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-214 Buza Cătălin  
  
A verificat: Prijilevschi Dumitru

UTM, Chișinău 2021

#include <stdio.h>

#include <stdlib.h>

#include<math.h>

int cub(int a)

{

return pow(a,3);

}

double diametru (double r)

{

const double pi=3.14159262;

return 2\*r;

}

double circumferinta (double r)

{

const double pi=3.14159262;

return 2\*r\*pi;

}

double arie (double r)

{

const double pi=3.14159262;

return r\*r\*pi;

}

int main()

{

printf("Probleme pentru utilizarea functiilor:\n");

printf("Problema nr.1:\nScrieti un program C pentru a gasi cubul oricarui numar folosind functia.");

int a1,c1;

a1=cub(2);

printf("\nCubul numarului este %d ",a1);

printf("\nProblema nr.2:\nScrieti un program C pentru a gasi diametrul, circumferinta si aria cercului folosind functiile.");

double d1,l1,a2;

d1=diametru(4);

l1=circumferinta(4);

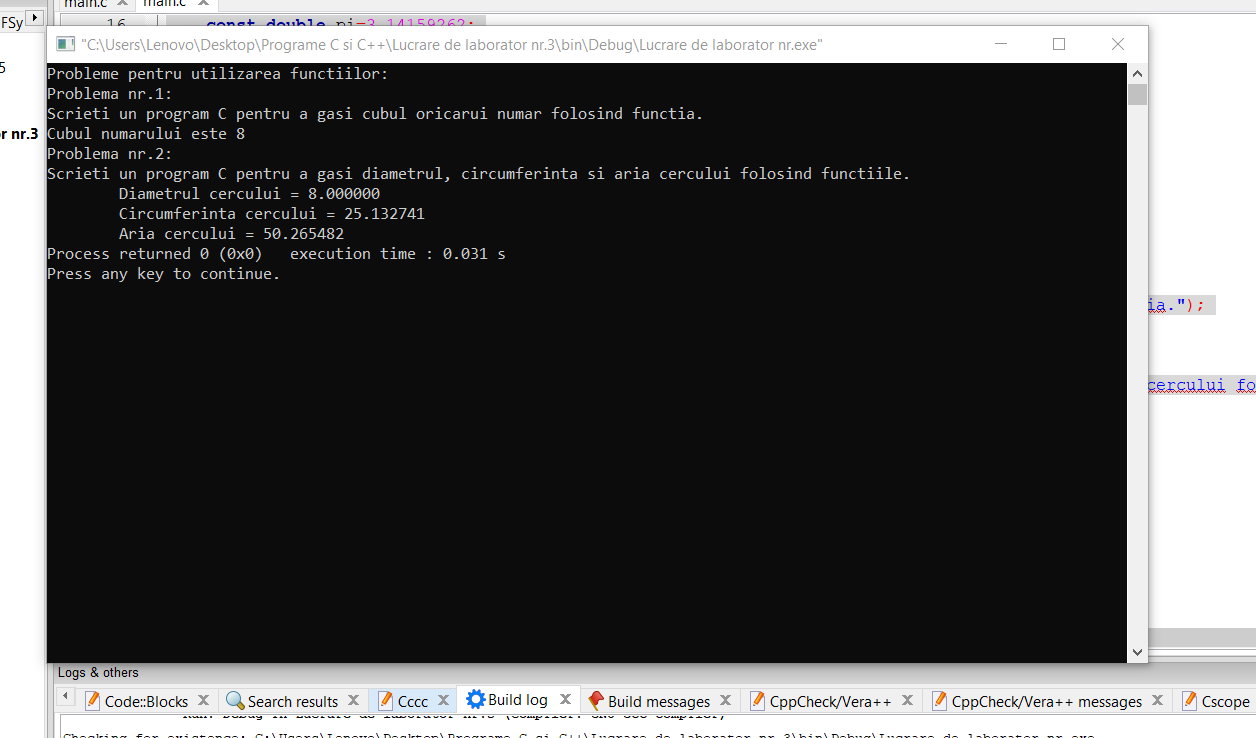
a2=arie(4);

printf("\n\tDiametrul cercului = %f", d1);

printf("\n\tCircumferinta cercului = %f", l1);

printf("\n\tAria cercului = %f",a2);

return 0;



