ATOM-N 2022
AUGUST 25-28,
CONSTANTA,
ROMANIA
ADVANCED TOPICS IN OPTOELECTRONICS, MICROELECTRONICS AND NANOTECHNOLOGIES

SPIE.
CONNECTING MINDS.
ADVANCING LIGHT.

Class template – Primary Approach for the Flexible Dynamic Memory Allocation in the Hybrid Modelling Software Development

Emil M. Oanță[a], Alexandru Pescaru[b]

[a] **Constanța Maritime University, General Sciences Department**

[b] **Constanța Maritime University, Department of Navigation**

Acknowledgement:
This work is inspired by the results of the ID1223 - "Computer Aided Advanced Studies in Applied Elasticity from an Interdisciplinary Perspective" Scientific Research Project, under the supervision of the National University Research Council (CNCSIS), Romania, 2007-2010.

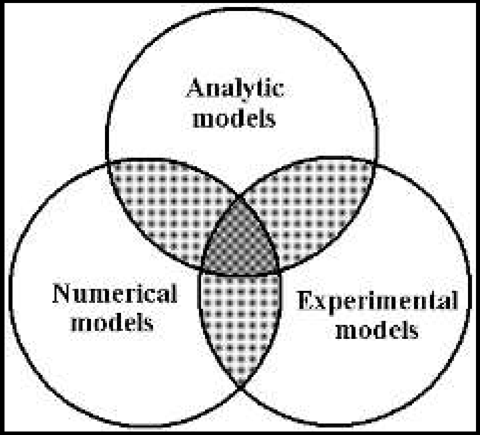
Table of contents

- Introduction, example of a hybrid model
- Motivation
- Methodology
  - Level 1 – data structures for mathematical methods
  - Level 2 – optimization: access time minimization
  - Level 3 – library of header files; rapid development
- Results
  - Template classes’ oriented development
  - Why not a general solution?
  - Some technical info and, of course, more questions
  - Type identification using a ‘common use’ compiler
  - T ‘generic’ type conversion
  - Operating the sample application
  - Some C++ Standard Template Library solutions
- Conclusion
- Acknowledgement
- References

Introduction

Actual conditions

- Complex phenomena;
- Huge volume of data;
- Multi-disciplinary approaches;
- Inter-domains influences;



New modelling paradigm → hybrid approaches

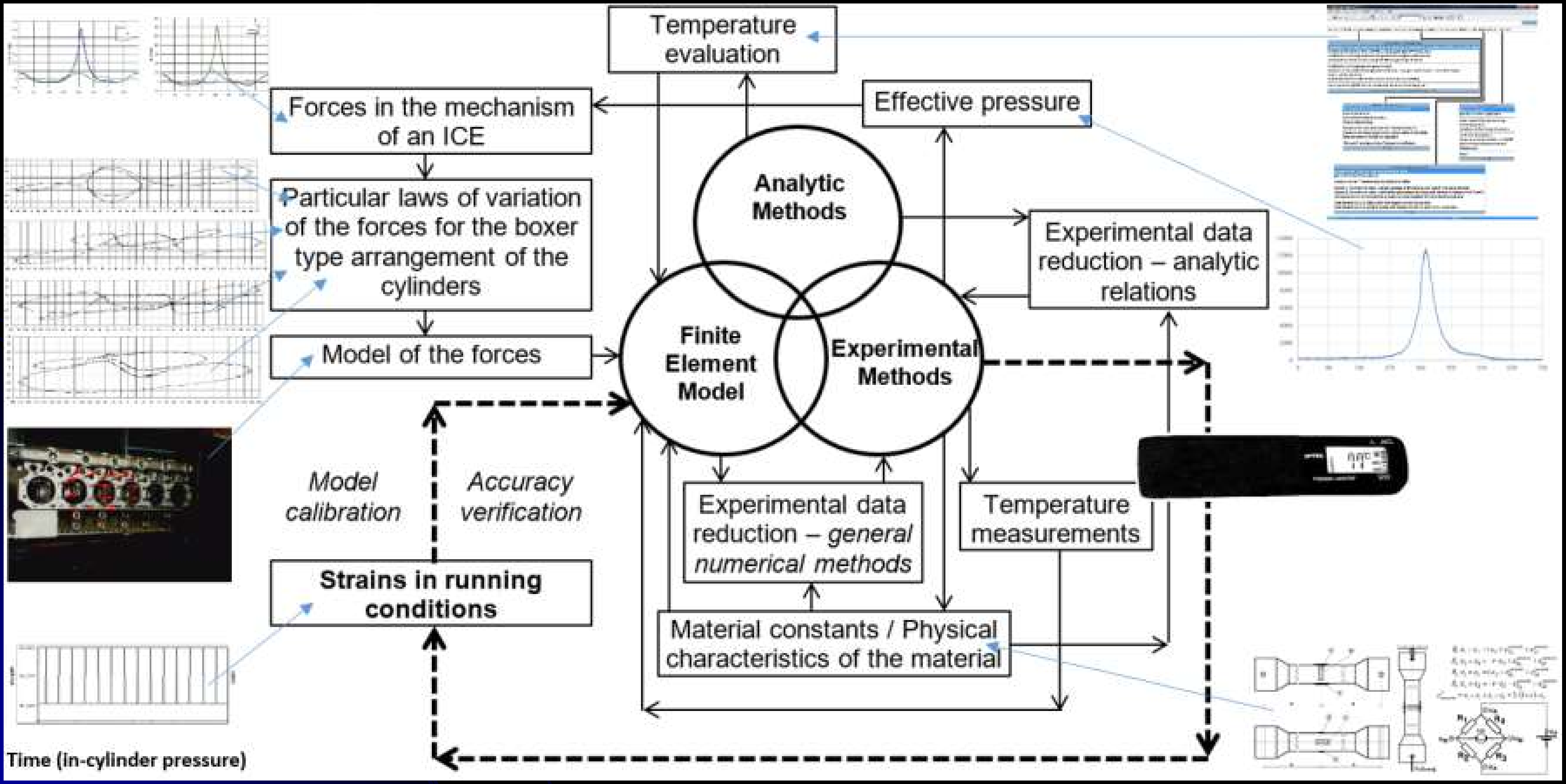
- Hybrid mode = ‘building blocks’ deeply connected;
- ‘Building block’ = study: theoretical model or experimental study;
- Theoretical approaches: analytical, numerical, semi-numerical etc.;
- ***Paramount: connections between the ‘building blocks’***;
- All the ‘building blocks’ are implemented as ***computer based solutions***;
- Connections = interfaces, i.e. ***computer based solutions***;

<<< Table of contents

Example – Hybrid model

An ICE's Structural model of the cylinder block:

- 12B165 – A navy vessel ICE;
- Running conditions;
- Experimental study → SAE2000, Detroit, USA;
- PhD, 'Cum laude', 88 letters / 105 reviewers;



Temperature evaluation

Forces in the mechanism of an ICE

Effective pressure

Particular laws of variation of the forces for the boxer type arrangement of the cylinders

Analytic Methods

Experimental data reduction – analytic relations

Model of the forces

Finite Element Model

Experimental Methods

Model calibration

Accuracy verification

Experimental data reduction – *general numerical methods*

Temperature measurements

Strains in running conditions

Material constants / Physical characteristics of the material

<<< Table of contents

Motivation

Key factor → *computer based solutions*!

Computers are used for distinct studies:

- Analytical approaches → original software;
- General numerical methods → original & commercial software;
- Dedicated numerical methods (FEM) → original & commercial software;
- Semi-numerical methods → original & commercial software;
- Dedicated algorithms & solvers → original & commercial software;
- Decisional problems → original software.

