FUNCȚII DE BIBLIOTECĂ IN LIMBAJUL C/C++

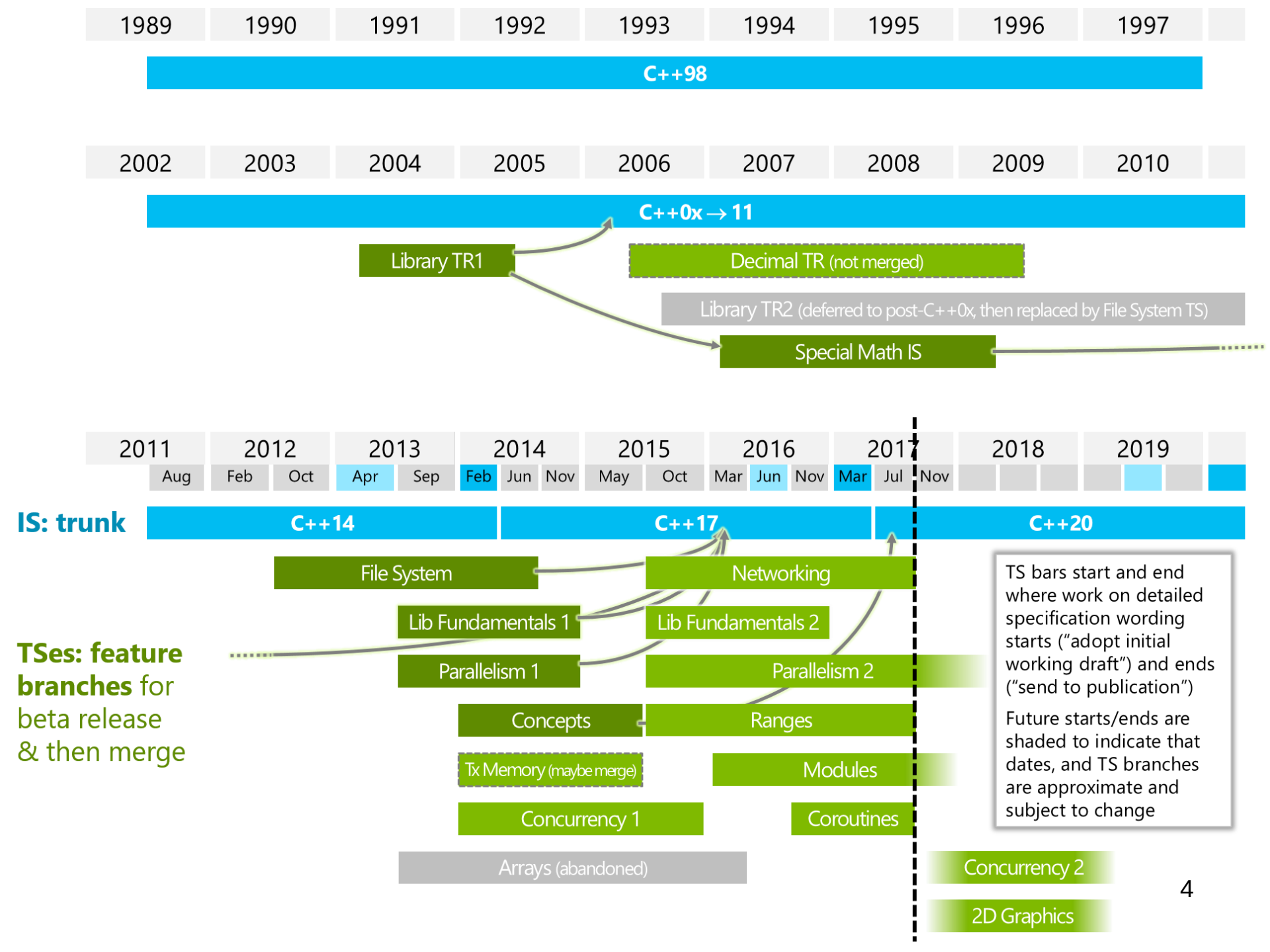
Cuprins

1. Noţiuni introductive

- C++0x(1y/2z) e un limbaj ce dorește dezvoltarea limbajului C++
  - există variante 2003, 2007(8), 2011(2), 2014, 2017
  - se preconizează o noua varianta în 2020 (C++2z).
- C++20 se află în prezent în revizii finale, după ce a fost aprobat un proiect pe 4 septembrie 2020. Următorul standard planificat este C ++ 23.
- Biblioteca C++ standard în noile variante conține:
  - Biblioteca C++ standard preexistentă (din C) și adăugiri, în total 51 fișiere header
  - Componente de bază pentru Intrări/Ieșiri C++, suport pentru calcul numeric și internaționalizare
  - Biblioteca STL
- Având în vedere evoluția altor limbaje de programare, C++ oferă acum noi facilități, inclusiv elemente de programare funcțională bazate pe funcții lambda, expresii regulare, etc.

- **Download Visual Studio:**

https://visualstudio.microsoft.com/downloads/


1989
1990
1991
1992
1993
1994
1995
1996
1997
C++98
2002
2003
2004
2005
2006
2007
2008
2009
2010
C++0x → 11
Library TR1
Decimal TR (not merged)
Library TR2 (deferred to post-C++0x, then replaced by File System TS)
Special Math IS
2011
2012
2013
2014
2015
2016
2017
2018
2019
Aug
Feb
Oct
Apr
Sep
Feb
Jun
Nov
May
Oct
Mar
Jun
Nov
Mar
Jul
Nov
IS: trunk
C++14
C++17
C++20
File System
Networking
Lib Fundamentals 1
Lib Fundamentals 2
TSes: feature branches for beta release & then merge
Parallelism 1
Parallelism 2
Concepts
Ranges
Tx Memory (maybe merge)
Modules
Concurrency 1
Coroutines
Arrays (abandoned)
Concurrency 2
2D Graphics
TS bars start and end where work on detailed specification wording starts ("adopt initial working draft") and ends ("send to publication")
