Ministerul Educaţiei, Culturii și Cercetării

al Republicii Moldova   
  
Universitatea Tehnică a Moldovei   
  
  
  
  
  
  
  
RAPORT

Lucrarea de laborator nr. 3   
la Programarea Calculatoarelor

A efectuat: st. gr. TI-211 Popa Cătălin   
  
A verificat: Stratulat Ștefan

UTM, Chișinău 2021

**Scopul lucrării**

Scopul lucrării este de a familiariza studentul cu scrierea și utilizarea funcțiilor și a recursiei, utilizînd limbajul C.

**Probleme pentru utilizarea funcțiilor:**

1. **Scrieți un program C pentru a găsi cubul oricărui număr folosind funcția.**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int cub(int a)

{

int cub;

cub=pow(a,3);

return cub;

}

int main()

{

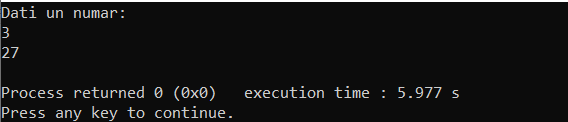
int a;

printf("Dati un numar: \n");

scanf("%d",&a);

printf("%d\n",cub(a));

}



**2. Scrieți un program C pentru a găsi diametrul, circumferința și aria cercului folosind funcțiile.**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int dia(int r)

{

return 2\*r;

}

int circum(int r)

{

return 2\*M\_PI\*r;

}

int aria(int r)

{

return 2\*M\_PI\*r\*r;

}

int main()

{

int r;

scanf("%d",&r);

printf("Diametrul este: %d\n",dia(r));

printf("---\n");

printf("Circumferinta este: %d\n",circum(r));

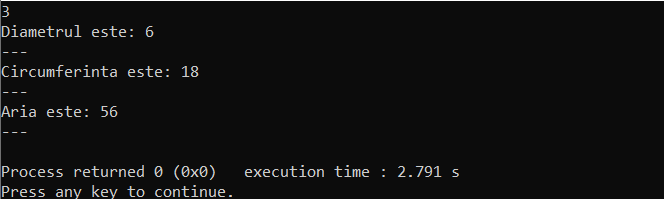
printf("---\n");

printf("Aria este: %d\n",aria(r));

printf("---\n");

return 0;

}



**3. Scrieți un program C pentru a găsi maxim și minim între două numere folosind funcții.**

#include <stdio.h>

#include <stdlib.h>

int max(int a, int b)

{

int max;

if(a>b)

{

max=a;

}

else

{

max=b;

}

return max;

}

int min(int a, int b)

{

int min;

if(a<b)

{

min=a;

}

else

{

min=b;

}

return min;

}

int main()

{

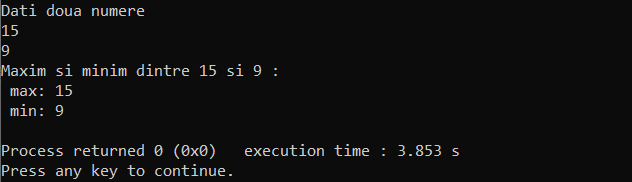
int a,b;

printf("Dati doua numere\n");

scanf("%d%d",&a,&b);

printf("Maxim si minim dintre %d si %d :\n max: %d\n min: %d\n",a,b,max(a,b),min(a,b));

}



**4. Scrieți un program C pentru a verifica dacă un număr este par sau impar folosind funcțiile.**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int par(int a)

{

int par;

if(a%2==0)

{

printf("Numarul este par\n");

}

else

{

printf("Numarul este impar\n");

}

return ;

}

int main()

{

int a;

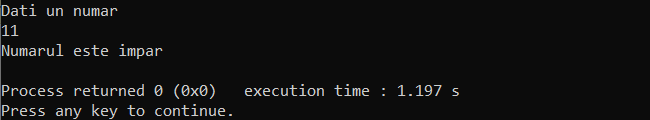
printf("Dati un numar\n");

scanf("%d",&a);

printf(" ",par(a));

return 0;

}



**5. Scrieți un program C pentru a verifica dacă un număr este prim, Armstrong sau un număr perfect folosind funcții.**

#include <stdio.h>

#include <stdlib.h>

int prim(int n)

{

int j;

for(j=2; j<=n/2; j++)