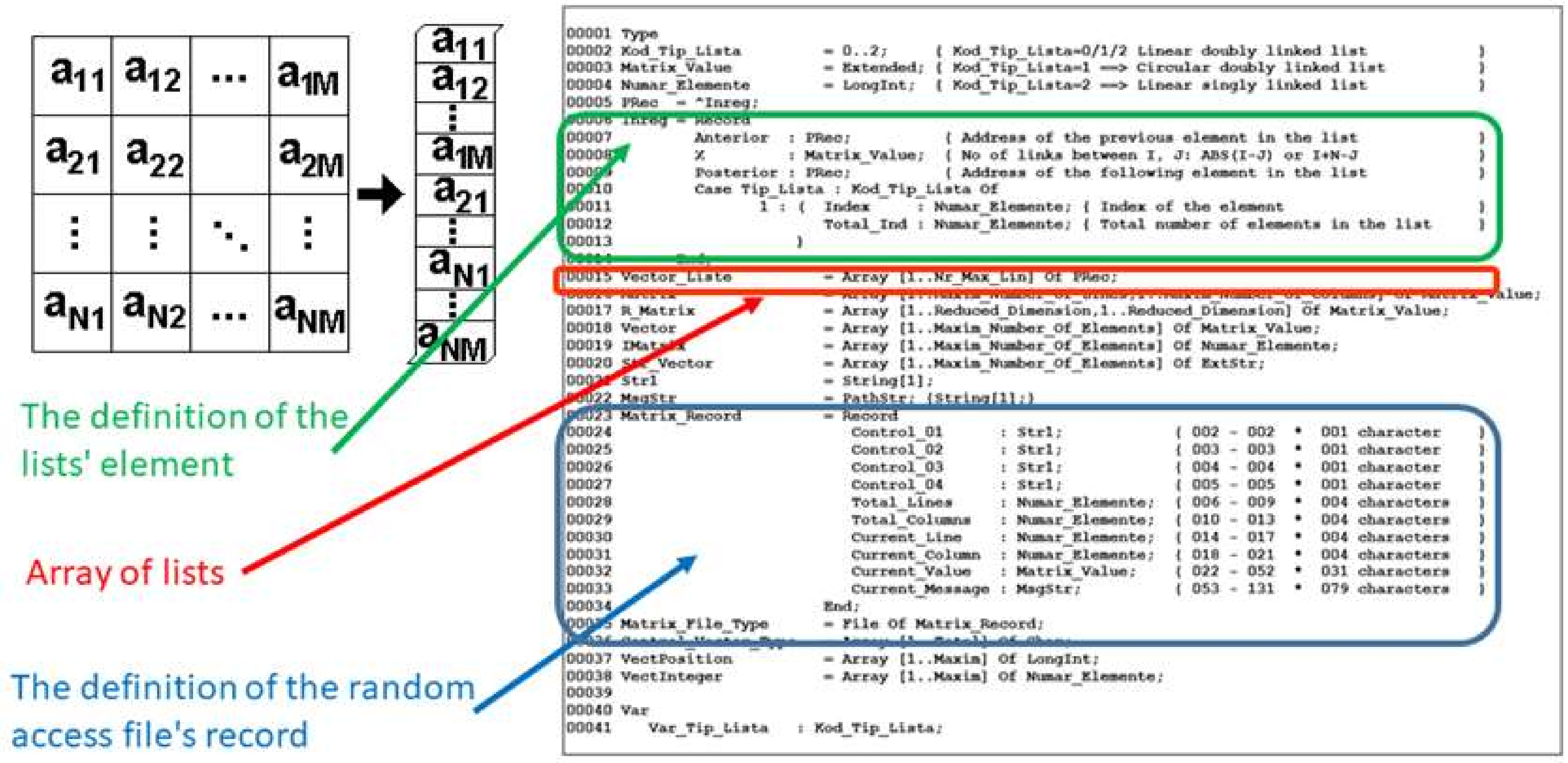
Computers are used to connect the studies:

- Based on the Application Program Interfaces → original software;
- Interfaces (CSV, JSON, original) → original software;
- Data integration → original software;
- Knowledge acquisition (using knowledge based systems) → original software;

<<< **Table of contents**

Methodology - Stages

Level 1 – data structures for mathematical methods



a_{11} a_{12} ... a_{1M}
a_{21} a_{22} a_{2M}
⋮ ⋮ ⋱ ⋮
a_{N1} a_{N2} ... a_{NM}
a_{11}
a_{12}
a_{1M}
a_{21}
a_{N1}
a_{NM}
The definition of the lists' element
Array of lists
The definition of the random access file's record

```
Type
Kod_Tip_Lista          = 0..2;      { Kod_Tip_Lista=0/1/2 Linear doubly linked list      }
Matrix_Value           = Extended;  { Kod_Tip_Lista=1 ==> Circular doubly linked list    }
Numar_Elemente         = LongInt;   { Kod_Tip_Lista=2 ==> Linear singly linked list      }
PRec  = ^Inreg;
Inreg = Record
         Anterior  : PRec;          { Address of the previous element in the list        }
         X         : Matrix_Value;  { No of links between I, J: ABS(I-J) or I+N-J       }
         Posterior : PRec;          { Address of the following element in the list       }
         Case Tip_Lista : Kod_Tip_Lista Of
               1 : (  Index     : Numar_Elemente; { Index of the element                 }
                      Total_Ind : Numar_Elemente; { Total number of elements in the list }
                   )
        End;
Vector_Liste           = Array [1..Nr_Max_Lin] Of PRec;
Matrix                 = Array [1..Maxim_Number_Of_Lines,1..Maxim_Number_Of_Columns] Of Matrix_Value;
R_Matrix               = Array [1..Reduced_Dimension,1..Reduced_Dimension] Of Matrix_Value;
Vector                 = Array [1..Maxim_Number_Of_Elements] Of Matrix_Value;
IMatrix                = Array [1..Maxim_Number_Of_Elements] Of Numar_Elemente;
Str_Vector             = Array [1..Maxim_Number_Of_Elements] Of ExtStr;
Str1                   = String[1];
MsgStr                 = PathStr; {String[1];}
Matrix_Record          = Record
                          Control_01      : Str1;            { 002 - 002  *  001 character  }
                          Control_02      : Str1;            { 003 - 003  *  001 character  }
                          Control_03      : Str1;            { 004 - 004  *  001 character  }
                          Control_04      : Str1;            { 005 - 005  *  001 character  }
                          Total_Lines     : Numar_Elemente;  { 006 - 009  *  004 characters }
                          Total_Columns   : Numar_Elemente;  { 010 - 013  *  004 characters }
                          Current_Line    : Numar_Elemente;  { 014 - 017  *  004 characters }
                          Current_Column  : Numar_Elemente;  { 018 - 021  *  004 characters }
                          Current_Value   : Matrix_Value;    { 022 - 052  *  031 characters }
                          Current_Message : MsgStr;          { 053 - 131  *  079 characters }
                         End;
Matrix_File_Type       = File Of Matrix_Record;
Control_Vector_Type    = Array [1..Total] Of Char;
VectPosition           = Array [1..Maxim] Of LongInt;
VectInteger            = Array [1..Maxim] Of Numar_Elemente;

Var
    Var_Tip_Lista    : Kod_Tip_Lista;
```

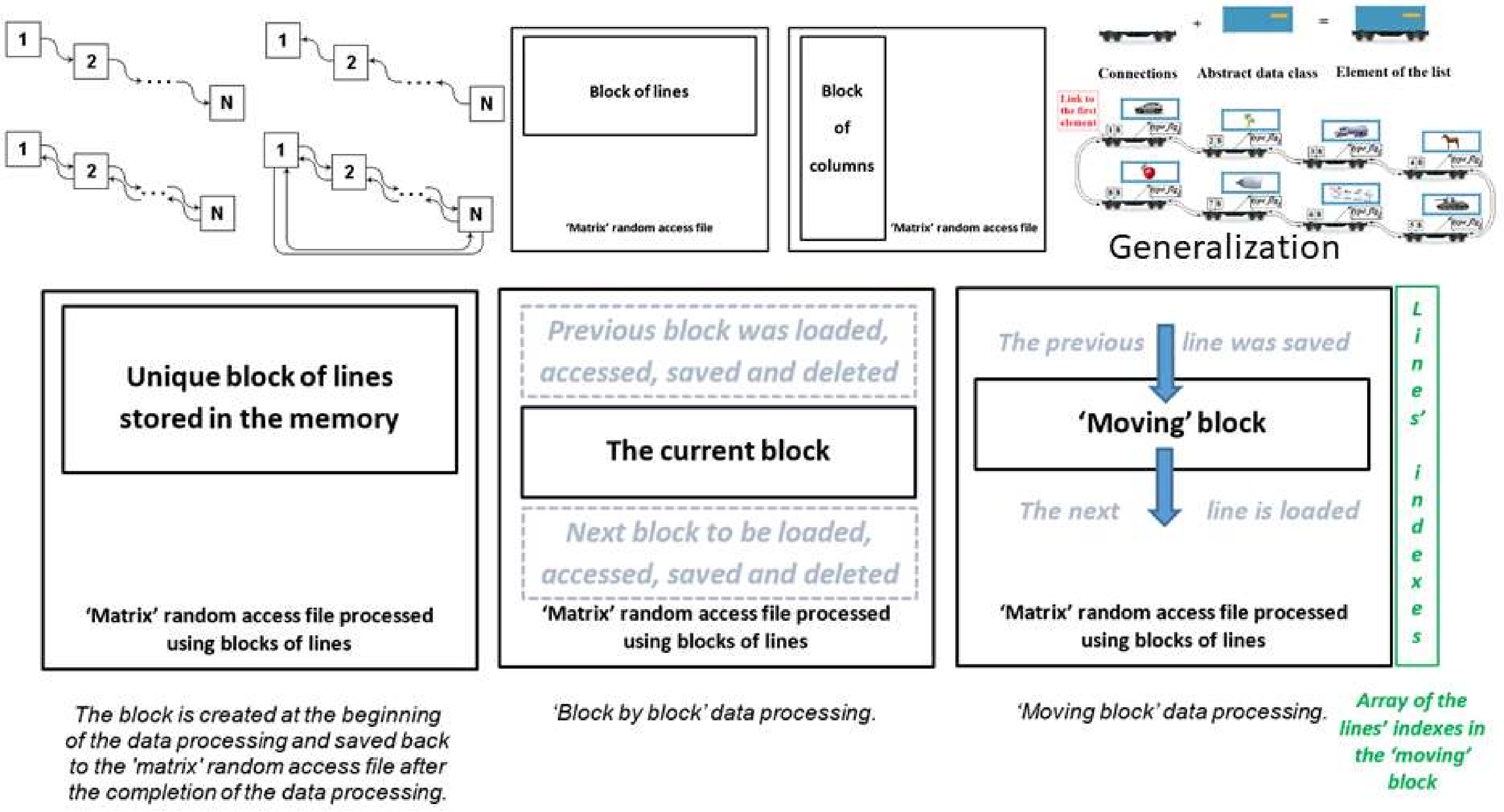
- Matrix methods are ubiquitous in science;
- Large matrices → finer discretization → higher accuracy;