Future starts/ends are shaded to indicate that dates, and TS branches are approximate and subject to change

- Biblioteca C++ standard
  - are 51 de fișiere header cu constante, macrofuncții, prototipuri de funcții, tipuri de date
  - se află în spațiul de nume *std*
- Headerele standard C (18) sunt redenumite în C++ astfel:
  - în loc de <stdio.h>, <stdlib.h>, <math.h>, etc.
  - avem <cstdio>, <cstdlib>, <cmath>, etc.
  - majoritatea compilatoarelor acceptă și versiunea veche ca *deprecated*

- Deși limbajul C nu este un limbaj orientat pe obiecte, biblioteca standard C este cea mai refolosită din domeniul programării aplicațiilor software

- De mai bine de 30 ani, aceste funcții de bibliotecă se folosesc în majoritatea aplicațiilor C/C++
- In Visual Studio C++ pentru a folosi standardul initial al functiilor de biblioteca se foloseste declaratia preprocessor:

#define _CRT_SECURE_NO_WARNINGS

- Biblioteca C
  - conține funcții și tipuri de date
  - constă din 18 fișiere de tip header – care pot fi clasificate pe grupe de funcții

2. Funcții pentru intrări-ieșiri (<cstdio>/<stdio.h>, <conio.h>)

- Generale
  - *printf( )/scanf( )/scanf_s( )/…*
  - *fprintf( )/fscanf( ) (fscanf_s(…))*
  - *…*
- Pentru caractere
  - *getchar( )/putchar( )*
  - *(f)getc( )/(f)putc( )*

  La nivel de consola (fara validare):
  - *getch( )/putch( ) (_getch( ))*
  - *getche( ) (_getche( ))*
- Pentru șiruri de caractere
  - *gets( )/puts( ) (gets_s(…))*
- Compilatoarele noi au introdus variante *wide* la functiile *printf( ), scanf( ), scanf_s( )* pentru caractere/siruri *Unicode*

