UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

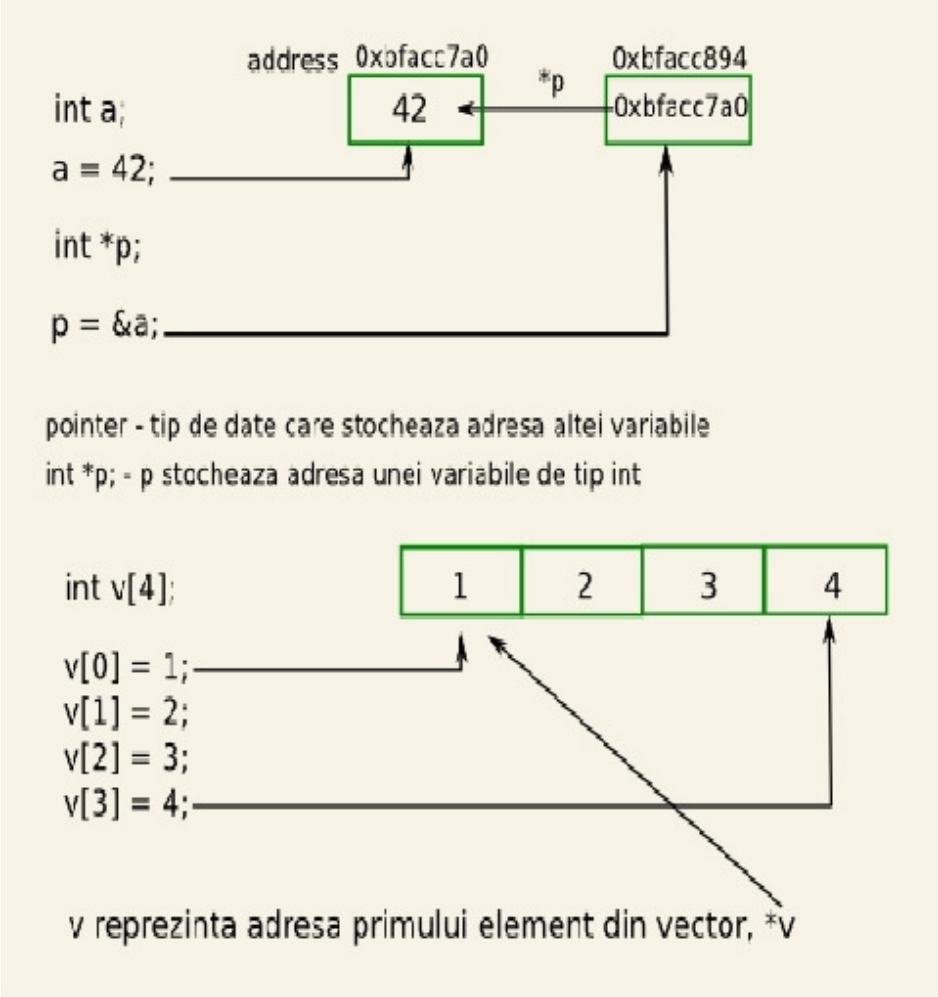
Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Sisteme de operare

2. Elemente de programare în limbajul C

Tipuri de date

- primare (char, int) – pot reține date
- pointer – pot reține adrese din memorie
  - `tip_variabila * nume_variabilă`
  - `int x, *p = &x;`
  - `frintf("%x %d", p, *p);`
  - `int *x=(int*)malloc(10*sizeof(int));`
  - `free(x);`



Tipuri de date (2)

- Array
  - `tip nume_variabila [dimensiune_1]....[dimensiune_n];`
  - `int v1[100][50];`
  - `tip nume_variabila [] = { v1, ..., vn};`
  - `int v2[] = {1, 5, 10};`
  - `printf("%d %d", v1[5][5], v2[1]);`

Tipuri de date (3)

- structuri
  - struct nume_structura {
    - /* câmpuri */
  - };
  - struct point {
    - int x;
    - int y;
    - char desc [1024];
  - };
  - struct point p, *r = &p;
  - printf("%d %d %s", p.x, p.y, p.desc);
  - printf("%d %d %s", r->x, r->y, r->desc);

Instrucțiuni de bază

IF

```
if (condiție) {
  /* instrucțiuni*/
}
else {
  /* instrucțiuni*/
}
```

SWITCH

```
switch(variabilă) {
        Case value_1: {
                /* instrucțiuni*/
                break;
        }
        -------------------------
        case value_n: {
                /* instrucțiuni*/
                break;
        }
        default: {
                /* instrucțiuni*/
                Break;
        }
}
```

Instrucțiuni de bază (2)

Loops

```
for (variabilă; condiție; modificare variabilă) {
        /* instrucțiuni*/
}

do {
        /* instrucțiuni*/
}
while(condiție);

while(condiție) {
        /* instrucțiuni*/
}
```

break / continue

- `break`
  - Încheie cea mai apropiată buclă înconjurătoare
- `Continue`
  - Încheie iterația curentă și trece forțat la următoarea iterație din buclă

Instrucțiuni de bază (2)

Funcții

- prototip
  - return_type function_name (arg_type arg1, ..., arg_type argN);
  - int sum(int p1, intp2);
- corpul funcției
  - return_type function_name (arg_type arg1, ..., arg_type argN) {

```
        /* instrucțiuni*/
    }
```

  - int sum(int p1, intp2) {

```
        return p1 + p2;
    }
```

Apelul unei funcții

- `function_name(args list);`
- `sum(1, 2);`

Directive de preprocesare

- include
  - #include <nume_biblioteca>
  - #include <stdio.h>
- define
  - #define BUFFER_SIZE 1024
  - v = (int*)malloc(BUFFER_SIZE * sizeof(int));
  - #define min(X, Y) ((X) < (Y) ? (X) : (Y))
  - printf("%d", min(10, 5));
- #ifdef MACRO ... #endif
- #ifndef MACRO ... #endif