип структура

Дефиниција:

```
struct Nasoka
{
       char ime[60];
       char kratko_ime[6];
       int shifra;
};
```

или

Дефиниција и декларација:

```
struct Nasoka
{
       char ime[60];
       char kratko_ime[6]; `
       int shifra;
}n1, n2; /* декларација на две
променливи */
```

Декларација:

```
struct Nasoka n1, n2;  /*но не смее Nasoka n1,
n2, мора клучниот збор struct да стои прв (само
во C!!!)*/
```

Операции со структури

Оператор	Објаснување
=	Доделување на променлива од типот структура на друга променлива од истиот тип
&	Земање на адреса на променлива од типот структура
. или ->	Пристап до членовите на структурата
sizeof	Одредување на големината на структурата во бајти

 Структурите не може да се споредуваат со помош на релационите оператори == и != (со нив може да се споредуваат одделни примитивни членови на структурата (`int`, `float`, итн.))

Иницијализација на променливи од типот структура

```
struct Nasoka prv = {"Kompjutersko sistemsko inzhenerstvo, avtomatika i robotika",
"KSIAR", 8}, vtor = {"Kompjuterski tehnologii i inzhenerstvo", "KTI", 3};
```

 Иницијализацијата на овој начин се прави веднаш после декларација на променливата!

Променливите може да се иницијализираат и преку доделување на вредности на поединечни членови од структурата (види пристап до елементи) или преку изедначување со веќе иницијализирана променлива од истата структура.

```
struct Nasoka prv = {"Kompjutersko sistemsko inzhenerstvo, avtomatika i robotika",
"KSIAR", 8}, tret;
tret = prv;
```

но

```
if(tret==prv)   //  грешка
```

Пристап до елементите од структурата (оператор „точка“)

```
prv.ime = "Kompjutersko sistemsko inzhenerstvo, avtomatika i robotika";
printf("%s", prv.ime);
```

Низи (полиња) од структури

```
struct Nasoka nasoki[8];        //низа од 8 елементи од тип Nasoka
nasoki[0].shifra;               //пристап до шифрата на првата насока
nasoki[0].ime[0];               //пристап до првата буква од името на првата насока
```

Вгнездени структури (структура во структура)

```
struct Fakultet
{
      char ime[30];
      char adresa[50];
      struct Nasoka nasoki[8];      //8 насоки како членови на факултетот
} feit;                            //декларација на променлива feit од тип Fakultet

feit.nasoki[0].ime = "Kompjutersko sistemsko inzhenerstvo, avtomatika i robotika";

feit.nasoki[0].ime[0] = 
```

Покажувач кон структура

```
struct Nasoka *p;
struct Nasoka nasoka;
p = &nasoka;       //p ја добива адресата на nasoka, односно покажуваат на иста мемориска адреса
struct Fakultet fakul1, fakul2;
fakul1 = fakul2;   //две променливи со иста содржина на различни мемориски локации
struct Fakultet *f;
f = &fakul1;       //f ја добива адресата на fakul1, односно покажуваат на иста мемориска адреса
```

Пристап до елементи на структура преку покажувач (оператор ->)

```
p->shifra = 8; е исто со (*p).shifra = 8;
nasoka.shifra = 7; е исто со (&nasoka)->shifra = 7;
```

Користење на typedef

Креира синоним за веќе дефиниран податочен тип, не креира нов податочен тип. Се употребува за креирање на скратени имиња на податоци.

```
typedef int cel;    //cel друго име за податочниот тип int
cel m;
typedef struct fakultet tip_fakultet;  //tip_fakultet е друго име за податочниот тип struct fakultet
```

Така, после горната дефиниција, наместо
```
struct fakultet feit;
```
може да се напише
```
tip_fakultet feit;
```

Пренесување на структура во функција во C

Структурите се пренесуваат на три начини:
- се пренесува цела структура по вредност
- се пренесуваат индивидуални членови на структурата
- се пренесува покажувач кон структура