VC++ *scanf_s( )* si alte facilitati

- Citește date formatate din fluxul standard de intrare. Aceste versiuni de *scanf_s( ), _scanf_l( ), wscanf_s( ), _wscanf_l( )* au îmbunătățiri de securitate, așa cum este descris în Caracteristicile de securitate din CRT (C Run-Time library (CRT)).
- *int scanf_s(const char *format [,argument]...);*
- *int _scanf_s_l(const char *format, locale_t locale [, argument]...);, etc.*

Parametrii:

- *format*, format șir de control.
- *argument*, argumente optionale
- *locale*, variabile locale folosite
- Exemplu:

char s[10];

scanf_s("%9s", s, (unsigned)_countof(s)); // buffer size is 10, width specification is 9

#define _CRT_SECURE_NO_WARNINGS

Directiva este utilizată pentru a considera funcția C/C++ *scanf( )* standard fără avertismente sau erori de securitate

```
//ASCII  code for characters, wide_characters
#define _CRT_SECURE_NO_WARNINGS
//#include <cstdio>
#include <cwchar> // Header file containing wide functions and types

int main( )
{
        char car;
        wchar_t lcar;
        printf("\nIntroduceti un caracter: ");
        scanf(" %c", &car);
        //scanf_s(" %c", &car,1);
        printf("\n Codul ASCII al caracterului introdus:\n \tzecimal:
%d\n\thexa: %x, \t lungimea= %d\n", car, car, sizeof(char));
        wprintf(L"\nIntroduceti un wide caracter: ");
        wscanf(L" %lc", &lcar);
        //wscanf_s(L" %lc", &lcar,2);
        wprintf(L"\n Codul ASCII al wide caracterului introdus:\n \tzecimal:
%d\n\thexa: %x, \t lungimea= %d\n", lcar, lcar, sizeof(lcar));
        return 0;
}//end main
```

```
// This program may use the scanf_s and wscanf_s functions on VC++
// to read formatted input.
#define _CRT_SECURE_NO_WARNINGS
//#include <cstdio>//also, contains wide functions in VC++
#include <cwchar> // Header file containing wide functions and types
#include <cstdlib>
const int dim = 80;

int main( ) {
        int i, result;
        float fp;
        char c, s[dim];
        wchar_t  wc, ws[dim];
        printf("Enter the following data: int, float, char, wide-char, string, wide-string\n");
        //result = scanf_s("%d %f %c %C %s %S", &i, &fp, &c, 1, &wc, 1, s,
(unsigned)_countof(s), ws, (unsigned)_countof(ws));
        result = scanf("%d %f %c %C %s %S", &i, &fp, &c, &wc, s, ws);
        printf("The number of fields input is %d\n", result);
        printf("The contents are: %d %f %c %C %s %S\n", i, fp, c, wc, s, ws);
        wprintf(L"Enter the following data: int, float, char, wide-char, string, wide-string\n");
        //result = wscanf_s(L"%d %f %hc %lc %S %ls", &i, &fp, &c, 1, &wc, 2, s,
(unsigned)_countof(s), ws, (unsigned)_countof(ws));
        result = wscanf(L"%d %f %hc %lc %S %ls", &i, &fp, &c, &wc, s, ws);
        wprintf(L"The number of fields input is %d\n", result);
        wprintf(L"The contents are: %d %f %C %c %hs %s\n", i, fp, c, wc, s, ws);
}//main
```

Alte facilitati *scanf( )*

- În C ++ 0x /1y/2z este posibilă utilizarea expresiilor regulare pentru a impune un proces de citire capabil să fie oprit nu printr-un spațiu alb (se vor detalia ulterior).
- *char name[20]="";*

  *scanf ("%[^\n]%*c", name);*
- Intre *[ ]* este setul de caractere de scanare, *[^\n]* spune că, în timp ce intrarea nu este o linie nouă *('\n'*), introduceți intrarea, caractere.
- *%*c* – caracterul * indică faptul că se extrage din flux caracterul newline (deci nu va afecta citiri ulterioare), dar acesta nu va fi memorat.

  scanf("%[^\n]s",name);

[^\n] stabilește doar delimitatorul pentru șirul scanat.