<<< Table of contents

Level 2 – optimizations →

Methodology - Stages

Level 2 – optimization: access time minimization



The block is created at the beginning of the data processing and saved back to the 'matrix' random access file after the completion of the data processing.

'Block by block' data processing.

'Moving block' data processing.

Array of the lines' indexes in the 'moving' block

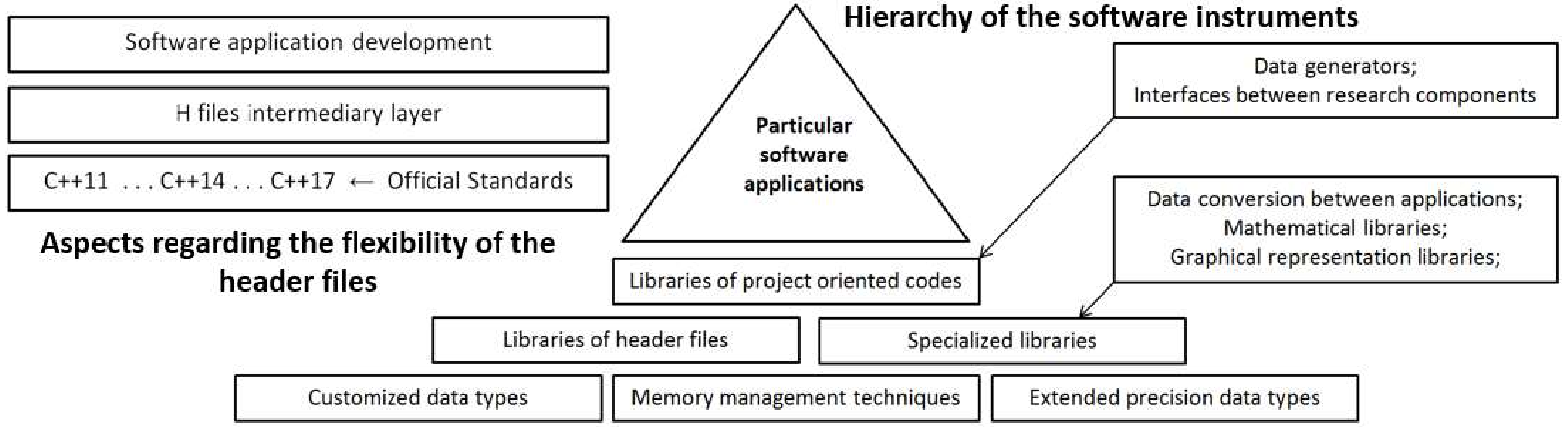
- Data structures for dynamical memory allocation;
- Vector of doubly linked lists for matrices' block processing;

<<< Table of contents

Level 3 – library of header files; rapid development →

Methodology - Stages

Level 3 – library of header files; rapid development


Hierarchy of the software instruments

Software application development

H files intermediary layer

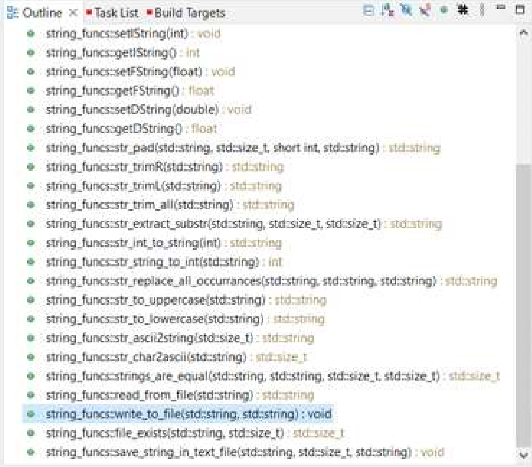
C++11 . . . C++14 . . . C++17 ← Official Standards

Aspects regarding the flexibility of the header files

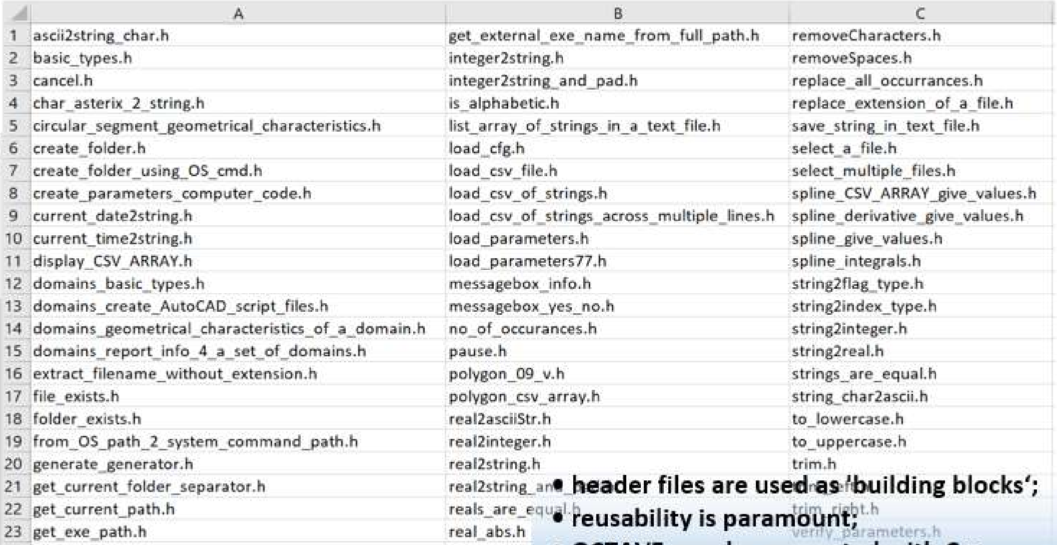
Particular software applications
Data generators;
Interfaces between research components
Data conversion between applications;
Mathematical libraries;
Graphical representation libraries;
Libraries of project oriented codes
Libraries of header files
Specialized libraries
Customized data types
Memory management techniques
Extended precision data types

New: class oriented programming

string_funcs class



```
string_funcs::setIString(int) : void
string_funcs::getIString() : int
string_funcs::setFString(float) : void
string_funcs::getFString() : float
string_funcs::setDString(double) : void
string_funcs::getDString() : float
string_funcs::str_pad(std::string, std::size_t, short int, std::string) : std::string
string_funcs::str_trimR(std::string) : std::string
string_funcs::str_trimL(std::string) : std::string
string_funcs::str_trim_all(std::string) : std::string
string_funcs::str_extract_substr(std::string, std::size_t, std::size_t) : std::string
string_funcs::str_int_to_string(int) : std::string
string_funcs::str_string_to_int(std::string) : int
string_funcs::str_replace_all_occurrances(std::string, std::string, std::string) : std::string
string_funcs::str_to_uppercase(std::string) : std::string
string_funcs::str_to_lowercase(std::string) : std::string
string_funcs::str_ascii2string(std::size_t) : std::string
string_funcs::str_char2ascii(std::string) : std::size_t
string_funcs::strings_are_equal(std::string, std::string, std::size_t, std::size_t) : std::size_t
string_funcs::read_from_file(std::string) : std::string
string_funcs::write_to_file(std::string, std::string) : void
string_funcs::file_exists(std::string, std::size_t) : std::size_t
string_funcs::save_string_in_text_file(std::string, std::size_t, std::string) : void
```