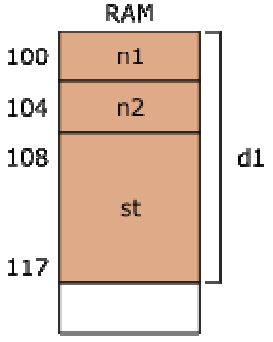
⚠ Примери се дадени во наредните задачи

Структури во меморија

Структурата во меморијата зафаќа онолку простор колку што зафаќаат сите членови во неа.
Во примерот на сликата `int` и `float` податочните членови зафаќаат по **4 бајти**, додека низата од **знаци** `st` зафаќа мемориски простор од 10 бајти (10 членови по 1 бајт).

Тип на податок	Големина во бајти
char	1
int	4
float	4
double	8

```
struct contact
{
    int n1;
    float n2;
    char st[10];
} d1;
```



Задачи:

1. Напишете функција за пресметување на растојание меѓу две точки зададени со нивните координати. Потоа напишете програма што ќе пресмета периметар на даден триаголник.

Решение:

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

struct Koordinati {
	float x;
	float y;
};

float Rastojanie(struct Koordinati a, struct Koordinati b)
{
	return sqrt(pow(a.x - b.x, 2) + pow(a.y - b.y, 2));
}

int main()
{
	struct Koordinati m[3]; //niza od tri parovi na koordinati
	int i;
	float perimetar, a, b, c;

	for (i = 0; i < 3; i++)
	{
		printf("Vnesi gi koordinatite na %d-ta tocka:\n", i + 1);
		fflush(stdout);
		printf("x = "); fflush(stdout);
		scanf("%f", &m[i].x);
		printf("y = ");  fflush(stdout);
		scanf("%f", &m[i].y);
	}

	a = Rastojanie(m[0], m[1]);
	b = Rastojanie(m[0], m[2]);
	c = Rastojanie(m[1], m[2]);

	perimetar = a + b + c;
	printf("Perimetarot na triagolnikot e %f\n", perimetar);

	return 0;
}
```

2. Да се напише програма во која ќе се дефинира структура за претставување на комплексни броеви. Потоа да се напишат функции за собирање, одземање и множење на два комплексни броја. Програмата треба да се тестира со внесување на два комплексни броја.

Решение:

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

struct Kompleksen
{
	int real;
	int imag;
};

typedef struct Kompleksen Kompleksen;

/* дефинирање на прототип на функции */
Kompleksen soberi(Kompleksen a, Kompleksen b);
Kompleksen odzemi(Kompleksen *p1, Kompleksen *p2);
void mnozhi(Kompleksen a, Kompleksen b, Kompleksen *c);
void pechati(const Kompleksen *p);

int main()
{
	Kompleksen k[2], rez;
	int i;
	printf("Vnesete dva kompleksni broja:\n"); fflush(stdout);
	for (i = 0; i < 2; i++)
	{
		printf("Realen:"); fflush(stdout);
		scanf("%d", &k[i].real);
		printf("Imaginaren:"); fflush(stdout);
		scanf("%d", &k[i].imag);
	}
	printf("Gi vnesovte broevite:\n"); fflush(stdout);
	pechati(&k[0]);
	pechati(&k[1]);
	printf("Nivniot zbir e:\n"); fflush(stdout);
	rez = soberi(k[0], k[1]);
	pechati(&rez);
	printf("Nivnata razlika e:\n"); fflush(stdout);
	rez = odzemi(&k[0], &k[1]);
	pechati(&rez);
	printf("Nivniot proizvod e:\n"); fflush(stdout);
	mnozhi(k[0], k[1], &rez);
	pechati(&rez);
	return 0;
}

void pechati(const Kompleksen *p) /* променливата p е константна, што значи дека
нејзината вредност не смее да биде променета во оваа функција */
{
	printf("%4d", p->real);
	p->imag >= 0 ? printf("+j%-4d\n", p->imag) : printf("-j%-4d\n", abs(p->imag)); /*
тернарен оператор */
}
```

```
Kompleksen soberi(Kompleksen a, Kompleksen b) /* пренос по вредност, се креираат локални
копии во функцијата */
{
	Kompleksen c = a;
	c.real += b.real;
	c.imag += b.imag;
	return c;
}

Kompleksen odzemi(Kompleksen *p1, Kompleksen *p2)  /* преносот по покажувач заштедува
меморија */
{
	Kompleksen c = *p1;
	c.real -= (*p2).real;
	c.imag -= (*p2).imag;
	return c;
}

void mnozhi(Kompleksen a, Kompleksen b, Kompleksen *c)  /* третиот аргумент се пренесува
по покажувач, за вредноста која ќе ја добие во оваа функција, да може да се искористи
онаму каде е повикана функцијата */
{
	c->real = a.real * b.real - a.imag * b.imag;
	c->imag = a.real * b.imag + a.imag * b.real;
}
```

