Tutoriat 1 SO
Recapitulare C

1. C Quick Recap

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(){

   // Tipuri de date
   printf("Data types\n\n");

   char ch = 'A';
   printf("char = %c\n", ch);
   printf("char (as int) = %d\n", ch); // 65
   printf("Size = %lu bytes\n\n", sizeof(ch)); //sizeof returneaza
marimea tipului in bytes

   int integer = 5;
   printf("int = %d\n", integer);
   printf("Size = %lu bytes\n\n", sizeof(integer));

   long lo = 9223372036854775807; // max
   printf("long = %li\n", lo);
   printf("Size = %lu bytes\n\n", sizeof(lo));

   float fl = 5.8;
   printf("float = %f\n", fl);
   printf("Size = %lu bytes\n\n", sizeof(fl));

   double db = 4.7;
   printf("double = %lf\n", db);
   printf("Size = %lu bytes\n\n", sizeof(db));

   puts("-----------------------");
   // Mai multe informatii, precum celelalte tipuri disponibile,
```

```
   // marimile in bytes si valorile corespunzatoare se pot gasi
aici:
   // https://www.tutorialspoint.com/cprogramming/c_data_types.htm


   //
-------------------------------------------------------------------
-----


   // Pointeri
   printf("\nPointers\n\n");

   // Un pointer este o variabila a carei valoare reprezinta o
adresa de memorie
   // Forma generala de declarare: tip *nume
   // Adresa unei variabile se poate obtine prin operatorul &
   // Valoarea de la adresa retinuta de un pointer se poate extrage
folosind *pointer (a.k.a dereferencing)

   // Exemple de declarare:
   void *vptr1;              // pointer de tip void neinitializat,
are o valoare random
   printf("Adresa din vptr1 = %p\n", vptr1);

   void *vptr2 = NULL;       // pointer de tip void initializat cu
NULL
   printf("Adresa din vptr2 = %p\n", vptr2); // (nil)

   void *vptr3 = &integer;   // pointer de tip void care retine
adresa unui int numit integer
   // Pentru a accesa valoarea de la adresa retinuta in vptr3,
trebuie sa facem cast catre tipul actual (int*), inainte de
dereferentiere
   // Spre exemplu, urmatoarea linie genereaza o eroare:
   // printf("Valoarea int-ului de la adresa %p este %d\n", vptr3,
*vptr3);       // invalid use of void expression
   // Modalitatea corecta este urmatoarea:
   printf("Valoarea int-ului de la adresa %p este %d\n", vptr3,
*(int*)vptr3);
```

```
   int *ptr1 = NULL;         // pointer de tip int initializat cu
NULL
   int *ptr2;                // pointer de tip int neinitializat
   int *ip = &integer;       // pointer de tip int care retine
adresa unui int numit integer
   long *lp = &lo;           //                long
long      lo
   float *fp = &fl;          // ...
   double *dp = &db;         // ...

   printf("Valoarea int-ului de la adresa %p este %d\n", ip, *ip);
// valoarea se extrage folosind *pointer
   printf("Valoarea float-ului de la adresa %p este %f\n", fp, *fp);


   puts("------------------------");
   //
-------------------------------------------------------------------
-----


   // Arrays
   printf("\nArrays\n\n");

   int a[10];                  // declararea statica a unui array cu
10 elemente de tip int, initializate random
   int b[10] = {0};            //              --------//--------
cu 0
   int c[] = {1, 2, 3, 4, 5};

   for(int i = 0; i < 5; i++){
       printf("%d ", c[i]);
   }
   // se va afisa 1 2 3 4 5
   puts("");

   // putem afisa adresa fiecarui element
   for(int i = 0; i < 5; i++){
       printf("%p\n", &c[i]);
```

```
    }
    /*
    0x7ffd038c5d90
    0x7ffd038c5d94
    0x7ffd038c5d98
    0x7ffd038c5d9c
    0x7ffd038c5da0
    Observam cum cresterea este din 4 in 4 bytes (marimea unui int)
    */


    // Mai multe informatii:
    // https://www.tutorialspoint.com/cprogramming/c_arrays.htm


    puts("\n-----------------------");
    //
-------------------------------------------------------------------------
-----


    // Alocare dinamica
    printf("\nAlocare dinamica\n\n");

    // void* malloc( size_t size );
    // functia aloca dinamic un bloc de memorie cu marimea
specificata si returneaza un pointer de tip void
    // https://en.cppreference.com/w/c/memory/malloc

    int *v = (int*)malloc(sizeof(int) * 5); // Alocarea dinamica a
unui array de 5 intregi (20 bytes)
                                            // Nu este obligatoriu sa
facem cast, adica
                                            // int *v =
malloc(sizeof(int) * 5) produce acelasi rezultat

    for(int i = 0; i < 5; i++){
        v[i] = 2*i;
    }
```

```
    // Accesarea elementelor prin intermediul operatorului []
    for(int i = 0; i < 5; i++){
        printf("%d ", v[i]);
    }
    puts("");