#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include<conio.h>

double max(double a,double b)

{

double max=0;

if (a>b) max=a;

else max=b;

return max;

}

double min(double c,double d)

{

double min=0;

if(c>d) min=d;

else min=c;

return min;

}

int par(x1)

{

if (x1%2==0) printf("\nNumarul %d este par",x1);

else printf("\nNumarul %d nu este par",x1);

return x1;

}

int impar(x2)

{

if (x2!=0) printf("\nNumarul %d este impar",x2);

else printf ("\nNumarul %d nu este impar",x2);

return x2;

}

int numar\_perfect(int numar)

{

int d,s=1;

for(d=2;d<=numar/2;d++)

if(numar%d==0)

s=s+d;

if(numar==s) printf("\nNumarul %d este numar perfect",numar);

else printf("\nNumarul %d nu este numar perfect",numar);

}

int main()

{

printf("Problema 3:\nScrieti un program C pentru a gasi maxim si minim intre doua numere folosind functii.");

double a1,x=10,y=11;

a1=max(x,y);

printf("\nMaximul dintre numerele %lf si %lf este %lf ",x,y,a1);

double a2,b2=10,c2=5;

a2=min(b2,c2);

printf("\nMinimul dintre numerele %lf si %lf este %lf",b2,c2,a2);

printf("\nProblema 4:\nScrieti un program C pentru a verifica daca un numar este par sau impar folosind functiile.");

int n,n1=222;//numar par//

n=par(n1);

int m,m1=5;//numar impar//

m=impar(m1);

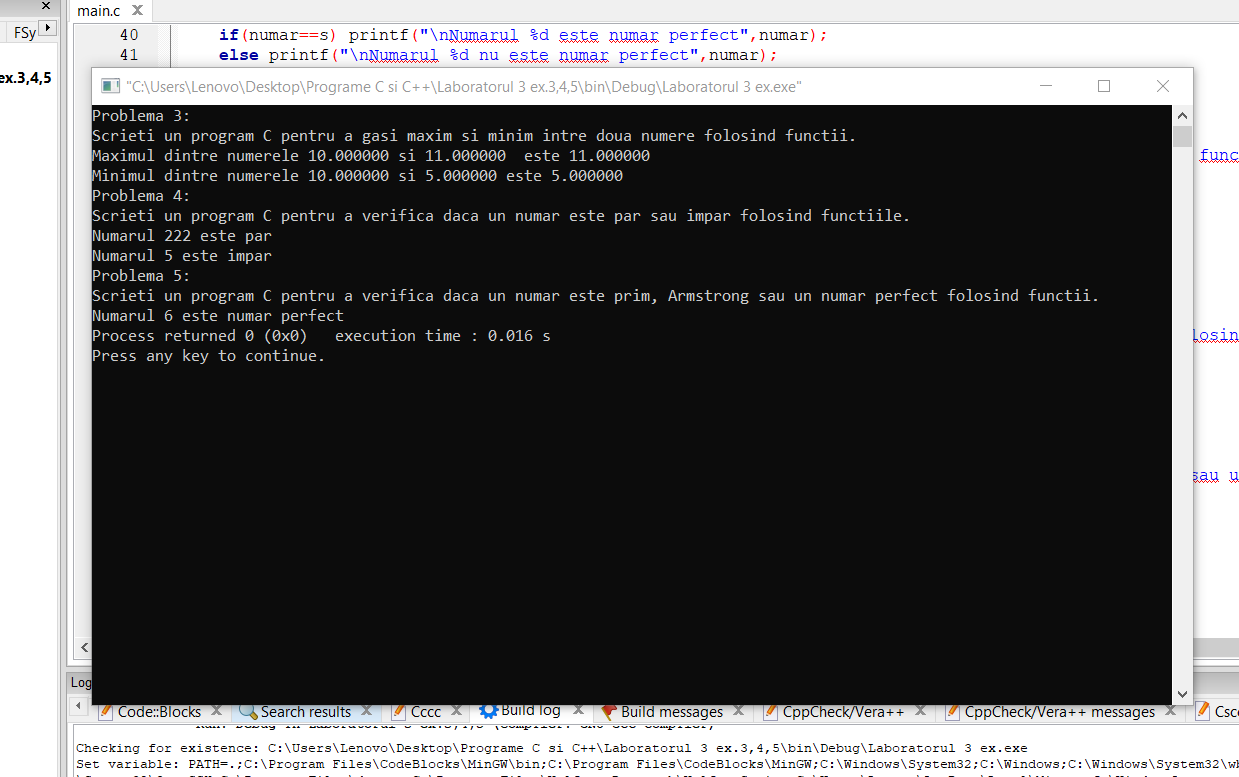
printf("\nProblema 5:\nScrieti un program C pentru a verifica daca un numar este prim, Armstrong sau un numar perfect folosind functii. ");

