Recapitulare

Responsabili:

- Florin Pop [mailto:florin.pop@cs.pub.ro], George Popescu [mailto:george.popescu@cs.pub.ro]

Obiective

În urma parcurgerii acestui laborator, studentul va fi capabil:

- să-şi estimeze gradul de acoperire al cunoştinţelor la Programare;
- să-şi facă o privire de ansamblu mai bună asupra noţiunilor învăţate la laborator şi a modului în care acestea sunt legate între ele;

Exerciții de laborator CB/CD

Vă invităm să evaluați activitatea echipei de **programare CB/CD** și să precizați punctele tari și punctele slabe și sugestiile voastre de îmbunătățire a materiei. Feedback-ul vostru este foarte important pentru noi să creștem calitatea materiei în anii următori și să îmbunătățim materiile pe care le veți face în continuare.

Găsiți formularul de feedback în partea dreaptă a paginii principale de programare de pe cs.curs.pub.ro într-un frame numit "FEEDBACK" (moodle [http://cs.curs.pub.ro/2016/course/view.php?id=17]). Trebuie să fiți inrolați la cursul de programare, altfel veți primi o eroare de acces.

Pentru a putea înțelege și valorifica feedback-ul mai ușor, apreciem orice formă de feedback negativ constructivă. Nu este suficient să ne spuneți, spre exemplu, *tema 5 a fost grea*, ne dorim să știm care au fost dificultățiile și, eventual, o propunere despre cum considerați că ar fi trebuit procedat.

Exerciţii de Laborator

* [Easy] Scrieţi un program care citeşte un număr n de la tastatură şi apoi alte n*n numere reale, pe care le plasează într-o matrice pătratică, alocată static, de dimensiune n. Să se sorteze apoi aceste numere crescător, de la stânga la dreapta şi de sus în jos, fără a folosi alţi vectori sau matrice, şi apoi să se afişeze matricea sortată.

* [Easy] Scrieţi un program care primește ca parametri un cuvânt şi apoi numele unui fișier text, apoi crează un nou fișier text pe baza primului fișier, în care toate literele din cuvântul citit vor fi înlocuite cu caracterul '*'.

* [Medium] Scrieţi un program care defineşte un vector de 10 numere întregi, pe care îl iniţializează apoi cu numere de la 0 la 9 şi apoi afişează conţinutul memoriei ocupate de vector, octet cu octet, fiecare octet fiind afişat cu două cifre hexazecimale.

* [Medium] a) Care este diferenţa dintre următoarele două expresii? Explicaţi concluzia.

```
int expr = (a + b)*(c - d);
```

```
int expr = ((a + b))*((c - d));
```

b) Care este diferenţa dintre următoarele două expresii? Explicaţi concluzia.

```
printf("Hello there, %s!\n", "student");
```

```
printf(("Hello there, %s!\n", "student"));
```

* [Hard] Fie următoarea structură:

```
struct test_struct {
  char a;
  short b;
  char c;
```

```
  int d;
};
```

Scrieţi un program care defineşte această structură şi o variabilă de acest tip şi care afişează diferenţa dintre dimensiunea totală a structurii (calculată cu operatorul `sizeof`) şi suma dimensiunilor fiecărui câmp în parte al structurii. Explicaţi rezultatul obţinut.

'SOLUŢIE:'

```
#include <stdio.h>

struct test_struct {
  char a;
  short b;
  char c;
  int d;
};// __attribute__((packed));
//decomentati linia anterioara pentru a instrui compilatorul sa nu mai puna
// "padding" in structuri.

int main() {
  struct test_struct t;
  void *start, *offset;

  printf("sizeof(struct): %d\n", sizeof(t));
  printf("sum(sizeof(campuri)): %d\n",
      sizeof(t.a) + sizeof(t.b) + sizeof(t.c) + sizeof(t.d));

  //Bonus, afisarea pozitiilor fiecarui camp in structura. Ce se afiseaza
  //daca se mentine si atributul __attribute__((packed)) in declaratia
  //structurii?
  start = &t;
  offset = &t.a;
  printf("Offset a: %d\n", offset - start);

  offset = &t.b;
  printf("Offset b: %d\n", offset - start);

  offset = &t.c;
  printf("Offset c: %d\n", offset - start);

  offset = &t.d;
  printf("Offset d: %d\n", offset - start);

  return 0;
}
```

* [Medium] Definiţi macro-ul `for_each(vect, n, val)`, care să reprezinte antetul unei bucle de iteraţie prin elementele de tip `int` ale vectorului `vect`. `n` reprezintă numărul de elemente ale vectorului şi `val` numele unei variabile care să conţină valoarea curentă din iteraţie. Un exemplu de funcţie care foloseşte acest macro arată astfel:

```
void afiseaza(int *v, int n) {
  int crt_val;

  for_each(v, n, crt_val) {
    printf("%d ", crt_val);
  }
  printf("\n");
}
```

* [Medium] Implementaţi funcţiile:

```
int max(int a, int b);
int min(int a, int b);
```

care calculează maximul, respectiv minimul dintre două numere fără a folosi nici o comparaţie (instrucţiune `if` sau operatorul ternar). Scrieţi apoi un program care primeşte două numere întregi ca parametri `'în linia de comandă'` şi afişează maximul şi minimul folosind cele două funcţii definite mai sus.