	A	B	C
1	ascii2string_char.h	get_external_exe_name_from_full_path.h	removeCharacters.h
2	basic_types.h	integer2string.h	removeSpaces.h
3	cancel.h	integer2string_and_pad.h	replace_all_occurrances.h
4	char_asterix_2_string.h	is_alphabetic.h	replace_extension_of_a_file.h
5	circular_segment_geometrical_characteristics.h	list_array_of_strings_in_a_text_file.h	save_string_in_text_file.h
6	create_folder.h	load_cfg.h	select_a_file.h
7	create_folder_using_OS_cmd.h	load_csv_file.h	select_multiple_files.h
8	create_parameters_computer_code.h	load_csv_of_strings.h	spline_CSV_ARRAY_give_values.h
9	current_date2string.h	load_csv_of_strings_across_multiple_lines.h	spline_derivative_give_values.h
10	current_time2string.h	load_parameters.h	spline_give_values.h
11	display_CSV_ARRAY.h	load_parameters77.h	spline_integrals.h
12	domains_basic_types.h	messagebox_info.h	string2flag_type.h
13	domains_create_AutoCAD_script_files.h	messagebox_yes_no.h	string2index_type.h
14	domains_geometrical_characteristics_of_a_domain.h	no_of_occurances.h	string2integer.h
15	domains_report_info_4_a_set_of_domains.h	pause.h	string2real.h
16	extract_filename_without_extension.h	polygon_09_v.h	strings_are_equal.h
17	file_exists.h	polygon_csv_array.h	string_char2ascii.h
18	folder_exists.h	real2asciiStr.h	to_lowercase.h
19	from_OS_path_2_system_command_path.h	real2integer.h	to_uppercase.h
20	generate_generator.h	real2string.h	trim.h
21	get_current_folder_separator.h	real2string_an	[illegible]
22	get_current_path.h	reals_are_equal.h	trim_right.h
23	get_exe_path.h	real_abs.h	verify_parameters.h

Library of header files in C++

- header files are used as 'building blocks';
- reusability is paramount;
- OCTAVE may be connected with C++;
- API of the CAD/CAE commercial software;

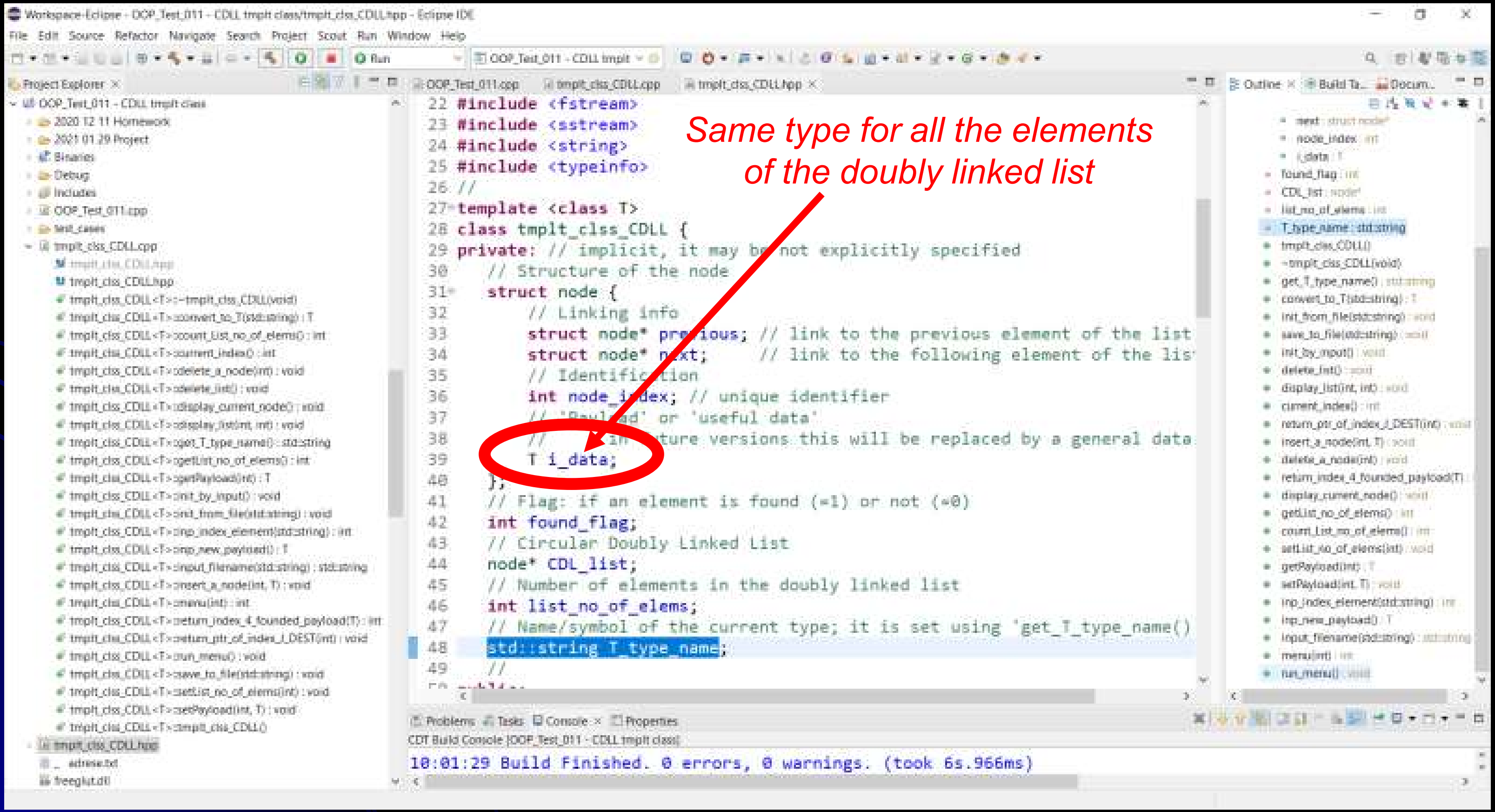
<<< Table of contents

According to these trends, new developments →

Results

Template classes’ oriented development

- Classes: functions in *old* header files are included in *dedicated* classes;
- Template classes: types are ‘generic’ ← high reusability;


Workspace-Eclipse - OOP_Test_011 - CDLL tmplt class/tmplt_clss_CDLL.hpp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Scout Run Window Help
Run
OOP_Test_011 - CDLL tmplt
Project Explorer
OOP_Test_011.cpp
tmplt_clss_CDLL.cpp
tmplt_clss_CDLL.hpp
Outline
Build Ta...
Docum...
Problems
Tasks
Console
Properties

```
OOP_Test_011 - CDLL tmplt class
  2020 12 11 Homework
  2021 01 29 Project
  Binaries
  Debug
  Includes
  OOP_Test_011.cpp
  test_cases
  tmplt_clss_CDLL.cpp
    tmplt_clss_CDLL.hpp
    tmplt_clss_CDLL.hpp
    tmplt_clss_CDLL<T>::~tmplt_clss_CDLL(void)
    tmplt_clss_CDLL<T>::convert_to_T(std::string) : T
    tmplt_clss_CDLL<T>::count_List_no_of_elems() : int
    tmplt_clss_CDLL<T>::current_index() : int
    tmplt_clss_CDLL<T>::delete_a_node(int) : void
    tmplt_clss_CDLL<T>::delete_list() : void
    tmplt_clss_CDLL<T>::display_current_node() : void
    tmplt_clss_CDLL<T>::display_list(int, int) : void
    tmplt_clss_CDLL<T>::get_T_type_name() : std::string
    tmplt_clss_CDLL<T>::getList_no_of_elems() : int
    tmplt_clss_CDLL<T>::getPayload(int) : T
    tmplt_clss_CDLL<T>::init_by_input() : void
    tmplt_clss_CDLL<T>::init_from_file(std::string) : void
    tmplt_clss_CDLL<T>::inp_index_element(std::string) : int
    tmplt_clss_CDLL<T>::inp_new_payload() : T
    tmplt_clss_CDLL<T>::input_filename(std::string) : std::string
    tmplt_clss_CDLL<T>::insert_a_node(int, T) : void
    tmplt_clss_CDLL<T>::menu(int) : int
    tmplt_clss_CDLL<T>::return_index_4_founded_payload(T) : int
    tmplt_clss_CDLL<T>::return_ptr_of_index_I_DEST(int) : void
    tmplt_clss_CDLL<T>::run_menu() : void
    tmplt_clss_CDLL<T>::save_to_file(std::string) : void
    tmplt_clss_CDLL<T>::setList_no_of_elems(int) : void
    tmplt_clss_CDLL<T>::setPayload(int, T) : void
    tmplt_clss_CDLL<T>::tmplt_clss_CDLL()
  tmplt_clss_CDLL.hpp
  adrese.txt
  freeglut.dll
```

```cpp
#include <fstream>
#include <sstream>
#include <string>
#include <typeinfo>
//
template <class T>
class tmplt_clss_CDLL {
private: // implicit, it may be not explicitly specified
    // Structure of the node
    struct node {
        // Linking info
        struct node* previous; // link to the previous element of the list
        struct node* next;     // link to the following element of the lis
        // Identification
        int node_index; // unique identifier
        // 'Payload' or 'useful data'
        //       in future versions this will be replaced by a general data
        T i_data;
    };
    // Flag: if an element is found (=1) or not (=0)
    int found_flag;
    // Circular Doubly Linked List
    node* CDL_list;
    // Number of elements in the doubly linked list
    int list_no_of_elems;
    // Name/symbol of the current type; it is set using 'get_T_type_name()
    std::string T_type_name;
    //
public:
```

```
next : struct node*
node_index : int
i_data : T
found_flag : int
CDL_list : node*
list_no_of_elems : int
T_type_name : std::string
tmplt_clss_CDLL()
~tmplt_clss_CDLL(void)
get_T_type_name() : std::string
convert_to_T(std::string) : T
init_from_file(std::string) : void
save_to_file(std::string) : void
init_by_input() : void
delete_list() : void
display_list(int, int) : void
current_index() : int
return_ptr_of_index_I_DEST(int) : void
insert_a_node(int, T) : void
delete_a_node(int) : void
return_index_4_founded_payload(T) : int
display_current_node() : void
getList_no_of_elems() : int
count_List_no_of_elems() : int
setList_no_of_elems(int) : void
getPayload(int) : T
setPayload(int, T) : void
inp_index_element(std::string) : int
inp_new_payload() : T
input_filename(std::string) : std::string
menu(int) : int
run_menu() : void
```

```
CDT Build Console [OOP_Test_011 - CDLL tmplt class]
10:01:29 Build Finished. 0 errors, 0 warnings. (took 6s.966ms)
```