3. Од тастатура се читаат непознат број податоци за студенти. Податоците се внесуваат така што во секој ред се дава:
- име,
- презиме,
- број на индекс (формат xxx/yy), и
- четири броја (поени од секоја задача)

со произволен број празни места или табулатори меѓу нив. Да се напише програма која ќе испечати список на студенти, каде во секој ред ќе има: презиме, име, број на индекс, вкупен број на поени од четирите задачи, сортиран според вкупниот број на поени. Имињата и презимињата да се напишат со голема почетна буква.

Решение:

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

struct Student
{
	char ime[15];
	char prezime[20];
	int  index[2];   /* бројот и годината од индексот во низа од два елементи */
	int  poeni;      /* овде ќе биде сместен вкупниот број на поени од четирите задачи
*/
};

void pochetnaGolema(char *s);
void sortiraj(struct Student a[], int n);

int main()
{
	struct Student st[50];
	char imeprez[36];
	int i, rbr = 0, zadaca[4];
	for (;;)
	{
		i = scanf("%s %s %d/%d %d %d %d %d\n", &st[rbr].ime, &st[rbr].prezime,
&st[rbr].index[0], &st[rbr].index[1], &zadaca[0], &zadaca[1], &zadaca[2], &zadaca[3]);
		if (i != 8) break;
		else
		{
			pochetnaGolema(st[rbr].ime);
			pochetnaGolema(st[rbr].prezime);
			st[rbr].poeni = 0;
			for (i = 0; i < 4; i++) st[rbr].poeni += zadaca[i];
			rbr++;
		}
	}
	sortiraj(st, rbr);


	for (i = 0; i < rbr; i++)
	{
		strcpy(imeprez, st[i].prezime); /* формираме единствен стринг кој ги чува
презимето и името */
		strcat(imeprez, " "); /* функцијата strcat(char *vo, char *od) ја прилепува
содржината на стрингот od кон стрингот vo */
		strcat(imeprez, st[i].ime);
		printf("%3d. %-35s %3d/%2d %4d\n", i + 1, imeprez, st[i].index[0],
st[i].index[1], st[i].poeni);
	}
	return 0;
```

```
}

void pochetnaGolema(char *s)
{
	*s = toupper(*s);
	while (*++s != '\0')
		*s = tolower(*s);
}