int numar1=6,perfect;

perfect=numar\_perfect(numar1);

return 0;

}



#include <stdio.h>

#include <stdlib.h>

#include<stdbool.h>

int interv\_numar\_prim(int a,int b)

{

int c,i;

bool q;

for(i=a;i<=b;i++){

q=true;

for(c = 2; c \* c <= i; c++)

if(i % c == 0)

{

q = false;

break;

}

if(q) printf("\nNumerele prime sunt :%d",i);

}

}

int interv\_numar\_puternic(int n1,int n2)

{

for(int i=n1;i<=n2;i++)

{

int k=i,fact=1,sum=0,r;

while(k!=0)

{

r=k%10;

fact=factorial(r);

k=k/10;

sum=sum+fact;

}

if(sum==i){

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

}

sum=0;

}

return 0;

}

int factorial(int f)

{

int mul=1;

for(int i=1; i<=f;i++)

{

mul=mul\*i;

}

return mul;

}

int main()

{

printf("Problema 6:\nScrieti un program C pentru a gasi toate numerele prime intre intervalul dat folosind functiile. ");

int a1=3,b1=7,c1;

c1=interv\_numar\_prim(a1,b1);

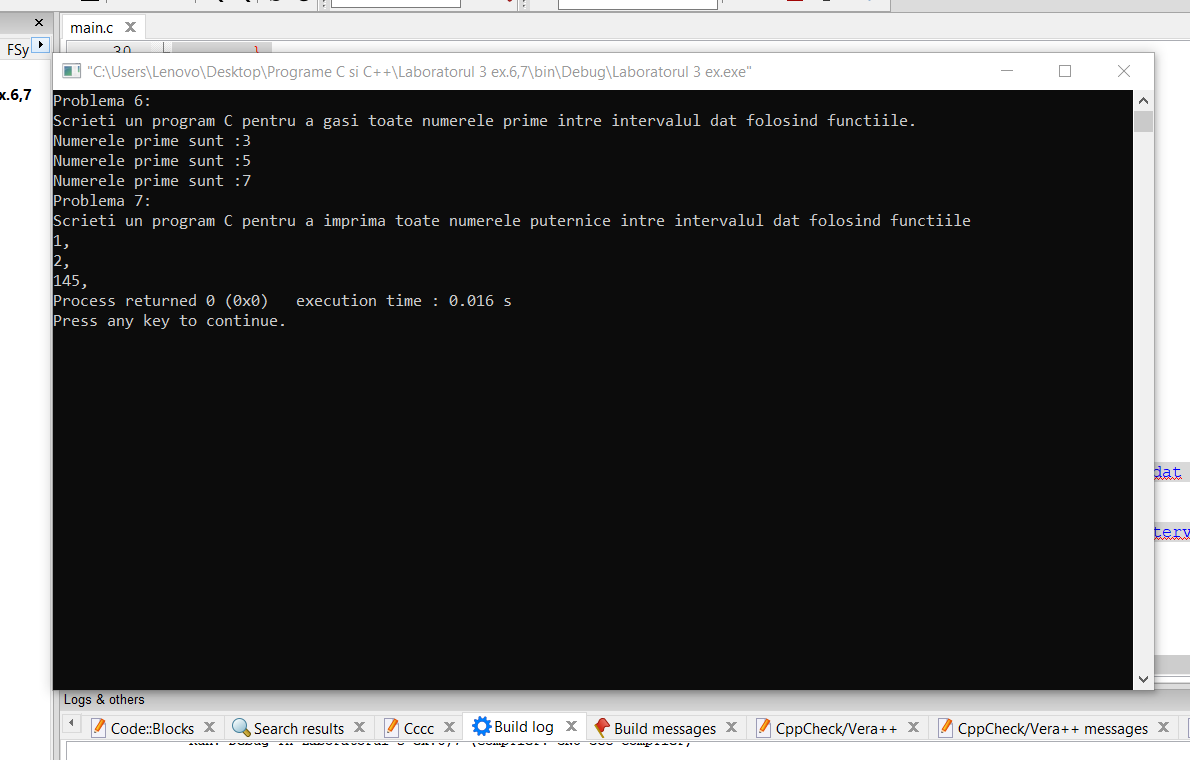
printf("\nProblema 7:\nScrieti un program C pentru a imprima toate numerele puternice intre intervalul dat folosind functiile");