Same type for all the elements of the doubly linked list

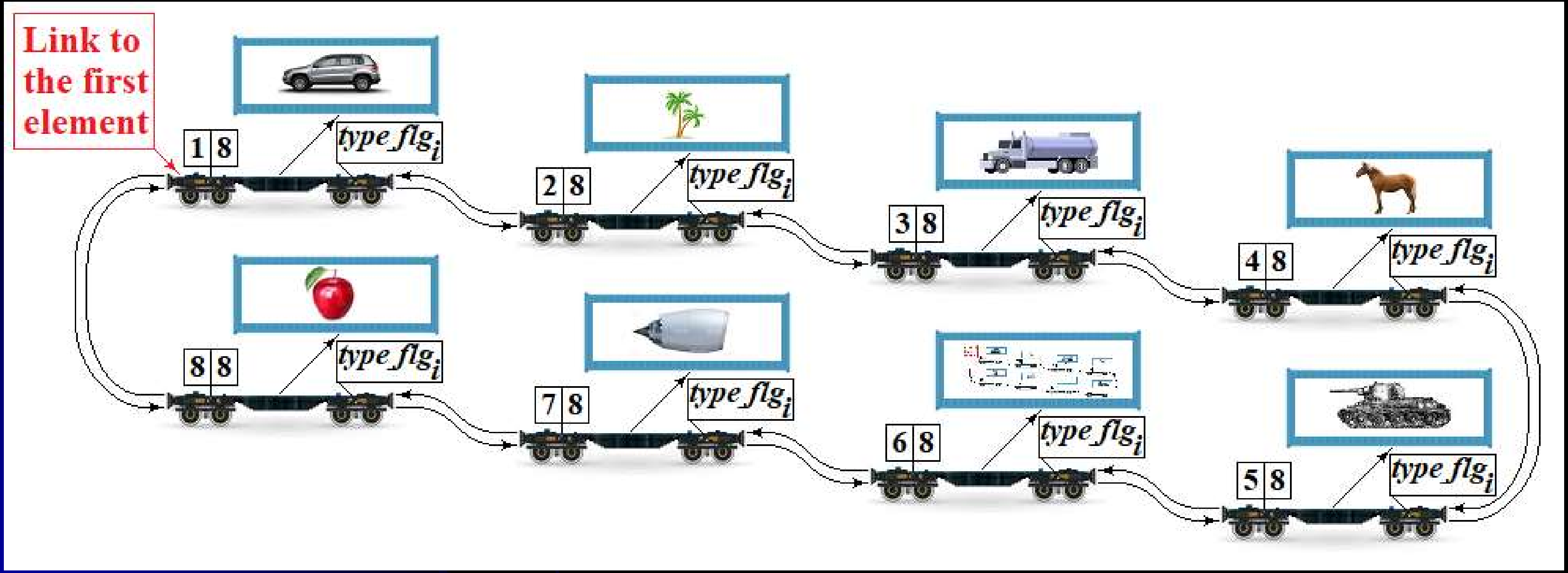
<<< Table of contents

Eclipse environment

Results

Why not a general solution?

- General solution: each element may have a distinct type;
- Template solution is not so general ← however why is it preferable?;
- SWOT analysis?
- Example 1: coordinates ← geometry, design, FEM, calculus domain;
- Example 2: identifiers ← graph theory, element definition (FEM);

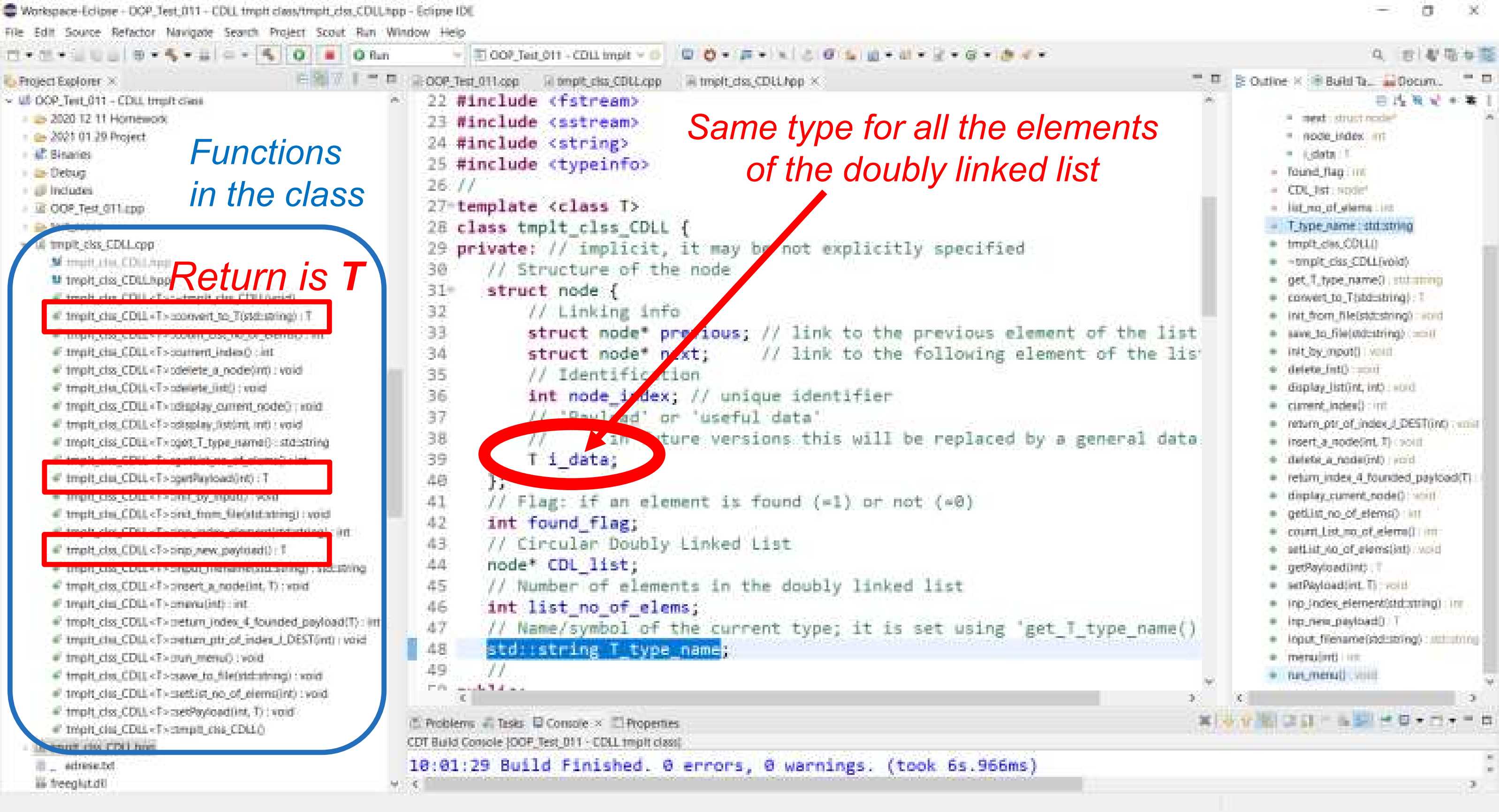


<<< Table of contents

Results

Some technical info and, of course, more questions

- Functions in header files and fields of the classes;
- Input/output variables of T type ← how to handle 'generic' types;