{

if(n%j!=0)

return 0;

else

return 1;

}

return 0;

}

int arm(int n)

{

int a,b,c;

a=n;

c=0;

while(n>0)

{

b=n%10;

c=c+(b\*b\*b);

n=n/10;

}

if(a==c)

printf("Este numar Armstrong\n",a);

else

printf("Nu este un numar Armstrong\n",a);

return 0;

}

int perfect(int n)

{

int s=0;

for(int j= 1;j<n;j++)

{

if(n%j==0)

s=s+j;

}

if(s==n)

printf("Este numar perfect\n",n);

else

printf("Nu este numar perfect\n",n);

return 0;

}

int main()

{

int n,a;

printf("Dati un numar \n");

scanf("%d",&n);

printf("---------------\n");

a=prim(n);

if(a==0)

{

printf("Este numar prim\n");

}

else

{

printf("Nu este numar prim\n");

}

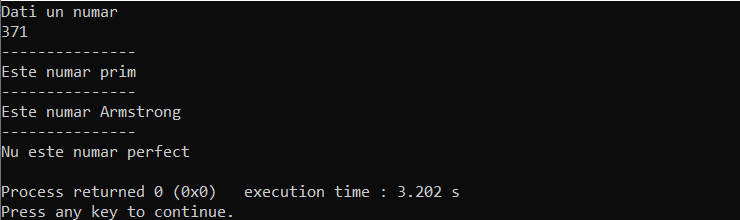
printf("---------------\n");

arm(n);

printf("---------------\n");

perfect(n);

}



**6. Scrieți un program C pentru a găsi toate numerele prime între intervalul dat folosind funcțiile.**

#include <stdio.h>

#include <stdlib.h>

int nprim(int a,int b)

{

int j,k;

for(j=a; j<=b; j++)

{

for(k=2; k<j; k++)

{

if(j%k==0)

{

break;

}

}

if(k>=j)

{

printf("|%d|\n",j);

}

}

return 0;

}

int main()

{

int a,b;

printf("Dati intervalul necesar: \n");

scanf("%d%d",&a,&b);

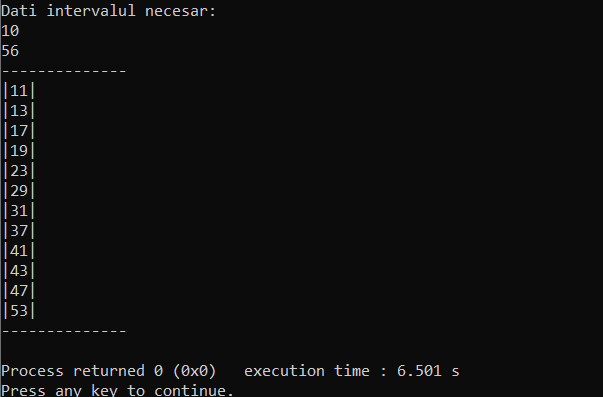
printf("--------------\n");

nprim(a,b);

printf("--------------\n");

return 0;

}



**7. Scrieți un program C pentru a imprima toate numerele puternice între intervalul dat folosind funcțiile.**

#include <stdio.h>

#include <stdlib.h>

int np(int a,int b)

{

int j,c;

for(j=a; j<=b; j++)

{

for(c=2; c<=j; c++)

{

if(j%c==0 && j%(c\*c)==0 )

{

printf("|%d|",j);

break;

}

}

}

return 0;

}

int main()

{

int a,b;

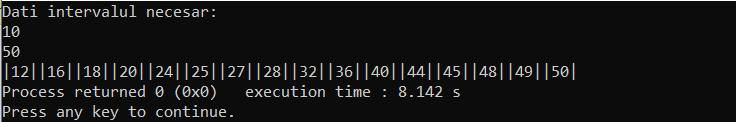
printf("Dati intervalul necesar: \n");

scanf("%d%d",&a,&b);

np(a,b);

return 0;

}



**8. Scrieți un program C pentru a imprima toate numerele Armstrong între intervalul dat folosind funcțiile.**

#include <stdio.h>

#include <stdlib.h>

int num(int a,int b)

{

int j,c,r,d=0,n;

for(j=a;j<=b;j++)

{

d=0;

c=j;

while(c>0)

