

PROGRAMARE I

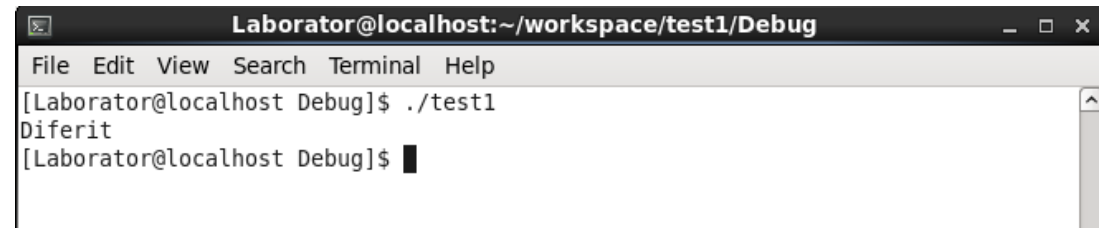
2016

Proba teoretica

6 probleme x 1.5p + 1p oficiu

Problema 1

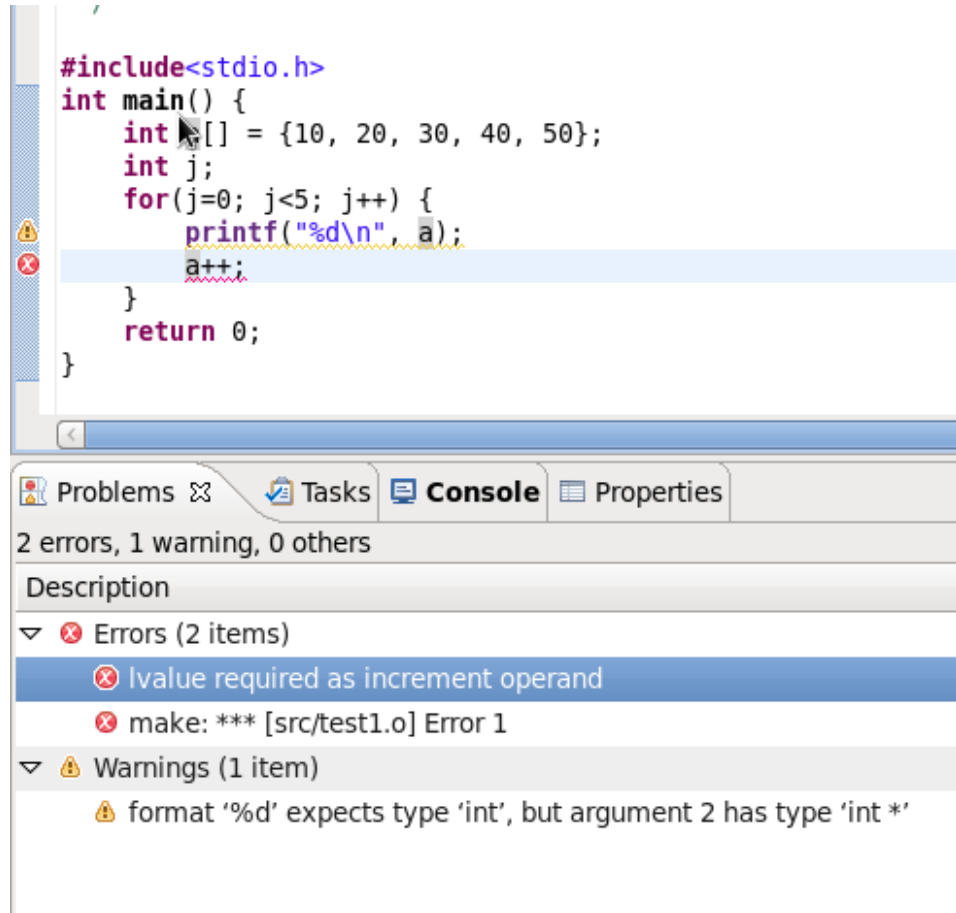
```
#include <stdio.h>
int main()
{
    char str1[] = "str[]";
    char str2[] = "str[]";
    if(str1 == str2)
    {
        printf("Egal\n");
    }
    else
    {
        printf("Diferit\n");
    }
    return 0;
}
```



```
Laborator@localhost:~/workspace/test1/Debug
File Edit View Search Terminal Help
[Laborator@localhost Debug]$ ./test1
Diferit
[Laborator@localhost Debug]$
```

Numele unui tablou este un pointer constant care are ca valoare adresa primului element din tablou
str1, str2 = adresele de inceput ale celor doua tablouri.
Sirurile au acelasi continut.
Adresele de inceput sunt insa diferite.