void sortiraj(struct Student a[], int n)
{
	int i, j;
	struct Student s;
	for (i = 0; i < n; i++)
		for (j = 0; j < n - i - 1; j++)
			if (a[j].poeni < a[j + 1].poeni)
			{
				s = a[j];
				a[j] = a[j + 1];
				a[j + 1] = s;
			}
}
```

4. Креирајте текстуална датотека Podatoci.txt во следниов формат:
   [prezime]…[ime]…[bod1]…[bod2]
   каде:
   - колоната [prezime] зафаќа 15 места;
   - колоната [ime] зафаќа 10 места;
   - колоната [bod1] и [bod2] зафаќаат по 4 места и претставуваат број на поени освоени на прв и втор колоквиум, соодветно. (при внесување на податоци во датотеката Podatoci.txt внимавајте на местата кои ги зафаќаат колоните).

   Напишете програма која ја чита содржината на датотеката Podatoci.txt и според податоците во неа ќе креира нова датотека, Polozhile.txt во која треба да се наоѓа список само на оние студенти кои го положиле испитот, сортиран според бројот на вкупни поени (се смета дека испитот е положен ако од двата колоквиуми се освоени по минимум 30 поени, а вкупниот број на поени е минимум 100).
   Датотеката Polozhile.txt содржи податоци во следниов формат:
   [prezime]…[ime]…[vkupno]

   Решение:

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

typedef struct Student {
	char prezime[15];
	char ime[10];
	int prv;
	int vtor;
} Student;

int main()
{
	FILE *Vlez, *Izlez;

	Student podatoci[15];  /* претпоставуваме дека во влезната датотека нема повеќе од
15 податоци за студенти */

	Student pom;
	int i = 0, j, k;

	if ((Vlez = fopen("Podatoci.txt", "r")) == NULL)
	{
		printf("Greska pri otvaranje na datotekata");
		exit(1);
	}
	Izlez = fopen("Polozhile.txt", "w");

	/*
	Ги читаме податоците од влезната датотека и ги сместуваме во низата podatoci.
	Променливата i го означува бројот на студенти кои го положиле испитот.
	*/

	while (fscanf(Vlez, "%15s%10s%4d%4d", &podatoci[i].prezime, &podatoci[i].ime,
&podatoci[i].prv, &podatoci[i].vtor) != EOF)
	{
		if ((podatoci[i].prv >= 30) && (podatoci[i].vtor >= 30) && (podatoci[i].prv
+ podatoci[i].vtor >= 100))
		{
			podatoci[i].prv += podatoci[i].vtor; /* во podatoci[i].prv сега ќе
биде сместен вкупниот број на поени */
			i++;
		}
	}
```

```
	/* Со SimpleSort ги сортираме елементите од низата studenti, според вкупниот број
на поени */
	for (j = 0; j < i - 1; j++)
		for (k = j + 1; k < i; k++)
			if (podatoci[k].prv > podatoci[j].prv)
			{
				pom = podatoci[j];
				podatoci[j] = podatoci[k];
				podatoci[k] = pom;
			}

	/* Сортираните податоци ги запишуваме во излезна датотека */
	for (j = 0; j < i; j++)
		fprintf(Izlez, "%-15s%-10s%-4d\n", podatoci[j].prezime, podatoci[j].ime,
podatoci[j].prv);

	fclose(Vlez);
	fclose(Izlez);

	return 0;
}
```

2. Операции со битови

Податоците во компјутерите се претставени како секвенци од битови (нули и единици). Секвенца од 8 бита претставува еден бајт.
Следните оператори извршуваат логички операции врз секој бит на операндот (операндите).

Оператор	Опис
& (AND)	Битовите во резултатот се поставуваат на 1 само ако соодветните битови во операндите имаат вредност 1
| (OR)	Битовите во резултатот се поставуваат на 1 ако барем еден од соодветните битови во операндите има вредност 1
^ (XOR)	Битовите во резултатот се поставуваат на 1 само ако еден од соодветните битови во операндите има вредност 1
<< (поместување во лево)	Битовите од првиот операнд се поместуваат во лево за онолку места колку што означува вториот операнд (од десно се пополнува со нули)
>> (поместување во десно)	Битовите од првиот операнд се поместуваат во десно за онолку места колку што означува вториот операнд (поплнувањето од лево е машински зависно)
~ (комплемент)	Сите нули се претвораат во единици и обратно

Операциите со битови се машински зависни (не даваат исти резултати на секоја машина).

Големината на примитивните податочни типови (колку бајти зафаќаат) е машински зависна.