Observație: După citirea unui alt tip de date (numeric), se va ignora ultimul caracter cu *cin.ignore( )* pentru a citi alte date sir.

```
//Regular strings with other data
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
using namespace std;
#define DIM 20

int main( ) {
        int mark1, mark2;
        char name[DIM] = "";
        printf("\nEnter a string with space: ");
        scanf("%[^\n]%*c", name);
        printf(name);
        printf("\nEnter a new string with space: ");
        scanf("%[^\n]s", name);
        printf(name);

        printf("\nEnter an int: ");
        scanf("%d", &mark1);
        printf("\nThe mark is = %d ", mark1);
        printf("\nEnter another int: ");
        scanf("%d", &mark2);
        printf("\nThe mark is = %d ", mark2);

        cin.ignore( );
        printf("\nEnter a new string with space: ");
        scanf("%[^\n]s", name);
        printf(name);
        printf("\nEnter a new string without space: ");
        scanf("%s", name);
        printf(name);
        }
```

Exemplu *gets_s( )* si exp. regulare

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
using namespace std;
#include <regex>

const int max_dim = 30;

int main( ) {
        //C version
        printf("\nGets: ");
        char buf[max_dim];
        char* adr = gets_s(buf, sizeof(buf));
        //char* adr = gets_s(buf, _countof(buf));
        printf(buf);
        printf("\nAdr. init %p Adr. ret: %p", buf, adr);
        double d_var;
        printf("\nEnter a double value: ");
        scanf("%lf", &d_var);
        printf("\nThe double value is: %lf \n", d_var);

printf("\nGets new string: ");
cin.ignore( );
char *aadr = gets_s(buf, sizeof(buf));
//char* aadr = gets_s(buf, _countof(buf));
printf(buf);
printf("\nAdr. init %p Adr. ret: %p", buf, aadr);
```

```
//C++ version
        cout << "\n C++ - Introduceti numele si prenumele studentului: ";
        cin.getline(buf, sizeof(buf));
        //cin.getline(buf, _countof(buf));
        regex pattern(" ");
        while (!regex_search(buf, pattern))//verifica separare nume prenume cu spatiu
        {
                cout << "\nNu ati introdus corect, reintroduceti numele de forma: Nume
Prenume cu spatiu !!";
                cin.getline(buf, sizeof(buf));
                //cin.getline(buf, _countof(buf));
        }
        cout << "\nIntrodu o valoare double: ";
        cin >> d_var;
        cout << "\n Studentul este: " << buf << "\t Valoarea double e: " << d_var << endl;
        cout << "\nIntroduceti inca odata numele si prenumele studentului (nu se valideaza
spatiu intre nume si prenume): "<<endl;
        cin.ignore();
        cin.getline(buf, sizeof(buf));
        //cin.getline(buf, _countof(buf));
        cout << "\nIntrodu inca o valoare double: ";
        cin >> d_var;
        cout << "\n Studentul este: " << buf << "\t Valoarea double e: " << d_var << endl;
}
```

Example *wcin* and *wcout*:

```
//Siruri de caractere wide
#include <iostream>
using namespace std;
const int MAX = 255;