Workspace-Eclipse - OOP_Test_011 - CDLL tmplt class/tmplt_clss_CDLL.hpp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Scout Run Window Help
Run
OOP_Test_011 - CDLL tmplt
Project Explorer
OOP_Test_011.cpp
tmplt_clss_CDLL.cpp
tmplt_clss_CDLL.hpp
Outline
Build Ta...
Docum...
Problems
Tasks
Console
Properties

Functions in the class

Return is *T*

```
OOP_Test_011 - CDLL tmplt class
    2020 12 11 Homework
    2021 01 29 Project
    Binaries
    Debug
    Includes
    OOP_Test_011.cpp
    tmplt_clss_CDLL.cpp
        tmplt_clss_CDLL.hpp
        tmplt_clss_CDLL.hpp
        tmplt_clss_CDLL<T>::~tmplt_clss_CDLL(void)
        tmplt_clss_CDLL<T>::convert_to_T(std::string) : T
        tmplt_clss_CDLL<T>::count_list_no_of_elems() : int
        tmplt_clss_CDLL<T>::current_index() : int
        tmplt_clss_CDLL<T>::delete_a_node(int) : void
        tmplt_clss_CDLL<T>::delete_list() : void
        tmplt_clss_CDLL<T>::display_current_node() : void
        tmplt_clss_CDLL<T>::display_list(int, int) : void
        tmplt_clss_CDLL<T>::get_T_type_name() : std::string
        tmplt_clss_CDLL<T>::getList_no_of_elems() : int
        tmplt_clss_CDLL<T>::getPayload(int) : T
        tmplt_clss_CDLL<T>::init_by_input() : void
        tmplt_clss_CDLL<T>::init_from_file(std::string) : void
        tmplt_clss_CDLL<T>::inp_index_element(std::string) : int
        tmplt_clss_CDLL<T>::inp_new_payload() : T
        tmplt_clss_CDLL<T>::input_filename(std::string) : std::string
        tmplt_clss_CDLL<T>::insert_a_node(int, T) : void
        tmplt_clss_CDLL<T>::menu(int) : int
        tmplt_clss_CDLL<T>::return_index_4_founded_payload(T) : int
        tmplt_clss_CDLL<T>::return_ptr_of_index_i_DEST(int) : void
        tmplt_clss_CDLL<T>::run_menu() : void
        tmplt_clss_CDLL<T>::save_to_file(std::string) : void
        tmplt_clss_CDLL<T>::setList_no_of_elems(int) : void
        tmplt_clss_CDLL<T>::setPayload(int, T) : void
        tmplt_clss_CDLL<T>::tmplt_clss_CDLL()
    tmplt_clss_CDLL.hpp
    adresa.txt
    freeglut.dll
```

Same type for all the elements of the doubly linked list

```cpp
#include <fstream>
#include <sstream>
#include <string>
#include <typeinfo>
//
template <class T>
class tmplt_clss_CDLL {
private: // implicit, it may be not explicitly specified
    // Structure of the node
    struct node {
        // Linking info
        struct node* previous; // link to the previous element of the list
        struct node* next;     // link to the following element of the lis
        // Identification
        int node_index; // unique identifier
        // 'Payload' or 'useful data'
        // in future versions this will be replaced by a general data
        T i_data;
    };
    // Flag: if an element is found (=1) or not (=0)
    int found_flag;
    // Circular Doubly Linked List
    node* CDL_list;
    // Number of elements in the doubly linked list
    int list_no_of_elems;
    // Name/symbol of the current type; it is set using 'get_T_type_name()
    std::string T_type_name;
    //
public:
```

```
        next : struct node*
        node_index : int
        i_data : T
    found_flag : int
    CDL_list : node*
    list_no_of_elems : int
    T_type_name : std::string
    tmplt_clss_CDLL()
    ~tmplt_clss_CDLL(void)
    get_T_type_name() : std::string
    convert_to_T(std::string) : T
    init_from_file(std::string) : void
    save_to_file(std::string) : void
    init_by_input() : void
    delete_list() : void
    display_list(int, int) : void
    current_index() : int
    return_ptr_of_index_i_DEST(int) : void
    insert_a_node(int, T) : void
    delete_a_node(int) : void
    return_index_4_founded_payload(T) : int
    display_current_node() : void
    getList_no_of_elems() : int
    count_List_no_of_elems() : int
    setList_no_of_elems(int) : void
    getPayload(int) : T
    setPayload(int, T) : void
    inp_index_element(std::string) : int
    inp_new_payload() : T
    input_filename(std::string) : std::string
    menu(int) : int
    run_menu() : void
```

```
CDT Build Console [OOP_Test_011 - CDLL tmplt class]
10:01:29 Build Finished. 0 errors, 0 warnings. (took 6s.966ms)
```