Problema 2



```
#include<stdio.h>
int main() {
    int a[] = {10, 20, 30, 40, 50};
    int j;
    for(j=0; j<5; j++) {
        printf("%d\n", a);
        a++;
    }
    return 0;
}
```

The screenshot shows a code editor with a C program. The program declares an array 'a' with values {10, 20, 30, 40, 50} and a loop that prints each element and increments 'a'. The 'a++' line is highlighted in blue. Below the code, the 'Problems' panel shows 2 errors and 1 warning. The first error is 'lvalue required as increment operand' for the 'a++' line. The second error is 'make: *** [src/test1.o] Error 1'. The warning is 'format '%d' expects type 'int', but argument 2 has type 'int *' for the 'printf' statement.

Programul contine erori.

Corect ar fi fost sa scriem:

`printf("%p\n",a+j); // afisare adresa`

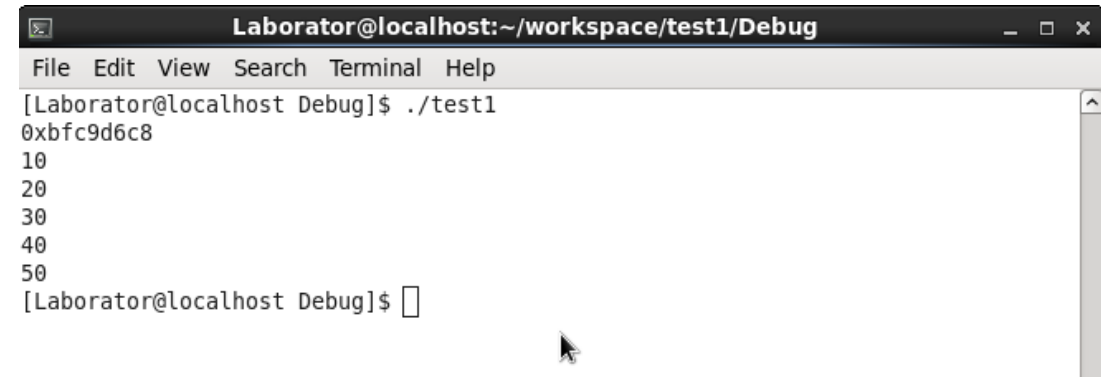
`printf("%d\n",*(a+j));//afisare elem`

- Numele unui tablou este un pointer constant care are ca valoare adresa primului element din tablou. `a++` nu este correct.

Problema 2

```
#include<stdio.h>
int main() {
    int a[] = {10, 20, 30, 40, 50};
    int j;
    printf("%p\n", a);
    for(j=0; j<5; j++)
    {
        printf("%d\n", *(a+j));
        //a++;
    }
    return 0;
}
```

```
#include <stdio.h>
int main()
{
    int a[]={10, 20, 30, 40, 50};
    int j;
    int *b;
    b= a;
    for(j=0;j<5;j++)
    {
        printf("%d\n",*b);
        b++;
    }
    return 0;
}
```



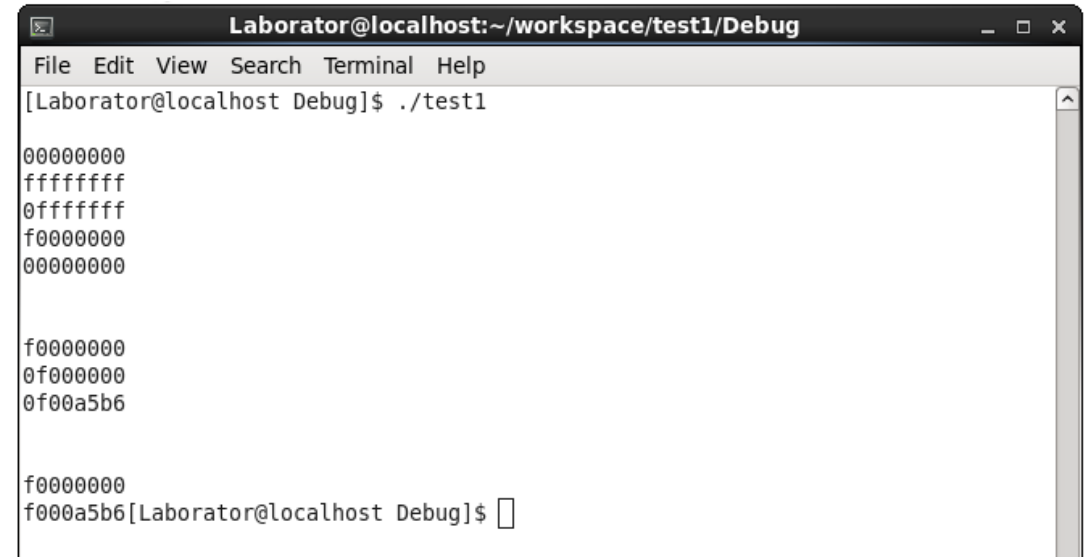
```
Laborator@localhost:~/workspace/test1/Debug
File Edit View Search Terminal Help
[Laborator@localhost Debug]$ ./test1
0xbfc9d6c8
10
20
30
40
50
[Laborator@localhost Debug]$
```