int d1=1,d2=300,d3;

d3=interv\_numar\_puternic(d1,d2);

return 0;

}



#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <math.h>

int interv\_num\_armstrong(int x,int y)

{

int i1,rem,sum=0,entr,nr,i,m=1,z=0;

for(i1=x; i1<=y; i1++){

entr = i1;

nr=i1;

while(nr>0){

nr = nr / 10;

z+=1;

}

nr=i1;

while(nr!=0){

rem = nr % 10;

nr = nr / 10;

for(i=1;i<=z;i++){

m = m\*rem;

}

sum +=m;

m=1;

}

if(sum == entr)

printf("%d ",sum);

sum=0;

z=0;

}

}

int interv\_num\_perfecte(int m,int n )

{

int num,i,Sum;

for(num = m;num <= n; num++)

{

for(i = 1, Sum = 0; i < num; i++)

{

if(num %i == 0)

Sum = Sum + i;

}

if(Sum == num)

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

}

}

int main()

{

printf("Problema 8:\nScrieti un program C pentru a imprima toate numerele Armstrong intre intervalul dat folosind functiile.");

int x1=1,x2=200;

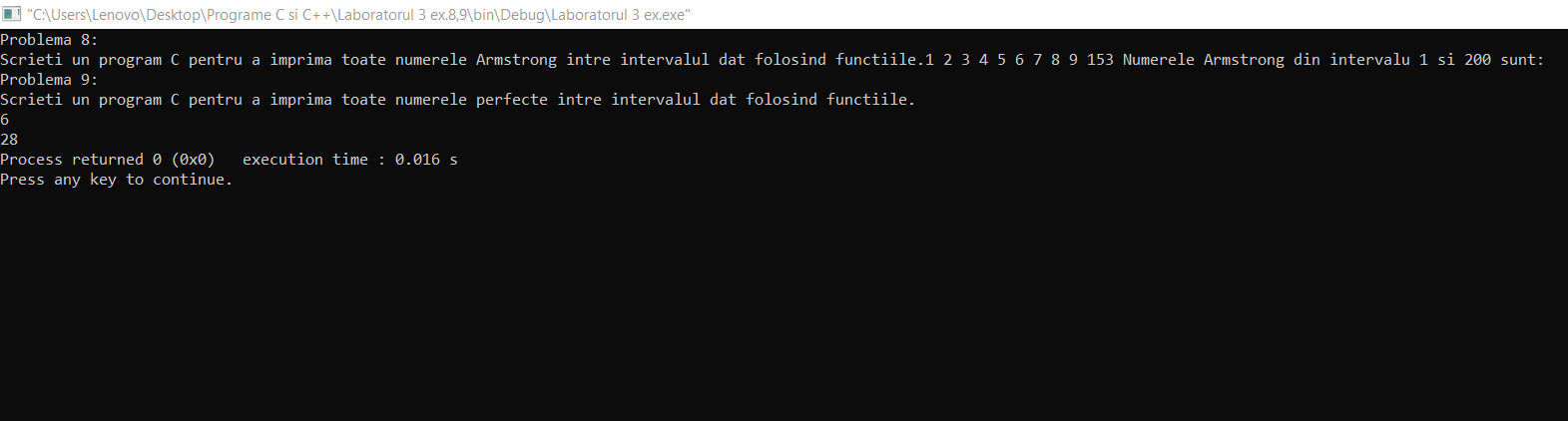
printf("\Numerele Armstrong din intervalu %d si %d sunt:",x1,x2,interv\_num\_armstrong(x1,x2));