    // Accesarea elementelor prin dereferentiere
    for(int i = 0; i < 5; i++){
        printf("%d ", *(v+i));
    }
    printf("\n\n");


    // Alocarea dinamica unei matrici de dimensiune NxM:
    int n = 2, m = 3;                           // evident, acestea
se cunosc la runtime
    int **M = (int **)malloc(n * sizeof(int *)); // array cu n
pointeri de tip int
    for(int i = 0; i < n; i++){
        M[i] = (int *)malloc(m * sizeof(int));   // fiecare dintre ei
aratand spre m intregi
    }
    // vizualizare: https://i.stack.imgur.com/FwiGs.jpg

    // initializare
    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++){
            M[i][j] = 5;
        }
    }
    // afisare
    for(int i = 0; i < n; i++){
        for(int j = 0; j < m; j++){
            printf("%d ", M[i][j]);
        }
        puts("");
    }

    // void free( void* ptr );
```

```
   // Deallocates the space previously allocated by malloc(),
calloc(), aligned_alloc(), (since C11) or realloc().
   // Sursa: https://en.cppreference.com/w/c/memory/free

   free(v); // eliberam memoria vectorului v

   // eliberam memoria matricei M
   for(int i = 0; i < n; i++){
       free(M[i]);
   }
   free(M);
   // Observatie: Daca eliberam mai intai vectorul de pointeri de
tip int (adica free(M))
   // pierdem adresele celor n vectori cu m elemente de tip int,
ceea ce inseamna ca nu
   // le mai putem da free



   // void* realloc (void* ptr, size_t size);
   // modifica marimea blocului de memorie spre care "pointeaza" ptr
   // functia poate muta blocul de memorie catre o locatie noua,
returnand noua adresa
   // continutul blocului de memorie este pastrat pana la minimul
dintre vechea si noua marime,
   // chiar daca blocul este mutat la alta locatie
   // Sursa: https://www.cplusplus.com/reference/cstdlib/realloc/

   // Exemplu:
   int *x = (int*)malloc(sizeof(int) * 5);
   for(int i = 0; i < 5; i++){
       x[i] = i+1;
   }
   // x = [1, 2, 3, 4, 5]
   x = realloc(x, sizeof(int) * 10);          // dublam marimea
blocului de memorie
   for(int i = 5; i < 10; i++){
       x[i] = i+1;                            // modificam ultimele
5 valori ale vectorului (nu le atingem pe primele 5)
   }
   for(int i = 0; i < 10; i++){
```

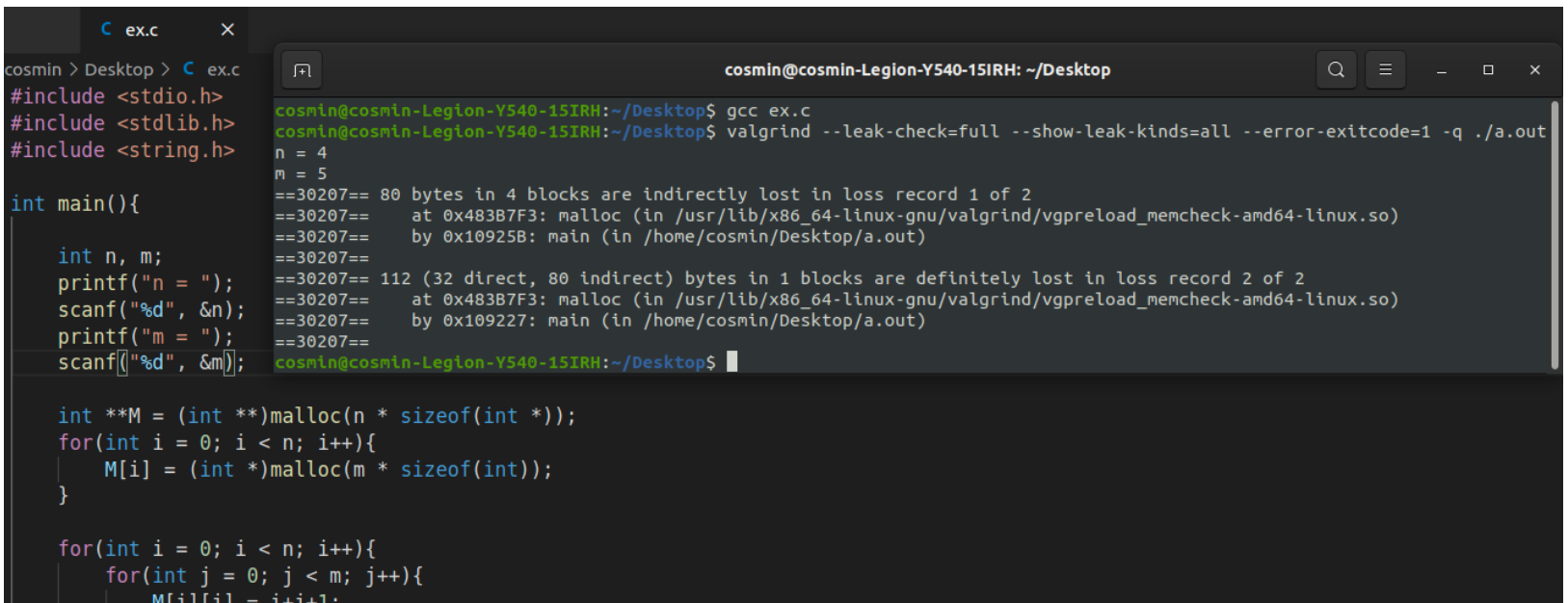
```
        printf("%d ", x[i]);
    }
    // x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]     // Observam cum
primele 5 valori au ramas neschimbate dupa realloc()
    free(x);