* [Hard] Fie programul C simplu de mai jos:

```
int func() {
  return 1 + 2 + 3 + 4;
}

int main() {
  return func(10, 20, 30, 40);
}
```

Încercaţi să-l compilaţi. Ce observaţi? Ce explicaţie aveţi pentru acest comportament?

* [Medium] Implementaţi funcţiile:

```
int sort(void * , int (*f) (void *, void*));
//funcţie care sortează un vector si foloseste drept comparator functia f.

int comparator (void * a, void * b);
//funcţie care întoarce un număr negativ dacă a<b, 0 pentru a==b, un număr pozitiv pentru a>b, particularizată pentru numere întregi.
```

Scrieţi apoi un program care primeşte ca parametru al liniei de comandă un nume de fişier care este de forma:

N

A1 A2 A3 … An

unde:

N→ numărul de elemente ale vectorului

Ai → numere întregi.

'SOLUŢIE:'

```
#include <stdio.h>

//functia de sortare generica
int sort(void * v, int n, int size, int (*f) (void *, void*)) {
  int i,j;
  void *aux = malloc(size);
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
      //incrementarea pointerilor void* se face cu 1
      if (f(v + i*size, v + j*size) > 0) {
        memcpy(aux, v + i*size, size);
        memcpy(v + i*size, v + j*size, size);
        memcpy(v + j*size, aux, size);
      }
    }
  }

}

//functia care compara doua elemente date prin pointeri void*
int comparator (void * a, void * b) {
  return *((int*)a) - *((int*)b);
}


int main(int argc, char *argv[]) {
  if (argc < 2)
    return 1;

  FILE * f = fopen(argv[1], "r");

  //citire din fisier si alocare de memorie
  int i, *v, n;
  fscanf(f, "%d", &n);
  v = (int*)malloc(n * sizeof(int));

  for (i = 0; i < n; i++) {
    fscanf(f, "%d", &v[i]);
  }

  fclose(f);

  //afisare date initiale
```

```
  for (i = 0; i < n; i++) {
    printf("%d ", v[i]);
  }
  printf("\n");

  //sortare si afisare dupa sortare
  sort(v, n, sizeof(int), comparator);

  for (i = 0; i < n; i++) {
    printf("%d ", v[i]);
  }
}
```

* [Hardcore] Într-un fişier MP3 datele legate de titlul melodiei, artist, album, etc sunt stocate conform cu standardul ID3, într-o structură numită `tag ID3`. Această structură ocupă ultimii 128 octeţi din fişier (vezi descrierea detaliată în continuare). Prezenţa sa este determinată de primii 3 octeţi din tag. Astfel, dacă primii 3 octeţi din cei 128 conţin caracterele TAG atunci tagul ID3 este prezent. Altfel, se consideră că este absent.

Câmpurile unui tag ID3 sunt reprezentate în fişier pe ultimii 128 de octeţi, după cum urmează:

Field	Length
header	3
title	30
artist	30
album	30
year	4
comment	28 or 30
zero-byte	1
track	1
genre	1

Se cere să realizaţi un program care, primind în linia de comandă o serie de nume de fişiere mp3 să afişeze (în cazul în care există): # numele artistului # titlul melodiei # titlul albumului # anul înregistrării # genul melodiei Numele genurilor vor fi citite din fisierul `genres.dat`. Fişierul `genres.dat` conţine un număr de linii, fiecare linie conţinând indexul genului şi numele lui.

* [Medium] Scrieţi un program care primeşte ca parametru un nume de fişier şi îi afişează pe ecran conţinutul în hexazecimal, pe linii de câte 64 de caractere (asemănător cu utilitarul `hexdump` din Linux).

* [Medium] Scrieţi un program care permite căutarea unui cuvânt sau a unei măşti într-un text. Masca poate conţine caractere (care vor fi verificate ca atare) sau caracterul ? cu semnificaţia că poate fi înlocuit cu orice caracter. Textul se va citi dintr-un fişier al cărui nume va fi trimis ca parametru în linia de comandă. Se vor afişa cuvintele găsite şi linia pe care au fost găsite.

Teste recapitulative

Testul 1

1. (task1.c) Fie funcţiile definite astfel:

```
int functie(void);
int functie();
```

Sunt cele două definiţii echivalente? Scrieţi un program C care să justifice răspunsul dat.

2. (task2.c) Scrieţi un program care să afişeze propriul cod sursă.

