

Programarea Calculatoarelor

Laborator 8

Pointeri la structuri

Putem defini pointeri la structuri in acelasi mod in care am defini un pointer la orice alta variabila.

Declarare:

```
struct struct_name *struct_pointer;
```

Pentru a accesa membrii unei structuri prin intermediul pointerilor, folosim operatorul ->
`struct_pointer->member_name;`

Instructiunea de mai sus este echivalenta cu:
`(*struct_pointer).member_name;`

Pentru a trimite un pointer la o structura ca argument unei functii, folosim urmatoarea sintaxa:

```
return_type function_name(struct struct_name *param_name);
```

Exemplu:

```
struct Book {
    int book_id;
    char title[50];
    char author[50];
    char subject[100];
};

void printBook(struct Book *book);

int main () {
    struct Book *book_pointer, book1;
    book_pointer = &book1;
    //specificatia pentru book1:
    book1.book_id = 2000;
    strcpy(book1.title, "Defence Against the Dark Arts");
    strcpy(book1.author, "Severus Snape");
    strcpy(book1.subject, "Defence Against the Dark Arts");

    printBook(&book1);
    return 0;
}
```

```

void printBook(struct Book *book) {
    printf("Book title: %s\n", book->title);
    printf("Book author: %s\n", book->author);
    printf("Book subject: %s\n", book->subject);
    printf("Book identifier: %d\n", book->book_id);
}

```

Putem adăuga la structură pointeri la funcții

Exemplu:

```

struct Book {
    int book_id;
    char title[50];
    char author[50];
    char subject[100];
}

void (*printBook)(struct Book *book);

void printBook1(struct Book *book) {
    printf("Book title: %s\n", book->title);
    printf("Book author: %s\n", book->author);
    printf("Book subject: %s\n", book->subject);
    printf("Book identifier: %d\n", book->book_id);
}

void printBook2(struct Book *book) {
    printf("Details: %s, %s, %s, %d\n", book->title, book->author, book->subject, book->book_id);
}

int main () {

    /* declaratii, initializari,... citiri */

    int x;
    scanf("%d",&x);

    switch(x)
    {
        case 1:
            book1.printBook=&printBook1;
            break;
        default:
            book1.printBook=&printBook2;
    }
}

```

```
book1.printBook(&book1);

return 0;
}
```

Accesarea membrilor structurilor prin intermediul pointerilor utilizand alocarea dinamica:

Putem alocam memorie dinamic utilizand functia malloc(din header-ul "stdlib.h")

Sintaxa:

```
ptr = (cast_type*)malloc(byte_size);
```

Exemplu:

```
int main() {
    struct Book *book_ptr;
    int i, n;

    printf("Enter number of books: ");
    scanf("%d", &n);

    //alocam memorie pentru n structuri Book cu book_ptr
    //indicand spre adresa de baza
    book_ptr = (struct Book*)malloc(n * sizeof(struct Book));

    for(i = 0; i < n; ++i) {
        printf("Enter book id, title, author and subject");
        scanf("%d", &(book_ptr+i)->book_id);
        scanf("%s", &(book_ptr+i)->title);
        scanf("%s", &(book_ptr+i)->author);
        scanf("%s", &(book_ptr+i)->subject);
    }
    //afisam informatiile despre cartile citite anterior
    for(i = 0; i < n; ++i) {
        printBook1(book_ptr+i);
    }

    return 0;
}
```

Probleme

1. Sa se construiasca o structura pentru a reprezenta numerele complexe.

- a) Adăugați în structură pointeri la funcții de afișare și citire ;
- b) Scrieti functii pentru suma si inmultirea a doi complecsi care primesc pointeri la cei doi operanzi de tip complex si intorc rezultatul folosind structura de tip complex
- c) Scrieți o funcție ce va avea ca parametri
 - pointeri la două numere complexe X si Y
 - și un parametru de tip pointer la funcție pe care il vom folosi sa specificam una dintre operațiile de suma sau produs.

Folosind aceasta funcție afisati $X^2 * Y$, respectiv $2X + Y$ pentru doi complecsi X si Y cititi de la tastatura