    puts("\n-----------------------");
    //
-------------------------------------------------------------------
-----


    // Structuri si pointeri:
    puts("\nStructuri si pointeri\n");

    struct Person {
        int age;
        char firstName[50];
        char lastName[50];
    };
    void printPerson(struct Person *person){
        printf("First name: %s\nLast name: %s\nAge: %d\n",
person->firstName,
               person->lastName, person->age);
    }

    struct Person *personPointer, person;
    personPointer = &person;

    person.age = 21;
    strcpy(person.firstName, "Ionescu");
    strcpy(person.lastName, "Andrei");

    // ambele au acelasi efect:
    printPerson(&person);
    printPerson(personPointer);

    // Alocarea dinamica a unui array de n persoane:
    struct Person *personArray;
```

```c
    printf("\n\nIntroduceti numarul de persoane\n");
    scanf("%d", &n);
    personArray = (struct Person*)malloc(n * sizeof(struct Person));

    // Observatie:
    // Pentru a accesa membrii unei structuri prin intermediul
pointerilor, folosim operatorul ->
    for(int i = 0; i < n; i++){
        printf("Introduceti numele persoanei %d\n", i+1);
        scanf("%s", (personArray + i)->firstName);
        printf("Introduceti prenumele persoanei %d\n", i+1);
        scanf("%s", (personArray + i)->lastName);
        printf("Introduceti varsta persoanei %d\n", i+1);
        scanf("%d", &(personArray + i)->age);
    }

    for(int i = 0; i < n; i++){
        printPerson(personArray + i);
    }
    free(personArray);

    return 0;
}
```

2. Valgrind

Instalare: **sudo apt-get install valgrind**

Comanda la rularea executabilului: **valgrind --leak-check=full --show-leak-kinds=all --error-exitcode=1 -q ./a.out**



3. Probleme de C

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/// Problema 1
/// Creati o variabila de tip int si un pointer, modificati valoarea variabilei folosind pointer-ul
void p1(){
   int x = 13;
   int *a = &x;
   *a = 15;
   printf("%d", x);
}

/// Problema 2
/// Creati un vector de o lungime citita
void p2(){
   int l;
   scanf("%d", &l);
   int *a = (int*)malloc(l * sizeof (int));
   for(int i=0;i<l;i++)
       a[i] = i+1;
   for(int i=0;i<l;i++)
       printf("%d ", a[i]);
   free(a);
}

/// Problema 3
/// Creati o matrice de 10x10 folosind pointeri
void p3(){
   int **m = (int**) malloc(5 * sizeof (int*));
   int i, j;
   for(i=0;i<5;i++)
       m[i] = malloc(5 * sizeof (int));
   m[1][2] = 4;
   for(i=0;i<5;++i) {
       for (j = 0; j < 5; ++j)
           printf("%d ", m[i][j]);
       printf("\n");
   }
   for(i=0;i<5;++i)
```

```
        free(m[i]);
    free(m);
}

/// Problema 4 - siruri de caractere
/// Cititi un text si un cuvant. Numarati de cate ori apare acel cuvant
in text
void p4(){
    unsigned long text_size = 255, word_size = 255;
    char *text = (char*) malloc(sizeof (char )*text_size);
    char *word = (char*) malloc(sizeof (char )*word_size);
    getline(&text, &text_size, stdin);
    getline(&word, &word_size, stdin);
    /// abc\0
    /// a\0
    word[strlen(word)-1] = '\0';
    text[strlen(text)-1] = '\0';
    int count = 0;
    /// ana are mere
    /// NULL
    char *buffer = strtok(text, " ");
    while (buffer!=NULL){
        if(strcmp(buffer, word)==0)
            count++;
        buffer = strtok(NULL, " ");
    }
    printf("Cuvantul apare de %d ori\n", count);
    free(text);
    free(word);
}

int main() {
    p4();
    return 0;
}
```

4. **Virtual Machine**
   Imagini pre-configurate de Ubuntu aici.

5. **Linux Commands**
   Cheat sheet pentru comenzi de linux aici.