Дефинисање структуре

Структуре (записи - record) су сложени типови података и састоје се од одређеног броја елемената. Елементи структуре се називају *поља*. Поља могу бити различитих типова и обележавају се идентификаторима. Могу бити прости или сложени типови података, тј елементи структуре могу бити друге структуре или низ, цели, реални или знаковни податак.

Разлика између записа и низа је у томе што низ чине елементи истог типа а структуру могу да чине елементи различитих типова. Записи су начин да се организују сложени подаци, посебно код дугачких програма, због тога што у многим ситуацијама омогућавају групи промењивих да буду третиране као једна целина, уместо свака за себе.

Дефиниција структуре:

```
struct ime_strukture
{
        tip ime_promenjive_1:
        tip ime_promenjive_2:
        …
};
```

Идентификатори који се користе као имена промењивих (поља), могу се независно користити само унутар структуре у којој су дефинисани.

Елементима поља структуре се приступа следећом конструкцијом: ime_strukture.clan.

Пример 01: Унети и приказати датум помоћу записа (структуре) посебним уносом сваког елемента поља.

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

int main()
{
        struct datum
        {
                int dan;
                char mesec[10];
                int godina;
        };
        datum d;
        cout << "Unesi dan: ";
        cin >> d.dan;
        cout << "Unesi mesec: ";
        cin >> d.mesec;
        cout << "Unesi godinu: ";
        cin >> d.godina;
        cout << d.dan << ". " << d.mesec << " " << d.godina << " g." << endl;
        return 0;
}
```

```
Unesi dan: 31
Unesi mesec: januar
Unesi godinu: 3016
31. januar 3016 g.
```

Пример02 : Унети и приказати датум помоћу записа (структуре) посебним уносом свих елемената поља заједно.

```
#include<iostream>
using namespace std;
int main()
{
        struct datum
        {
                int dan;
                char mesec[10];
                int godina;
        };
        datum d = {2, "januar", 1986};
        cout << d.dan << ". " << d.mesec << " " << d.godina << " g." << endl;
        return 0;
}
```

Пример03 : Написати програм који ће дефинисати структуру са називом ученик, а која садржи елементе: име, презиме, разред. Направити и две промењиве типа ученик: Марко и Петар.

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

int main()
{
	struct ucenik
	{
		char ime[20];
		char prezime[25];
		int razred;
	};
	ucenik Petar = { "Petar", "Petrovic", 3 };
	ucenik Marko;
	cout << "Unesi ime: ";
	cin >> Marko.ime;
	cout << "Unesi prezime: ";
	cin >> Marko.prezime;
	cout << "Unesi razred: ";
	cin >> Marko.razred;
	cout << Petar.ime << " " << Petar.prezime << " " << Petar.razred
		<< ". razred" << endl;
	cout << Marko.ime << " " << Marko.prezime << " " << Marko.razred
		<< ". razred" << endl;
	return 0;
}
```

```
Unesi ime: Marko
Unesi prezime: Markovic
Unesi razred: 4
Petar Petrovic 3. razred
Marko Markovic 4. razred
```

<u>Структуре и функције</u>

Структуре могу бити параметри функција. Структура се сматра једним податком па се као таква преноси у функцију помоћу вредности. Помоћу функције sizeof() се може проверити стварно заузеће меморије структуром.

Пример 01: Проверити да ли је унешени датум (дан, месец, година) могућ или не.

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

int main()
{
	struct datum
	{
		int dan;
		int mesec;
		int godina;
	};
	int mesec[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
	datum d;
	cout << "Unesi dan: " << endl;
	cin >> d.dan;
	cout << "Unesi mesec: " << endl;
	cin >> d.mesec;
	cout << "Unesi godinu: " << endl;
	cin >> d.godina;
	if ((d.godina % 4 == 0) && (d.godina % 100 != 0)) mesec[1] = 29;
	if ((d.dan < 1) || (d.dan > mesec[d.mesec - 1]) || (d.mesec < 1)
		|| (d.mesec > 12) || (d.godina < 0)) cout << "Datum nije moguc!" << endl;
	else cout << d.dan << ". " << d.mesec << ". " << d.godina << ". godine" << endl;
	return 0;
}
```

```
Unesi dan:
10
Unesi mesec:
5
Unesi godinu:
1900
10. 5. 1900. godine
Press any key to continue . . .
```

```
Unesi dan:
40
Unesi mesec:
10
Unesi godinu:
2016
Datum nije moguc!
Press any key to continue . . .
```

Пример02 : Узети податке о особи и проверити колико меморије структура податак заузима.

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

int main()
{
	struct osoba
	{
		string ime;
		string prezime;
		int godine;
	};
	osoba covek = { "Marko", "Markovic", 16 };
	cout << "Cela struktura zauzima " << sizeof(covek) << " bajtova." << endl;
	return 0;
}
```

```
Cela struktura zauzima 60 bajtova.
Press any key to continue . . .
```

Пример 03: Узети податке о особи и приказати их у функцији.