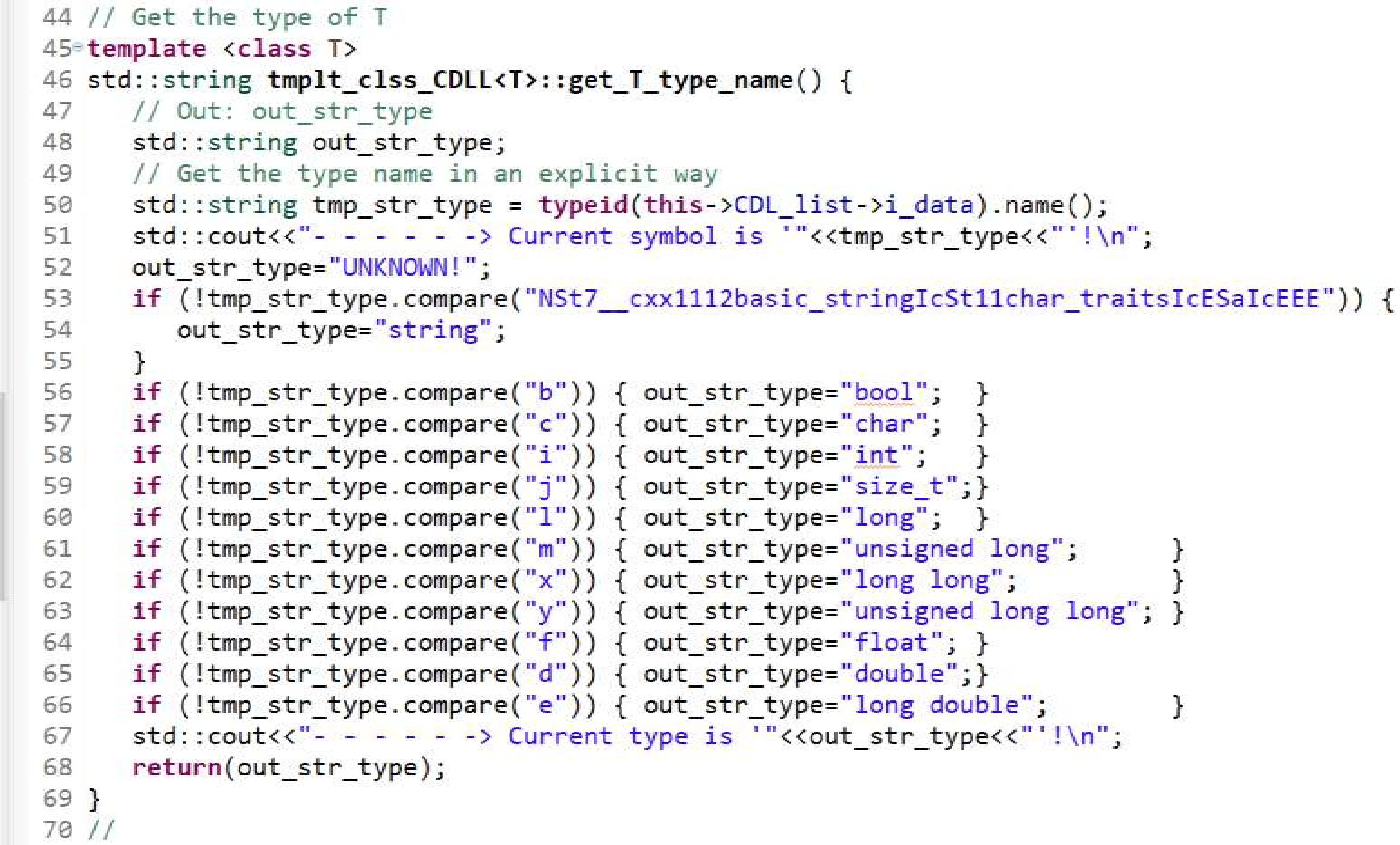
<<< Table of contents

T type identification →

Results

Type identification using a 'common use' compiler

- Function (programming 'trick') used to identify a 'generic' type . . .



```
// Get the type of T
template <class T>
std::string tmplt_clss_CDLL<T>::get_T_type_name() {
    // Out: out_str_type
    std::string out_str_type;
    // Get the type name in an explicit way
    std::string tmp_str_type = typeid(this->CDL_list->i_data).name();
    std::cout<<"- - - - - -> Current symbol is '"<<tmp_str_type<<"'!\n";
    out_str_type="UNKNOWN!";
    if (!tmp_str_type.compare("NSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEE")) {
        out_str_type="string";
    }
    if (!tmp_str_type.compare("b")) { out_str_type="bool";  }
    if (!tmp_str_type.compare("c")) { out_str_type="char";  }
    if (!tmp_str_type.compare("i")) { out_str_type="int";   }
    if (!tmp_str_type.compare("j")) { out_str_type="size_t";}
    if (!tmp_str_type.compare("l")) { out_str_type="long";  }
    if (!tmp_str_type.compare("m")) { out_str_type="unsigned long";      }
    if (!tmp_str_type.compare("x")) { out_str_type="long long";          }
    if (!tmp_str_type.compare("y")) { out_str_type="unsigned long long"; }
    if (!tmp_str_type.compare("f")) { out_str_type="float"; }
    if (!tmp_str_type.compare("d")) { out_str_type="double";}
    if (!tmp_str_type.compare("e")) { out_str_type="long double";        }
    std::cout<<"- - - - - -> Current type is '"<<out_str_type<<"'!\n";
    return(out_str_type);
}
//
```

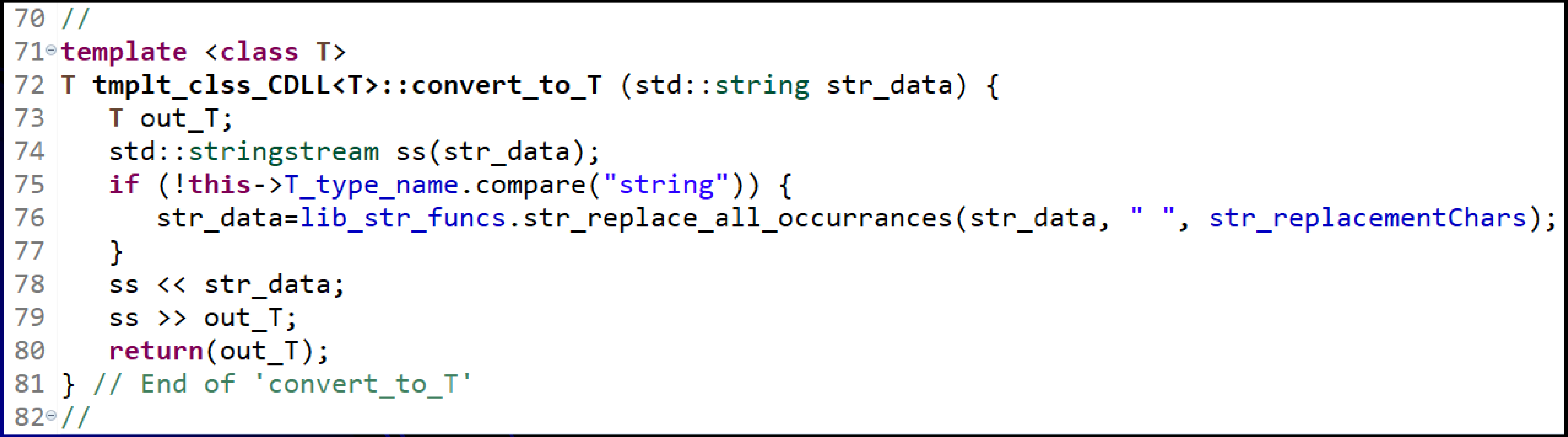
<<< Table of contents

T type conversion →

Results

T ‘generic’ type conversion

- Function (programming trick) used to convert a ‘generic’ type . . .
- Idea: convert *T* generic type to string & from string;
- Handling strings is paramount!
- Strings connect the data to the input/output (?*CSV*?) text files;



```
//
template <class T>
T tmplt_clss_CDLL<T>::convert_to_T (std::string str_data) {
   T out_T;
   std::stringstream ss(str_data);
   if (!this->T_type_name.compare("string")) {
      str_data=lib_str_funcs.str_replace_all_occurrances(str_data, " ", str_replacementChars);
   }
   ss << str_data;
   ss >> out_T;
   return(out_T);
} // End of 'convert_to_T'
//
```