Problema 3

```
#include<stdio.h>
int main() {
    printf("\n%08x",0u);
    printf("\n%08x",~0u);
    printf("\n%08x",(~0u)>>4);
    printf("\n%08x",~((~0u)>>4));
    printf("\n%08x",~((~0u)>> 4) & 0xa5B6u);

    printf("\n\n");
    printf("\n%08x",~((~0u)>>4));
    printf("\n%08x",~((~0u)>>4))>>4);
    printf("\n%08x",~((~0u)>>4))>>4 | 0xA5b6u);

    printf("\n\n");
    printf("\n%08x",~((~0u)>>4));
    printf("\n%08x",~((~0u)>> 4) ^ 0xa5B6u);
    return 0;
}
```



```
Laborator@localhost:~/workspace/test1/Debug
File Edit View Search Terminal Help
[Laborator@localhost Debug]$ ./test1

00000000
ffffffff
0fffffff
f0000000
00000000

f0000000
0f000000
0f00a5b6

f0000000
f000a5b6[Laborator@localhost Debug]$
```

Problema 3

0u = 0 (unsigned)

sizeof(0u) = 4; //4 octeti=32 de biti

0xa5B6u = 0xA5b6u=0x0000A5B6u

~ NOT

& AND

| OR

^ XOR

>> 4 shift dreapta cu 4 pozitii, se completeaza cu 0-uri

pozitiile ramase libere

ATENTIE LA:

- Nr paranteze deschise si inchise
- Sa completati in stanga cu 0-uri pana la nr maxim de biti pe care este reprezentat numarul in cazul in care e nevoie
- Sa nu aveti cifra mai mare decat baza de reprezentare

int a= 0b1010; //baza 2 0b in fata

int b= 01010; //baza 8 0 in fata

int c= 0x1010; //baza 16 0x in fata

int d = 1010; //baza 10

printf("\n %08x %08x %08x %08x",a,b,c,d);

Int => 32 biti => 4octeti

a= 00 00 00 0A

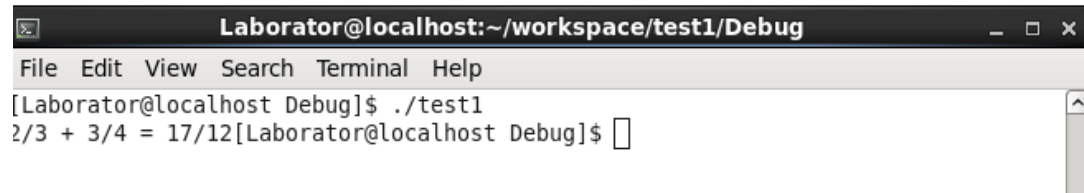
b= 001 000 001 000=00 00 02 08

c= 00 00 10 10

d= 11 1111 0010=00 00 03 f2

Problema 4

```
#include <stdio.h>
struct Fractie
{
    int a, b;
};
typedef struct Fractie Fractie;
Fractie suma(Fractie f1, Fractie f2)
{
    Fractie f;
    f.a = f1.a*f2.b + f1.b*f2.a;
    f.b=f1.b*f2.b;
    return f;
}
int main()
{
    Fractie f, f1, f2;
    f1.a=2;
    f1.b=3;
    f2.a=3;
    f2.b=4;
    f=suma(f1,f2);
    printf("%d/%d + %d/%d = %d/%d", f1.a, f1.b, f2.a, f2.b, f.a, f.b);
    return 0;
}
```



```
Laborator@localhost:~/workspace/test1/Debug
File Edit View Search Terminal Help
[Laborator@localhost Debug]$ ./test1
2/3 + 3/4 = 17/12[Laborator@localhost Debug]$
```

Declarația (prototipul) unei funcții

Declarația unei funcții se găsește într-un fișier header (sau antet) și este compusă din antetul funcției urmat de semnul de caracterul ;.