int main( ) {
   wchar_t wnume[MAX], wprenume[MAX];
   int an;
   wcout << L"\nIntroduceti numele- wide string: ";
   wcin >> wnume;
   wcout << L"\nIntroduceti prenumele- wide string: ";
   wcin >> wprenume;
   wcout << L"\nIntroduceti anul nasterii: ";
   cin >> an;
   wcout << endl << L"Datele persoanei: " << wnume << "
                " << wprenume << ", " << an <<endl;
   }//end main
```

3. Funcții pentru șiruri de caractere și afișari pe ecran

- Din biblioteca <string.h>/<cstring>
  - *strlen( )*
  - *strcat( )*
  - *strcmp( )*
  - *strdup( )*
  - *...*
- Pentru afișări pe ecran în mod text din <stdio.h>/<cstdio>
  - *textcolor( )*
  - *textbackgroundcolor( )*
  - *cprintf( )*
  - *cputs( )*

4. Funcții bazate pe caracter (ctype.h>/cctype)

- Caracteristici
  - utilizate pentru
    - testarea caracterelor
    - conversie a caracterelor
  - au ca si parametru caracterul de testat
  - returnează de obicei *True* sau *False*

Funcții de testare a caracterelor

- *isalpha( )* -> T dacă parametrul caracter este o literă alfabetică
- *isalnum( )* -> T dacă parametrul caracter este alfanumeric
- *islower( )* -> T dacă parametrul caracter este o literă mică
- *isupper( )* -> T dacă parametrul caracter este o literă mare
- *isdigit( )* -> T dacă parametrul caracter este o cifră între 0 și 9
- *iscntrl( )* -> T dacă parametrul caracter este o valoare ASCII între 0 și 31
- *isspace( )* -> T dacă parametrul caracter este spațiu ‘\n’, ‘\r’, ‘\t’, ‘\v’
- *isxdigit( )* -> T, dacă parametrul caracter este hexazecimal (0 to 9, A to F, sau a to f)

4. Funcții bazate pe caracter

- Funcții de conversie a caracterelor

  – Aceste funcții nu realizează efectiv conversia, doar returnează argumentul dat ca intrare, în noul format
  – *tolower( )* -> returnează argumentul convertit la literă mică
  – *toupper( )* -> returnează argumentul convertit la literă mare

  Formatul instructiunilor este:

int tolower(int** caracter**);
int toupper(int** caracter**);

Exemplu:

```
#include <ctype.h>

#include <stdio.h>
#include <string.h>

int main(void){
int loop;
char string[]="THIS IS A TEST";
for(loop=0;loop<strlen(string);loop++)
string[loop]=tolower(string[loop]);
//will convert character by character to lower case

printf("%s\n",string);
return 0;
}
```

```
//Character conversions lower<->upper no libraries
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

unsigned to_upper(unsigned, unsigned);

int main( ){
unsigned int mask_to_lower=0x20;
unsigned int mask_to_upper=0xDF;
char c;
printf("\nEnter a lower character: ");
fflush(stdin);
scanf(" %c",&c);
printf("\nThe equivalent upper character is = %c ", (char)(c & mask_to_upper));
printf("\nThe equivalent upper character with a function is = %c ",
(char)to_upper(c,mask_to_upper));
printf("\nEnter a upper character: ");
fflush(stdin);
scanf(" %c",&c);
printf("\nThe equivalent lower character is = %c ", (char)(c | mask_to_lower));
}

unsigned to_upper(unsigned a, unsigned mask){ return a & mask;}
```

5. Functii de alocare dinamica a memoriei

Sunt prezentate in precedentele cursuri

6. Functii pentru modul video

Nu sunt standardizate si implica utilizarea bibliotecii grafice cu fisierul header *graphics.h*.

Versiunea *C++ 2z* preconizeaza integrarea graficii 2D.

```
#include<graphics.h>

int main( )
{
   int gd = DETECT, gm;