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

struct osoba
{
	string ime, prezime;
	int godine;
};

void funkcija(osoba x);

int main()
{
	osoba covek = { "Marko", "Markovic", 16 };
	funkcija(covek);
	return 0;
}

void funkcija(osoba x)
{
	cout << "Podaci o osobi:" << endl;
	cout << "Ime: " << x.ime << endl;
	cout << "Prezime: " << x.prezime << endl;
	cout << "Godine: " << x.godine << endl;
}
```

```
Podaci o osobi:
Ime: Marko
Prezime: Markovic
Godine: 16
Press any key to continue . . .
```

Показивачи и структуре

Показивач на запис се дефинише:

```
struct osoba
{
	string ime;
	int godine;
};
osoba a, *pa;
```

За приступ пољу ime помоћу показивача : (*pa).ime

Заграде су обавезне пошто је приоритет оператора . већи од приоритета оператора *.

Да нема заграда у примеру: *pa.ime, прво би се упутило на поље па тек онда би се користио показивач.

Пошто се показивачи доста користе, уведен је нови оператор -> којим се приступа пољима структуре унутар показивача.

Сада су ови изрази идентични: (*pa).ime и pa->ime.

Пример: Структура тачака се састоји од координата x и y. Иницијализовати тачку (3.2, 4.3) и показивач на дату тачку. Преко показивача приказати координате тачке.

```
#include<iostream>
using namespace std;
int main()
{
	struct tacka
	{
		float x, y;
	};
	tacka a = {3.2, 4.3}, *pa;
	pa = &a;
	cout << (*pa).x << endl;
	cout << pa->y << endl;
	return 0;
}
```

Као резултат добија се:

```
3.2
4.3
```

Коришћење показивача у структурама са функцијама

Пример: Структура ученик се састоји од имена, презимена и разреда. Написати функције за унос и приказ података типа ученик. У главном програму дефинисати промењиву типа ученик и позвати обе функције.

```
#include<iostream>
#include<string>
using namespace std;
struct ucenik
{
	string ime, prezime;
	int razred
};
void unos(ucenik *pu)
{
	cout << ”Uneti ime, prezime i razred: ”;
	cin >> pu->ime;
	cin >> pu->prezime;
	cin >> pu->razred;
}
```

```
void prikaz(ucenik u)
{
	cout << u.ime << endl;
	cout << u.prezime << endl;
	cout << u.razred << endl;
}
int main()
{
	ucenik x;
	unos (&x);
	prikaz (x);
	return 0;
}
```

Као резултат се добија:

```
Uneti ime, prezime i razred: Petar Patrovic 2
Petar
Petrovic
2
```

Показивачи и структуре

Пример: Написати програм којим ће се сабрати два комплексна броја. Комплексне бројеве приказати као структуру, а операцију сабирања решити у функцији.

```
#include<iostream>
using namespace std;
struct KB
{
	float re, im;
};
KB sabiranje (KB z1, KB z2);
int main()
{
	KB z1, z2, z3;
	cout << “Uneti realni i imaginarni deo prvog kompleksnog broja: ”;
	cin >> z1.re >> z1.im;
	cout << “Uneti realni i imaginarni deo drugog kompleksnog broja: ”;
	cin >> z2.re >> z2.im;
	z3 = sabiranje(z1, z2);
	cout << “z3 = ” << z3.re << “ + i” << z3.im << endl;
	return 0;
}
KB sabiranje (KB z1, KB z2)
{
	KB z;
	z.re = z1.re + z2.re;
	z.im = z1.im + z2.im;
	return z;}
```

Као резултат се добија:

```
Uneti realni i imaginarni deo prvog kompleksnog broja: 11 22
Uneti realni i imaginarni deo drugog kompleksnog broja: 12 23
z3 = 23 + i45
```

Пример: Ако су дате координате тачака A(4,4) B(8,2) C(4,2), која су темена правоуглог троугла, написати програм који рачуна површину овог троугла. Користити структуре за дефинисање тачака и троугла.

```
#include<iostream>
using namespace std;
int main()
{
	struct tacka
	{
		int x, y;
	};
	struct trougao
	{
		tacka A, B, C;
	};
	trougao T = {{4,4}, {8,2}, {4,2}};
	float P = ((T.B.x – T.A.x) * (T.A.y – T.C.y)) / 2;
	cout << ” P = ” << P << endl;
	return 0;
}
```

Као резултат се добија: 4

Пример: Написати програм који рачуна обим троугла. Троугао је представљен са три тачке а свака тачка има координате x и y. Обим троугла решити у функцији. У главном програму унети бројеве и приказати резултате.

```
#include<iostream>
#include<cmath>
using namespace std;
struct tacka
{
	int x, y;
};
struct trougao
{
	tacka A, B, C;
};
float rastojanje(tacka A, tacka B);
float obim (trougao T);
int main()
{
	trougao ABC = {{2,2}, {6,2}, {4,5}};
	cout << ”Obim trougla: ” << obim(ABC) << endl;
	return 0;
}
float rastojanje(tacka A, tacka B)
{
	return sqrt(pow(A.x – B.x, 2) + pow(A.y – B.y, 2));
}
float obim (trougao T)
{
	float O, a, b, c;
	a = rastojanje(T.A, T.B);
	b = rastojanje(T.B, T.C);
	c = rastojanje(T.A, T.C);
	O = a + b + c;
	return 0;
}
```

Као резултат се добија:

```
obim trougla: 8.6
```