Prototipul unei funcții nu

"spune" cum prelucrază funcția informațiile primite, ci indică numai legaăturile sale cu exteriorul.

În cazul nostru:

Fractie suma(Fractie f1, Fractie f2);

Definiția unei funcții

Definiția unei funcții se va găsi întotdeauna într-un fișier cu cod sursă și este formată din antetul funcției urmat de un bloc în care se găsește algoritmul, exprimat în limbajul de programare C, care descrie modul în care funcția realizează prelucrările informației primite din exterior. Blocul începe întotdeauna cu caracterul { și se încheie cu }.

Fractie suma(Fractie f1,Fractie f2)

```
{
...
}
```

Problema 5

```
#include <stdio.h>
void f1(int *a, int n)
{
    int i;
    printf("n = "); scanf("%d", &n);
    for(i=0; i<n; i++)
    {
        scanf("%d", &a[i]);
    }
}
void f2(int *a, int *n)
{
    int i;
    printf("n = "); scanf("%d", n);
    for(i=0; i<*n; i++)
    {
        scanf("%d", a+i);
    }
}
void afisareVector(int *a, int n)
{
    int i;
    for(i=0; i<n; i++)
        printf("%d ", *(a+i));
}
int main(void)
{
    int a[10], n = 0;
    f1(a, n);
    printf("Vectorul are %d elemente\n", n);
    afisareVector(a, n);

    printf("\n\n\n");
    f2(a, &n);
    printf("Vectorul are %d elemente\n", n);
    afisareVector(a, n);
    return 0;
}
```

```
[Laborator@qualindser7 Debug]$ ./test1
n = 4
3
45
3
15
Vectorul are 0 elemente
```

```
n = 4
3
45
3
15
Vectorul are 4 elemente
3 45 3 15 [Laborator@qualindser7 Debug]$
```

Parametrii transferati unei functii in C pot fi valori sau adrese. Prima variant este numita si apel prin valoare, copie valoarea unui arg (param formal) al unei fctii. Modificarile efectuate asupra param nu au effect asupra argumentului.

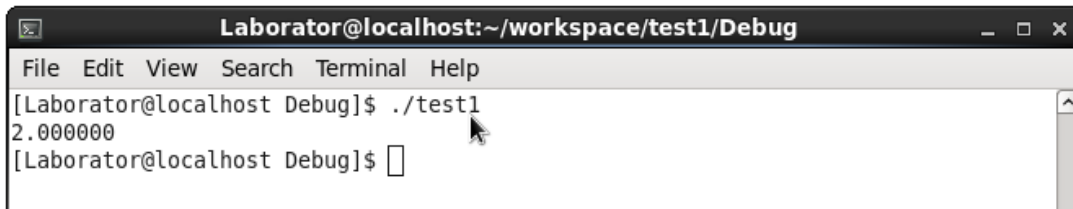
Cazul lui f1 n=0;

Al doilea mod de transfer este cel prin referinta(prin adresa). In acest caz se copie adresa argumentului, iar in interiorul functiei aceasta adresa este folosita pentru a avea acces la argumentul folosit efectiv la apelare, deci modificarile asupra argumentului il vor afecta.

Cazul lui f2 n se modifica din 0 in 4.

Problema 6.

```
#include <stdio.h>
int main (void)
{
    double x;
    int a = 7, b = 3;
    x = (int)1.5L * (double)(a/b);
    printf("%lf\n", x);
    return 0;
}
```



The screenshot shows a terminal window titled "Laborator@localhost:~/workspace/test1/Debug". The terminal has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The command prompt shows the execution of the program: `[Laborator@localhost Debug]$ ./test1`. The output is `2.000000`. The prompt then shows `[Laborator@localhost Debug]$` with a cursor.

(int)1.5L = 1
(a/b)=2
(double)(a/b)=2.000000

u (unsigned),
l (lowercase L) (long),
ll (long long),
ul (unsigned long),
ull (unsigned long long),
lu (long unsigned),
llu (long long unsigned)
and uppercase versions.