{

n=c%10;

d=d+n\*n\*n;

c/=10;

}

if(d==j)

{

printf("|%d|\n",j);

}

}

return 0;

}

int main()

{

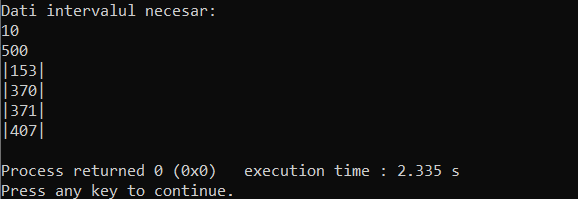
int a,b;

printf("Dati intervalul necesar:\n");

scanf("%d%d",&a,&b);

num(a,b);

return 0; }



**9. Scrieți un program C pentru a imprima toate numerele perfecte între intervalul dat folosind funcțiile.**

#include <stdio.h>

#include <stdlib.h>

int np(int c,int d)

{

int j,a,b;

for(j=c; j<=d; j++)

{

a=0;

b=j;

for(b=1; b<j; b++)

{

if(j%b==0)

{

a=a+b;

}

}

if(a==j)

{

printf("|%d|\n",j);

}

}

return 0;

}

int main()

{

int c,d;

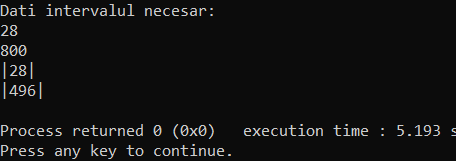
printf("Dati intervalul necesar: \n");

scanf("%d%d",&c,&d);

np(c,d);

return 0;

}



**Probleme pentru utilizarea funcțiilor recursive:**

**1. Scrieți un program C pentru a găsi puterea oricărui număr folosind recursivitatea.**

#include <stdio.h>

#include <stdlib.h>

int putere(int a,int n)

{

if(n!=0)

{

return (a\*pow(a,n-1));

}

else

{

return 1;

}

}

void main()

{

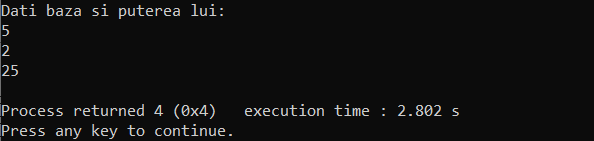
int a,n;

printf("Dati baza si puterea lui:\n");

scanf("%d%d",&a,&n);

printf("%d \n",putere(a,n));

}



**2. Scrieți un program C pentru a imprima toate numerele naturale cuprinse între 1 și n folosind recursivitatea.**

#include <stdio.h>

#include <stdlib.h>

int num(int n)

{

if(n)

{

num(n-1);

}

else

{

return 0;

}

printf("%d \n",n);

}

void main()

{

int n;

printf("Dati intervalul lui n:\n");

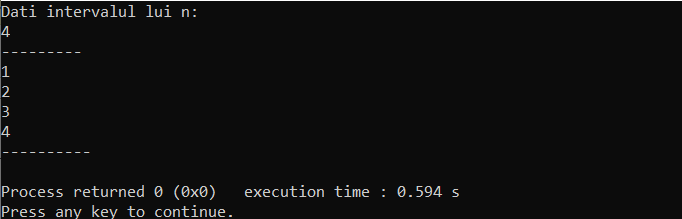
scanf("%d",&n);

printf("---------\n");

num(n);

printf("----------\n");

}



**3. Scrieți un program C pentru a imprima toate numerele pare sau impare în intervalul dat folosind recursivitatea.**

#include <stdio.h>

#include <stdlib.h>

void num(int a,int b)

{

if(a>b)

{

return;

}

printf(" |%d| \n",a);

num(a+2,b);

}

void main()

{

int a,b;

printf("Dati intervalul lui n: \n");

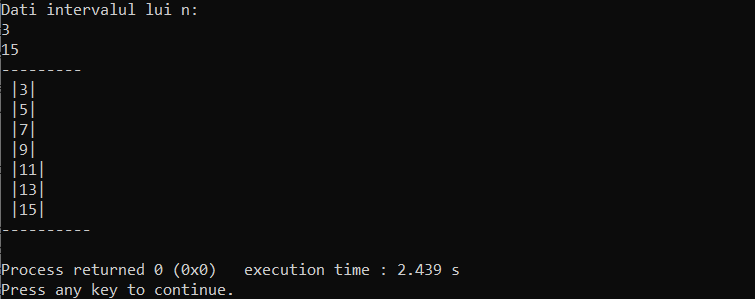
scanf("%d%d",&a,&b);

printf("---------\n");

num(a,b);

printf("----------\n");