printf("\nProblema 9:\nScrieti un program C pentru a imprima toate numerele perfecte intre intervalul dat folosind functiile.");

int m1=1,n1=100,a;

a=interv\_num\_perfecte(m1,n1);

return 0;

}

#include <stdio.h>

#include <stdlib.h>

int putere(int m,int m3)

{

return m3\*pow(m3,m-1);

}

int afisare (int n)

{

if(n == 0)

return 0;

else

{

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

return afisare(n-1);

}

}

int par(int x,int y)

{

if(x > y) return 0;

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

par(x+2, y);

}

int main()

{ // recursivitatea //

printf("Problema nr.1\nScrieti un program C pentru a gasi puterea oricarui numar folosind recursivitatea");

int m1=2,o,m2=5;

o=putere(m1,m2);

printf("\nPuterea lui %d la %d este %d",m2,m1,o);

printf("\nProblema nr.2\nScrieti un program C pentru a imprima toate numerele naturale cuprinse intre 1 si n folosind recursivitatea");

int n1=5,n2;

n2=afisare(n1);

printf("\nProblema nr.3\nScrieti un program C pentru a imprima toate numerele pare sau impare in intervalul dat folosind recursivitatea. ");

int x1=2,y1=10;

par(x1,y1);

return 0;}

#include <stdio.h>

#include <stdlib.h>

int suma(int n)

{

if(n==0) return 0;

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

}

int factorial(int n2)

{

if (n2==0) return 1;

else return n2\*factorial(n2-1);

}

int fibonacci (int m)

{

if (m==0) return 0;

else if (m == 1) return 1;

else return fibonacci(m-1)+fibonacci(m-2);

}

int main()

{

printf("\nEx.4:\nScrieti un program C pentru a gasi suma tuturor numerelor naturale cuprinse intre 1 si n folosind recursivitatea");

int a1=1,n1=10;

suma (n1);

printf("\nSuma numerelor de la %d la %d este %d",a1,n1,suma(n1));

printf("\nEx.9:\nScrieti un program C pentru a gasi factorialul oricarui numar folosind recursivitatea.");

int n3=5;

factorial(n3);

printf("\nFactorialul %d este %d",n3,factorial(n3));

printf("\nEx.10:\nScrieti un program C pentru a genera al n-lea termen Fibonacci folosind recursivitate");