   initgraph(&gd, &gm, "C:\\TC\\BGI");//BGI library
  …
    closegraph( );
   return 0;
}
```

7. Functii Mathematice (<*math.h*>/<*cmath*>)

Functii Trigonometrice

double acos(double** x**);

Returneaza arc cosin al lui *x* in radiani. Domeniu:

Valoarea lui *x* trebuie sa fie in domeniul -1 la +1 (inclusiv).
Valoarea de return e in domeniul 0 la pi (inclusiv).

double asin(double** x**);

Returneaza arc sin al lui *x* in radiani. Domeniu:

Valoarea lui *x* trebuie sa fie in domeniul -1 la +1 (inclusiv).
Valoarea de return e in domeniul -pi/2 la +pi/2 (inclusiv).

Similar pentru:

double atan(double** x**);

Returneaza arc tangenta lui *x* in radiani.

double atan2(double** y**, double** x**);

Returneaza arc tangenta lui *y/x* in radiani bazat pe semnele celor doua valori pentru a determina cadranul corect.

Domeniu:

Atat *y* cat si *x* nu pot fi zero. Valoarea de return e in domeniul lui -pi/2 la +pi/2 (inclusiv).

double cos(double** x**);

Returneaza cosinusul unghiului exprimat in radiani *x (x= ɵ*pi/180)*. Domeniu:

Valoarea lui *x nu are domeniu*. Valoarea de return e in domeniul -1 la +1 (inclusiv).

double cosh(double** x**);

Returneaza cosinus hyperbolic al lui *x*. Domeniu:

Nu este un domeniu pentru parametru si valoarea de return.

Functii similare pentru :

double sin(double** x**);
double sinh(double** x**);
double tan(double** x**);
double tanh(double** x**);

Functii
Exponentiale, Logaritmice si Putere

double exp(double** x**);

Returneaza valoarea lui *e* ridicata la puterea lui *x*.

double frexp(double** x**, int ***exponent**);

Numarul real *x este impartit* in *mantisa* si *exponent*.

Valoarea de return este *mantisa* si la pointerul intreg *exponent* vom avea exponentul.

Valoarea lui *x =mantisa * 2^exponent*.

Domeniu: e de forma *mantisa* in domeniul .5 (inclusiv) la 1 (exclusiv) (dependent de standardul folosit).

double log(double** x**);

Returneaza logaritmul natural (logaritm in baze-*e*) a lui *x*.

double log10(double** x**);

Returneaza logaritm in baza 10 a lui *x*.

double modf(double** x**, double ***integer**);

Imparte numarul double *x* in componentele intregi si fractionara.

Valoarea de return este partea fractionara (partea dupa punct), si seteaza pointerul *integer* la partea intreaga.

double pow(double** x**, double** y**);

Returneaza *x* ridicat la puterea lui *y*. Domeniu:

x nu poate fi negativ daca *y* e o valoare fractionara.

x nu poate fi zero daca *y* e mai mic sau egal cu zero.

double sqrt(double** x**);

Returneaza radical din *x*.

Domeniu: Argumentul nu poate fi negativ. Valoarea de return e totdeauna pozitiva. Exista supraincarcari pentru argumente *float* si la **unele variante si *int*.**

Alte Functii Matematice

double ceil(double** x**);

Returneaza valoarea primului intreg mai mic sau egal cu *x*.

double fabs(double** x**);

Returneaza valoarea absoluta a lui *x* (o valoare negativa devine pozitiva,o valoare pozitiva e nemodificata).

double floor(double** x**);

Returneaza valoarea primului intreg mai mare sau egal cu *x*.

double fmod(double x, double y);

Returneaza restul impartirii lui *x* la *y*. Domeniu:

Nu exista un domeniu pentru valoarea de return. Daca *y* e zero, atunci o eroare de domeniu va fi semnalizata, sau functia va returna zero (depinde de implementare).

8. Funcții pentru timp și dată (<time.h>/<ctime>)

- Header-ul de timp ofera mai multe functii utile pentru citirea si conversia timpului si datei curente. Unele functii sunt definite pentru un comportament local dat de setarea **LC_TIME**.
- Variabile secifice:
- **CLOCKS_PER_SEC** e numarul de *clock* pe secunda ale procesor-ului.
- **clock_t** e un tip utlizat la memorarea timpului procesor.
- **time_t** e un tip utlizat la memorarea timpului de tip calendar.

struct tm e o structura folosita a retine datele de tip *time* si *date* cu urmatorii membrii:

