

Laborator 10 – Limbaje de Programare

1 Recapitulare și exemple – Tipuri definite de utilizator

Să ne amintim câteva exemple de la curs privind tipuri de date definite de utilizator.

1.1 Tipul enumerare

1. Exemplu tipul enumerare:

```
enum zile {D, L, Ma, Mc, J, V, S};
//declara tipul enum zile
enum zile zi;
// declara variabila zi de tip zile

int nr_ore_lucru[7];
// numar de ore pe zi

for (zi = L; zi <= V; zi++)
    nr_ore_lucru[zi] = 8;
```

1.2 Tipul structură

1. Exemplu tip structură:

```
struct student {
// numele complet al tipului (incl. "struct")

    char nume[32], prenume[32];
    // doua tablouri de caractere
    char *domiciliu;
    // ADRESA! memoria pt. sir se alocă altundeva
    float medie_an[4];
    float nota_dipl;
} stud1, stud2;
// doua variabile declarate direct aici

struct student s, p, *ps;
// declaratie separata de variabile

s.nota_dipl = 9.25;

strcpy(s.nume, "Stefanovici");
// NU! s.nume = ... (s.nume e un tablou)

s.domiciliu = "str. Linistei nr. 2";
// sau alocare dinamica (malloc) si strcpy

s.medie_an[2] = 9.35; //ca orice variabila
p = s; //atribuire intre structuri

ps = &s; //ps pointer la structura, i se atribuie adresa lui s

//accesarea campurilor structurii prin intermediul lui ps
```

```
(*ps).nota_dipl = 9.50; //echivalent cu
ps -> nota_dipl = 9.50;
```

2. Declararea unui tip cu *typedef*:

```
typedef struct student {
    /* ceva campuri */
} student_t;

//sau

struct student {
    /* ceva campuri */
}; //defineste tipul

typedef struct student student_t;
//declara un nou nume pt. struct student
```

3. Tablouri de structuri, pentru gruparea logica a datelor:

```
char* nluni[12] = { "ian", /*...,*/ "dec" };
char zile_luna[12]
    = { 31, 28, 31, 30, /*...,*/ 30, 31 };

// e preferabila varianta urmatoare
struct luna {
    char *nume;
    int zile;
};
struct luna luni[12]
    = {{ "ian",31}, /*...,*/ {"dec",31}};
```

4. Structuri recursive – o lista (simplu înlănțuită) de cuvinte:

```
// struct wl e un tip , incomplet definit
struct wl {
    char *word; // cuvantul
    struct wl *next;
    // pointer la structura de acelasi tip
};
//definitia e completa
```

5. Structuri recursive – un arbore binar, având informația din noduri numere întregi:

```
typedef struct t tree;
// defineste tipul incomplet tree = struct t

struct t {
    int val;
    tree *left , *right;
    // foloseste numele din typedef
};
// aici tipul struct t e complet
// si echivalent cu tree
```

6. Structuri cu câmpuri pe biți:

```
struct date_t {
    unsigned sec , min : 6; // nr. de biti
    // se permit tipuri intregi

    unsigned hour , day: 5;
    unsigned month: 4;
```

```

    unsigned year: 6;
} data = {0, 0, 17, 19, 5, 39 };

//...
printf("%u.%u\n", data.day, data.month);

```

1.3 Tipul uniune

1. Exemplu tipul uniune:

```

union u{ // tip uniune
    int i;
    double r;
    char *s;
} val; // trei variante pentru fiecare tip

enum { INT, REAL, SIR } tip;
// tine minte varianta memorata

val.r = 3.14; tip = REAL;
//sau
val.i = 100; tip = INT;
//sau
val.s = "ceva"; tip = SIR;

```

2 Lucrarea 9

Să se implementeze următoarele cerințe:

- Definiți un tip structură pentru reprezentarea numerelor complexe. Folosind tipul definit, scrieți funcții care implementează operațiile cunoscute cu numere complexe (adunarea, înmulțirea, împărțirea, calculul modulului, etc.).
- Definiți un tip pentru reprezentarea studenților dintr-un an, astfel: fiecare student va avea nume, număr matricol, medie multi anuală și poate fi bursier, integralist nebursier sau restanțier; pentru studenții bursieri se mai reține și cuantumul bursei (nr. real), pentru cei integralisti nebursieri – adresa de domiciliu, în timp ce pentru studenții restanțieri se pastrează numărul creditelor restante (Hint: aveți nevoie de o combinație de structură și uniune pentru reprezentarea acestui tip cu variante, plus un tip enumerare pentru a distinge între cele 3 variante). Se vor citi într-un tablou datele pentru un număr oarecare de studenți, și se vor afișa pe ecran întâi studenții bursieri, apoi cei nebursieri, apoi ceilalți, cu toate datele ce îi caracterizează.
- Definiți cu ajutorul pointerilor la structuri un tip recursiv pentru păstrarea unei liste de cuvinte. Creați o listă din cuvintele citite de la intrare sau dintr-un fișier text. Fiecare element din listă va conține și un contor astfel încât dacă un cuvânt se repetă, nu se va adăuga o nouă intrare în listă, ci doar se va incrementa valoarea contorului asociat cuvântului deja existent.