3. (task3.c) Rezolvaţi următoarele cerinţe:

- 3.1 Definiţi structura `Numar_Complex`, cu câmpurile parte reală, parte imaginară.

- 3.2. Definiţi funcţia `Numar_Complex* citire (char *filename,int *n)`, care să citească din fişierul dat ca parametru numărul natural n, după care să aloce un vector de n numere complexe, care se vor citi apoi de pe următoarele n linii ale fişierului.

Formatul fisierului:

n x1 y1 x2 y2 … xn yn

- 3.3 Scrieţi un program care citeşte un vector de n numere complexe folosind funcţia de la punctul precedent, şi apoi afişează numărul de modul maxim şi poziţia acestuia în vector.

4. (task4.c) Scrieţi un program care poate primi (ca argumente în linia de comandă) opţiunile "-a" şi "-p". Programul primeşte ca prim parametru în linia de comandă un număr întreg n.

*Opţiunea "-a" va avea ca efect afişarea adresei variabilei din program în care se reţine numărul. *Opţiunea "-p" va avea ca efect afişarea pătratului numarului. Pentru a calcula pătratul numărului se va folosi o macroinscructiune. *Se permite rularea executabilului cu ambele opţiuni, caz în care se va afişa atât adresa variabilei cât şi pătratul valorii. La rularea executabilului fără opţiuni, nu se va afiţa nimic. *Se va afişa un mesaj de eroare în oricare dintre cazurile: lipsa parametrului n, parametrii nu sunt cei specificaţi mai sus, număr prea mare de parametri.

5. (task5.c) Scrieţi un program care interschimbă două variabile x şi y, fără a folosi o variabilă auxiliară.

6. Modificaţi următorul fişier makefile, astfel încât executabilul de la problema 4 să se numească task6.

```
#makefile

CC=gcc
CFLAGS=-Wall

all: task1 task2 task3 task4 task5

task1: task1.c
task2: task2.c
task3: task3.c
task4: task4.c
task5: task5.c
```

Testul 2

1. (task1.c) De ce urmatorul program nu afişează rezultatul corect/aşteptat?

```
#include <stdio.h>

int main()
{
  float f=0.0f;
  int i;
  for(i=0;i<10;i++)
    f = f + 0.1f;

  if(f == 1.0f)
    printf("f is 1.0 \n");
  else
    printf("f is NOT 1.0\n");

  return 0;
}
```

2. (task2.c) Scrieţi un program care să afişeze câţi biţi dintr-un număr reprezentat pe 4 octeţi sunt 1.

3. (task3.c) Rezolvaţi următoarele cerinţe:

- 3.1 Definiţi structura Numar_Real, cu câmpurile parte întreagă, parte fracţionară.
- 3.2 Definiţi funcţia `Numar_Real* citire(char *filename,int *n)`, care să citească din fişierul dat ca parametru dimensiunea vectorului v, să îl aloce dinamic, apoi să citeasca n numere reale, şi să le reţină într-un vector, sub formă de structuri definite ca la punctul precedent.

Formatul fişierului:

n nr_1 nr_2 … nr_n

- 3.3 Scrieţi un program care să testeze funcţia de citire de la punctul precedent, afişând numerele din vector în ordinea crescătoare a părţii fracţionare. Afişarea se va face în forma:

i1 f1 i2 f2 … in fn, unde i reprezintă partea întreagă, iar f partea fracţionară.

4. (task4.c) Scrieţi un program care primeşte (ca argumente în linia de comandă) două numere a şi b, şi poate primi una dintre opţiunile: "-m" sau "-f". * Veţi defini un macro care calculează minimul dintre a şi b, precum şi o funcţie cu acelaşi scop: <tt>int minim(int a, int b)</tt>. * Dacă este apelat cu optiunea "-m", programul va afisa minimul dintre a si b, folosind macrodefinitia. * Dacă este apelat cu optiunea "-f", programul va afişa adresa funcţiei <tt>minim</tt>. Dacă este apelat cu ambele optiuni, vor fi afişate ambele informaţii. * Programul va afişa un mesaj de eroare în oricare dintre următoarele cazuri: nu sunt daţi parametrii a şi b (sau unul dintre ei), parametrii nu sunt cei specificaţi mai sus, număr prea mare de parametri.

5. (task5.c) Scrieţi un program care afişează caracterul ';' (cod ASCII: 59), fără a folosi `'deloc' ';'` în sursa C. (task5.c)

6. Modificaţi următorul fişier makefile, adăugând o regulă `clean`, care să şteargă fişierele executabile, precum şi fisierele temporare, ale căror nume se termină cu '~'.

```
#makefile

CC=gcc
CFLAGS=-Wall

all: task1 task2 task3 task4 task5

task1: task1.c
task2: task2.c
task3: task3.c
task4: task4.c
task5: task5.c
```

Exemplu Examen [https://drive.google.com/open?id=1Qe3Zuu-LrPpHXWEuefwxz3lrqDAGCBQg]