}



**4. Scrieți un program C pentru a găsi suma tuturor numerelor naturale cuprinse între 1 și n folosind recursivitatea.**

#include <stdio.h>

#include <stdlib.h>

int suma(int n)

{

if(n==0)

{

return 0;

}

else

{

return (n+suma(n-1));

}

}

void main()

{

int n;

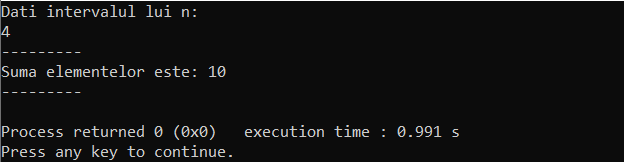
printf("Dati intervalul lui n:\n");

scanf("%d",&n);

printf("---------\n");

printf("Suma elementelor este: %d\n",suma(n));

printf("---------\n"); }



**5. Scrieți un program C pentru a găsi suma tuturor numerelor pare sau impare în intervalul dat folosind recursivitatea.**

#include <stdio.h>

#include <stdlib.h>

void main()

{

int a,b;

printf("Dati intervalul: \n");

scanf("%d%d",&a,&b);

printf("---------\n");

printf(" %d\n",num(a,b));

printf("----------\n");

}

int num(int a,int b)

{

if(a>b)

{

return 0;

}

else

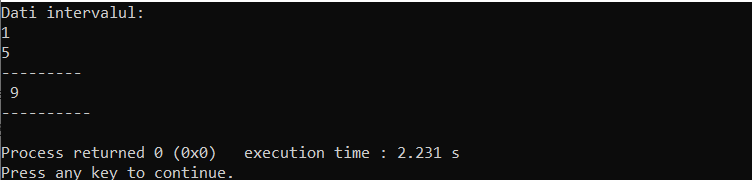
{

return (a+num(a+2,b));

}

return 0;

}



**6. Scrieți un program C pentru a găsi inversarea oricărui număr folosind recursivitatea.**

#include <stdio.h>

#include <stdlib.h>

int n=0,a;

int in(int b)

{

if(b)

{

a=b%10;

n=n\*10+a;

in(b/10);

}

else

{

return n;

}

return n;

}

int main()

{

int x;

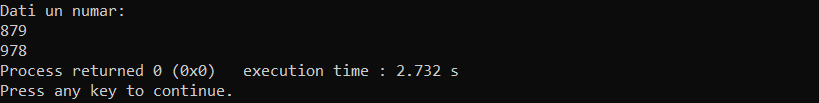
printf("Dati un numar: \n");

scanf("%d",&x);

printf("%d",in(x));

return 0;

}



**7. Scrieți un program C pentru a verifica dacă un număr este palindrom folosind recursivitatea.**

#include <stdio.h>

#include <stdlib.h>

int n=0,a,b;

int in(int d)

{

b=d;

if(d)

{

a=d%10;

n=n\*10+a;

in(d/10);

}

else return n;

if(b=n)

{

return n;

}

}

int main()

{

int n;

printf("Dati un numar: \n");

scanf("%d",&n);

if(n==in(n))

{

printf("Este palindrom\n");

}

else

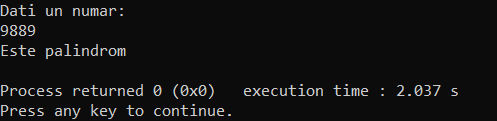
{

printf("Nu este palindrom\n");

}

return 0;

}



**8. Scrieți un program C pentru a găsi suma cifrelor unui număr dat folosind recursivitatea.**

#include <stdio.h>

#include <stdlib.h>

int s=0,a;

int suma(int n)

{

if(n)

{

a=n%10;

s+=a;

suma(n/10);

}

else return s;

return s;

}

int main()

{

int n;