```
int     tm_sec;     /*     seconds after the minute (0 to 61) */
int     tm_min;     /*     minutes after the hour (0 to 59) */
int     tm_hour;    /*     hours since midnight (0 to 23) */
int     tm_mday;    /*     day of the month (1 to 31) */
int tm_mon;     /* months since January (0 to 11) */
int tm_year;    /* years since 1900 */
int tm_wday;    /* days since Sunday (0 to 6 Sunday=0)*/
int tm_yday;    /* days since January 1 (0 to 365) */
int tm_isdst;   /* Daylight Savings Time */
```

Daca *tm_isdst* e zero, atunci *Daylight Savings Time* nu are efect. Daca e o valoare pozitiva, atunci *Daylight Savings Time* are efect. Daca e negativ, atunci functiei utilizate i se cere sa calculeze sau nu *Daylight Savings Time* ca efect pentru timpul dat.

Nota ca *tm_sec* poate avea o valoare pana la 61 pentru a permite pana la doua secunde bisecte.

Cateva functii:

char *asctime(const struct tm *timeptr);

Returneaza un pointer la un sir de caratere care reprezinta ziua si timpul structurii *timeptr*. Sirul e in urmatorul format:

DDD MMM dd hh:mm:ss YYYY

DDD ziua din saptamana (Sun, Mon, Tue, Wed, Thu, Fri, Sat)

MMM Luna din an (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec)

dd Ziua din luna (1,...,31)

hh Ora (0,...,23) **mm** Minutul (0,...,59) **ss** Secunda (0,...,59) **YYYY** Anul

time_t time(time_t *timer);

Calculeaza timpul curent calendar si il codeaza in formatul *time_t*.

Valoarea *time_t* e returnata. Daca *timer* nu e un pointer null, atunci valoarea e de asemenea memorata in obiectul pointat. Daca timpul nu e accesibil se returneaza -1.

Exemple:

```
#include<time.h>
#include<stdio.h>


int main( )
{
        time_t timer;

        timer = time(NULL);
        printf("The current time is: %s\n",
asctime(localtime(&timer)));
        }
```

```
#include <time.h>
#include <stdio.h>

struct tm newtime;
__time32_t aclock;

int main(void) {
        char buffer[32];
        errno_t errNum;
        _time32(&aclock);   // Get time in seconds.
        _localtime32_s(&newtime, &aclock);   // Convert time to struct tm form.

         // Print local time as a string.

        errNum = asctime_s(buffer, 32, &newtime);//new variant C++
        if (errNum) {
                printf("Error code: %d", (int)errNum);
                return 1;
        }
        printf("Current date and time: %s", buffer);
        return 0;
}
```

clock_t clock(void);

Returneaza timpul procesorului folosit de la inceputul implementarii (in mod normal de la inceputul programului). Valoarea de return impartita la **CLOCKS_PER_SEC** ne va da numarul de secunde. Daca valoarea nu e disponibila va returna -1.

double difftime(time_t** time1**, time_t** time2**);

Calculeaza diferenta de secunde intre *time1* si *time2* (time1-time2). Returneaza numarul de secunde.

Exemplu

```
#include <time.h>
#include <stdio.h>

int main(void){
clock_t ticks1, ticks2;

ticks1=clock( );
ticks2=ticks1;
while((ticks2/CLOCKS_PER_SEC-
ticks1/CLOCKS_PER_SEC)<1)

ticks2=clock( );
printf("Took %ld ticks to wait one second.\n",ticks2- ticks1);
printf("This value should be the same as CLOCKS_PER_SEC
which is %ld.\n",CLOCKS_PER_SEC);
}
```

9. Alte funcții de bibliotecă

- Sunt multe alte functii de biblioteca si compilatoarele noi C++ considera multe din functiile vechi de tip *deprecated* oferind alte functii cu semnatura diferita.
- Ca si functii prezentate in precedentele cursuri si laboratoare avem: *qsort( ), rand( ), srand( ), atoi( ), strtod( ), va_start( ),…*etc.
- Ca si functii nespecificate avem alte functii:- pentru stabilirea domeniului valorilor numerice, detectie de erori, testare si debugging, timp local, interactiuni cu SO, etc.