Задачи:

5. Напишете програма која ќе изврши множење и делење на даден цел број со број кој е степен со основа 2 (множењето и делењето да се изврши со користење на операции со битови).

Решение:

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

int main()
{
	int broj, mnozhitel, pomnozhen, podelen;
	printf("Vnesete cel broj: "); fflush(stdout);
	scanf("%d", &broj);

	printf("So koj broj sakate da mnozhite/delite? (Vneseniot broj treba da e stepen
so osnova 2) ");
	fflush(stdout);
	scanf("%d", &mnozhitel);

	pomnozhen = broj << (int)log2(mnozhitel);
	podelen = broj >> (int)log2(mnozhitel);

	printf("%d * %d = %d\n", broj, mnozhitel, pomnozhen); fflush(stdout);
	printf("%d / %d = %d\n", broj, mnozhitel, podelen); fflush(stdout);
	return 0;
}
```

Излез

```
Vnesete cel broj: 100
So koj broj sakate da mnozhite/delite? (Vneseniot broj treba da e stepen so osnova 2) 4
100 * 4 = 400
100 / 4 = 25
```

6. Напишете програма која на екран ќе ја прикаже бинарната претстава на даден декаден број.

Решение:

```
#include <stdio.h>
void prikazhiBitovi(int vrednost);

int main() {
	int x, kraj;
	printf("Int promenliva ima %d bajti\n", sizeof(int)); fflush(stdout);

	while (1) {
		printf("Vnesete cel broj: "); fflush(stdout);
		kraj = scanf("%d", &x);
		if (kraj == 0) break;
		prikazhiBitovi(x);
	}
	return 0;
}

void prikazhiBitovi(int vrednost) {
	int i;
	int vkBitovi = sizeof(vrednost) * 8;
	int maska = 1 << vkBitovi - 1;	/* првиот бит (од 32 бита) го поставуваме на
вредност 1 */

	printf("Dekadna vrednost: %d => Binarna vrednost: ", vrednost); fflush(stdin);

	for (i = 1; i <= vkBitovi; i++) {  //ги изминуваме сите битови
		vrednost & maska ? printf("1") : printf("0");
		vrednost <<= 1;
		if (i % 8 == 0) putchar(' ');
	}
	putchar('\n');
}
```

Излез

```
Int promenliva ima 4 bajti
Vnesete cel broj: 255
Dekadna vrednost: 255 => Binarna vrednost: 00000000 00000000 00000000 11111111
Vnesete cel broj: 1205
Dekadna vrednost: 1205 => Binarna vrednost: 00000000 00000000 00000100 10110101
Vnesete cel broj: 3456783
Dekadna vrednost: 3456783 => Binarna vrednost: 00000000 00110100 10111111 00001111
Vnesete cel broj: s
```

7. Напишете програма која ќе го постави/избрише n-тиот бит на даден бинарен број.

Решение:

```
#include <stdio.h>
char binaren[sizeof(int) * 8 + 1];    /* +1 заради null терминаторот */

char *voBinaren(int vrednost)
{
	int i;
	int vkBitovi = sizeof(int) * 8;
	int maska = 1 << vkBitovi - 1;

	for (i = 0; i < vkBitovi; i++)
	{
		if (vrednost & maska)
			binaren[i] = '1';
		else
			binaren[i] = '0';
		vrednost <<= 1;
	}
	binaren[32] = '\0';

	return binaren;
}

int main() {
	int broj, bit;
	printf("Vnesete cel broj: "); fflush(stdout);
	scanf("%d", &broj);

	printf("Vnesete go redniot broj na bitot vo binarnata pretstava na brojot koj
sakate da go postavite/izbrishete: "); fflush(stdout);
	scanf("%d", &bit);
	/* битот на најмалку значајна позиција има реден број 1 */

	int maskPostavi = 1 << (bit - 1);
	int postavi = broj | maskPostavi;
	/*  битот на позиција bit станува 1  */

	int maskIzbrishi = ~(1 << (bit - 1));
	int izbrishi = broj & maskIzbrishi;
	/*  битот на позиција bit станува 0  */

	printf("broj = %d voBinaren(broj) = %s\n", broj, voBinaren(broj)); fflush(stdout);
	printf("postavi = %d voBinaren(postavi) = %s\n", postavi, voBinaren(postavi));
fflush(stdout);
	printf("izbrishi = %d voBinaren(izbrishi) = %s\n", izbrishi, voBinaren(izbrishi));
fflush(stdout);
	return 0;
}
```

Излез