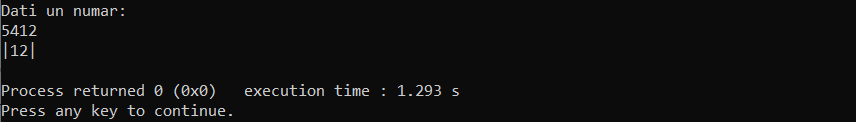
printf("Dati un numar: \n");

scanf("%d",&n);

printf("|%d|\n",suma(n));

return 0;

}



**9. Scrieți un program C pentru a găsi factorialul oricărui număr folosind recursivitatea.**

#include <stdio.h>

#include <stdlib.h>

int fact(int n)

{

if(n<=1)

{

return 1;

}

return n\*fact(n-1);

}

int main()

{

int n;

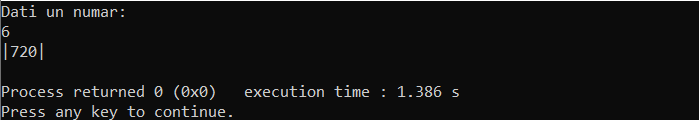
printf("Dati un numar: \n");

scanf("%d",&n);

printf("|%d|\n",fact(n));

return 0;

}



**10. Scrieți un program C pentru a genera al n-lea termen Fibonacci folosind recursivitate.**

#include <stdio.h>

#include <stdlib.h>

int fib(int x)

{

if(x==0)

{

return 0;

}

if(x==1)

{

return 1;

}

return fib(x-1)+fib(x-2);

}

int main()

{

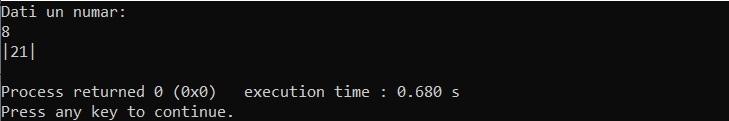
int n;

printf("Dati un numar: \n");

scanf("%d",&n);

printf("|%d|\n",fib(n));

return 0; }



**Rezultatul lucrării:**

* -ce reprezintă o funcție în limbajul C?

O functie reprezinta o secventa de instructiuni ce poate fi apelata de mai multe ori in cadrul unui program.

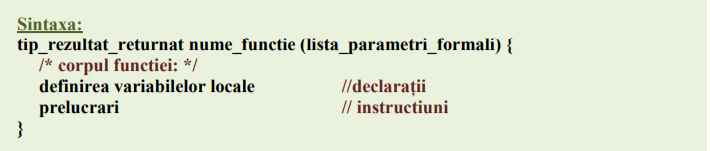
* cum se delcară o funcție în limbajul C?

Functia nu poate fi declarata in interiorul altei functii.

O functie are sintaxa generala astfel: tipul de date, numele functiei si lista declararii parametrilor.

Declaratia functiei este necesara compilatorului atunci cand este nevoie sa se resolve apelul functiei.

Daca functia a fost definite inaintea primului apel al sau, atunci declaratia nu mai este necesara.



* cum se apelează o funcție în limbajul C?

O functie se apeleaza prin specificarea numelui functiei, adica printr-o instructiune de apel.

* ce scop are instrucțiunea **return** pentru funcții în limbajul C?

Return intoarce, la finisarea functiei, noua valoare a variabilei.

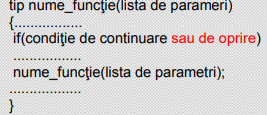
* ce reprezintă lista de parametri formali pentru o funcție în limbajul C?

Parametrii formali pentru o functie reprezinta lista de parametri care apare la definirea functiei.

* ce înseamnă o funcție recursivă?

O functie recursive reprezinta proprietatea unor notiuni de a se define prin ele insusi.

* cum se realizează în limbajul C o funcție recursivă?



* care este diferența între o funcție simplă și o funcție cu recursie?

Functiile de recursive se pot autoapela, pe cand cele obisnuite nu.

* de ce este o practică bună de a utiliza funcții în program ?

Functiile sunt practice in program deoarece ele iti usureaza lucrarea cu codul. Cu ele poti realiza aceeasi sarcina de mai multe ori chemand doar functia,dar nu sa scrii aceeasi secventa de cod de mai multe ori. Pe langa ceasta, ele ocupa si un spatiu mai mic, e destul sa o declari la inceput si dupa doar sa o apelezi.