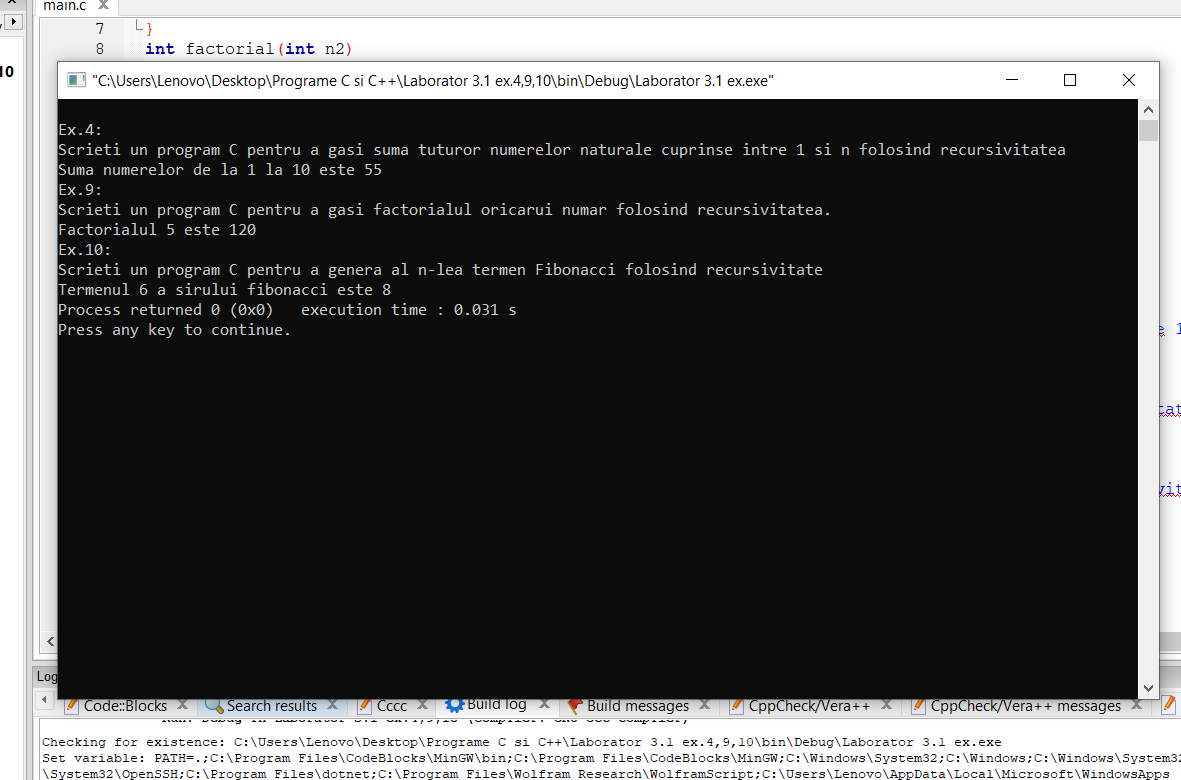
int m1=6;

fibonacci(m1);

printf("\nTermenul %d a sirului fibonacci este %d",m1,fibonacci(m1));

return 0;

}



Concluzie:În timpul efectuării lucrări de laborator nr. 3 am înteles conceptul de ***funcție*** și ***funcții recursive*** în limbajul C***. Funcţiile*** sînt elementele de bază ale unui program scris în C: orice program, de orice dimensiune, constă din una sau mai multe funcţii, care specifică operaţiile care trebuie efectuate. ***Funcţiile*** sînt într-adevăr singurul mod de a face faţă complexităţii programelor mari, permit desfacerea programelor mari în module mai mici, şi dau utilizatorului posibilitatea de a dezvolta programe, folosind ceea ce alţii au făcut deja, în loc să o ia de la început. ***Funcţiile*** în C pot fi definite recursiv, deci o funcţie se poate apela pe ea însăşi fie direct, fie indirect. În general însă recursivitatea nu face economie de memorie, deoarece trebuie menţinută o stivă cu valorile de prelucrat. În limbajul C orice funcţie este apelată prin numele ei, urmat de lista reală a argumentelor, închisă între paranteze. Revenirea dintr-o funcţie se face prin intermediul unei instrucţiuni ***return***. Valoarea pe care o funcţie o calculează poate fi returnată prin instrucţiunea return, care după cum am văzut are două formate. Ca argument al acestei instrucţiuni poate apărea orice expresie admisă în C. Funcţia returnează valoarea acestei expresii funcţiei apelante. O funcţie de tip ***void*** nu returnează în mod explicit o valoare. O instrucţiune return, fără expresie ca parametru, cauzează doar transferul controlului funcţiei apelante.