```
Vnesete cel broj: 34567
Vnesete go redniot broj na bitot vo binarnata pretstava na brojot koj sakate da go  postavite/izbrishete: 5
broj = 34567 voBinaren(broj) = 00000000000000001000011100000111
postavi = 34583 voBinaren(postavi) = 00000000000000001000011100010111
izbrishi = 34567 voBinaren(izbrishi) = 00000000000000001000011100000111
```

3. Енумерација

Енумереација претставува множество од целобројни константи претставени со идентификатори заради полесно разбирање на кодот. Вредностите на енумерациските променливи започнуваат од нула и се инкрементираат за еден, но можат и експлицитно да бидат доделени.
Енумерацијата е нов податочен тип кој содржи ограничено множество од вредности.

Пример 8: Да се напише програма која ќе користи енумерација за месеците и истите ќе ги испечати на екран.

```
#include <stdio.h>

enum meseci { JAN = 1, FEV, MAR, APR, MAJ, JUN, JUL, AVG, SEP, OCT, NOE, DEK };

int main()
{
	enum meseci mesec;   /* променлива од тип enum meseci */
	const char *mesecIme[] = { " ", "Januari", "Fevruaru", "Mart", "April", "Maj",
"Juni", "Juli", "Avgust", "Septemvri", "Oktomvri", "Noemvri", "Dekemvri" };

	for (mesec = JAN; mesec <= DEK; mesec++)
		printf("%2d.%-9s\n", mesec, mesecIme[mesec]);
	return 0;
}
```

Пример 9: Да се напише програма за користење на енумерација која ќе реализира квиз во кој корисникот внесува боја, и доколку бојата ја има во енумерацијата, на екран ќе се испечати порака која е соодветна за бојата.

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

enum spektar { red, orange, yellow, green, blue, violet };

const char * boi[] = { "red", "orange", "yellow", "green", "blue", "violet" };

int main()
{
	char izbor[10];
	enum spektar boja;
	bool najdena = 0; /* прима вредност 1 и 0. Мора да биде вклучена библиотеката
stdbool.h */

	puts("Vnesete boja (\\n za kraj):");  fflush(stdout); /* наместо printf("Vnesete
boja (\\n za kraj):"); */
	while (gets(izbor) != NULL && izbor[0] != '\0')
	{
		for (boja = red; boja <= violet; boja++)
		{
			if (strcmp(izbor, boi[boja]) == 0)
			{
				najdena = 1;
				break;
			}
		}

		if (najdena)
			switch (boja)
			{
			case red: puts("Roses are red."); fflush(stdout);
```

```
                        break;
                case orange: puts("Poppies are orange."); fflush(stdout);
                        break;
                case yellow: puts("Sunflowers are yellow."); fflush(stdout);
                        break;
                case green: puts("Grass is green."); fflush(stdout);
                        break;
                case blue: puts("Bluebells are blue."); fflush(stdout);
                        break;
                case violet: puts("Violets are violet."); fflush(stdout);
                        break;
                }
        else
                printf("Nemam idea za ovaa boja %s.\n", izbor); fflush(stdout);
        najdena = 0;
        puts("Sledna boja, (\\n za kraj):"); fflush(stdout);
    }
    puts("Kraj!"); fflush(stdout);
    return 0;
}
```

Дополнителна литература (поставена на портал):

Книга: **“C How to Program, Seventh Edition”, Deitel,** Chapter 10: C Structures, Unions, Bit Manipulation and Enumerations