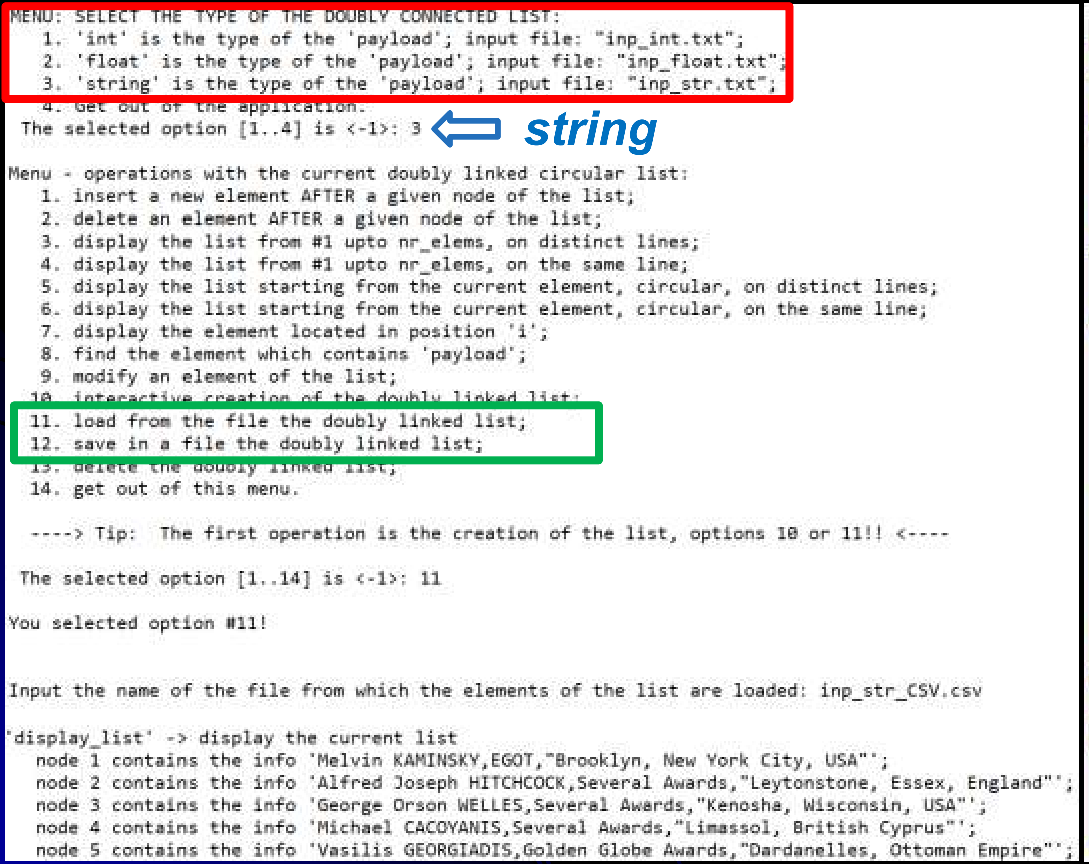
<<< Table of contents

Operating the sample application →

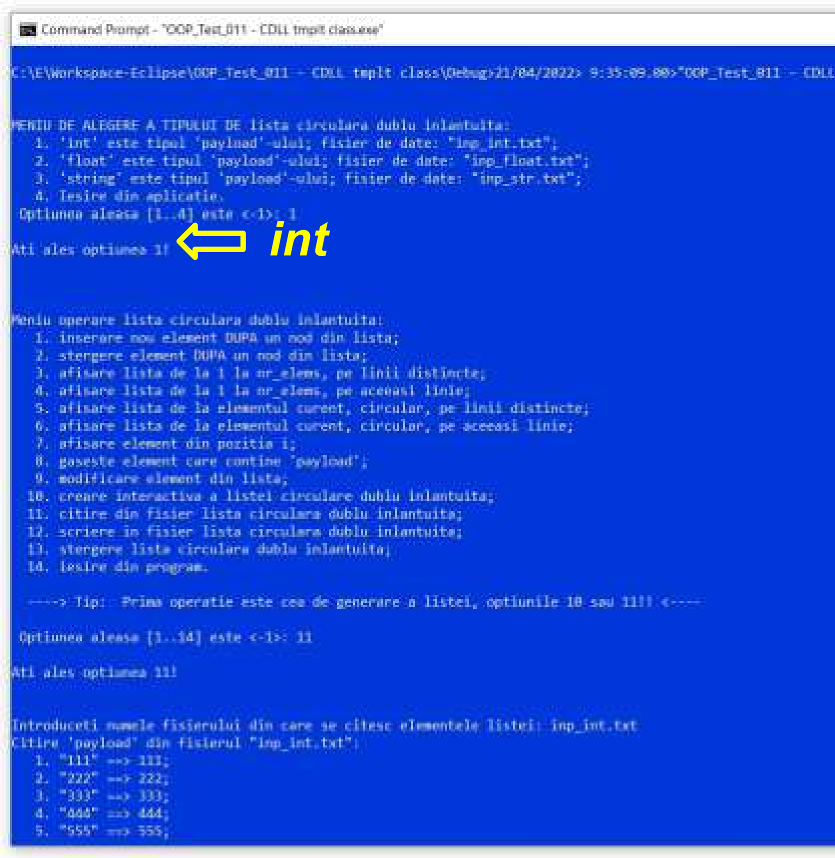
Results

Operating the sample application

- Operations → test the generation of various *T* types of doubly lnkd lists;
- Operations used to operate the doubly lnkd lists ← common in all apps;



```
MENU: SELECT THE TYPE OF THE DOUBLY CONNECTED LIST:
   1. 'int' is the type of the 'payload'; input file: "inp_int.txt";
   2. 'float' is the type of the 'payload'; input file: "inp_float.txt";
   3. 'string' is the type of the 'payload'; input file: "inp_str.txt";
   4. Get out of the application.
 The selected option [1..4] is <-1>: 3

Menu - operations with the current doubly linked circular list:
   1. insert a new element AFTER a given node of the list;
   2. delete an element AFTER a given node of the list;
   3. display the list from #1 upto nr_elems, on distinct lines;
   4. display the list from #1 upto nr_elems, on the same line;
   5. display the list starting from the current element, circular, on distinct lines;
   6. display the list starting from the current element, circular, on the same line;
   7. display the element located in position 'i';
   8. find the element which contains 'payload';
   9. modify an element of the list;
  10. interactive creation of the doubly linked list;
  11. load from the file the doubly linked list;
  12. save in a file the doubly linked list;
  13. delete the doubly linked list;
  14. get out of this menu.

  ----> Tip:  The first operation is the creation of the list, options 10 or 11!! <----

 The selected option [1..14] is <-1>: 11

You selected option #11!


Input the name of the file from which the elements of the list are loaded: inp_str_CSV.csv

'display_list' -> display the current list
   node 1 contains the info 'Melvin KAMINSKY,EGOT,"Brooklyn, New York City, USA"';
   node 2 contains the info 'Alfred Joseph HITCHCOCK,Several Awards,"Leytonstone, Essex, England"';
   node 3 contains the info 'George Orson WELLES,Several Awards,"Kenosha, Wisconsin, USA"';
   node 4 contains the info 'Michael CACOYANIS,Several Awards,"Limassol, British Cyprus"';
   node 5 contains the info 'Vasilis GEORGIADIS,Golden Globe Awards,"Dardanelles, Ottoman Empire"';
```



```
C:\E\Workspace-Eclipse\OOP_Test_011 - CDLL tmplt class\Debug>21/04/2022> 9:35:09.00>"OOP_Test_011 - CDLL

MENIU DE ALEGERE A TIPULUI DE lista circulara dublu inlantuita:
   1. 'int' este tipul 'payload'-ului; fisier de date: "inp_int.txt";
   2. 'float' este tipul 'payload'-ului; fisier de date: "inp_float.txt";
   3. 'string' este tipul 'payload'-ului; fisier de date: "inp_str.txt";
   4. Iesire din aplicatie.
 Optiunea aleasa [1..4] este <-1>: 1

Ati ales optiunea 1!


Meniu operare lista circulara dublu inlantuita:
   1. inserare nou element DUPA un nod din lista;
   2. stergere element DUPA un nod din lista;
   3. afisare lista de la 1 la nr_elems, pe linii distincte;
   4. afisare lista de la 1 la nr_elems, pe aceeasi linie;
   5. afisare lista de la elementul curent, circular, pe linii distincte;
   6. afisare lista de la elementul curent, circular, pe aceeasi linie;
   7. afisare element din pozitia i;
   8. gaseste element care contine 'payload';
   9. modificare element din lista;
  10. creare interactiva a listei circulare dublu inlantuita;
  11. citire din fisier lista circulara dublu inlantuita;
  12. scriere in fisier lista circulara dublu inlantuita;
  13. stergere lista circulara dublu inlantuita;
  14. Iesire din program.

  ----> Tip:  Prima operatie este cea de generare a listei, optiunile 10 sau 11!! <----

 Optiunea aleasa [1..14] este <-1>: 11

Ati ales optiunea 11!


Introduceti numele fisierului din care se citesc elementele listei: inp_int.txt
Citire 'payload' din fisierul "inp_int.txt":
   1. "111" ==> 111;
   2. "222" ==> 222;
   3. "333" ==> 333;
   4. "444" ==> 444;
   5. "555" ==> 555;
```

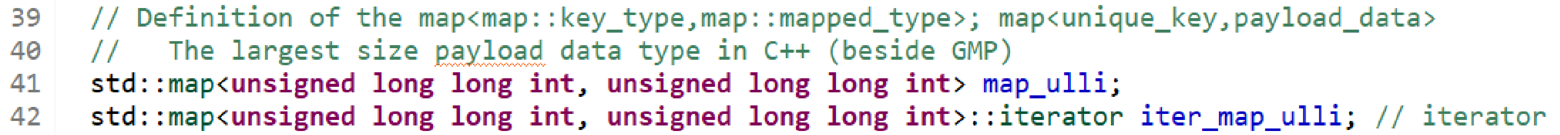
<<< Table of contents

Some *STL* solutions →

Results

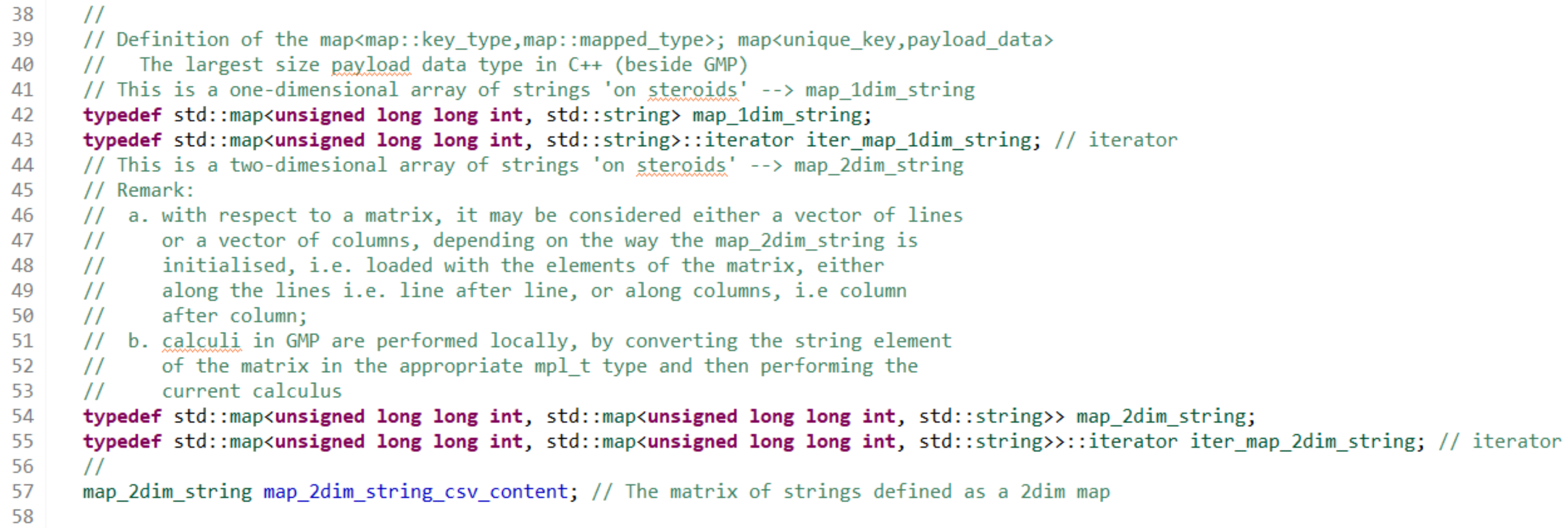
Some Standard Template Library C++ solutions

- Solution used to store random values, testing the degree of randomness;
- ‘map’ = associative container; ‘ulli’ stands for ‘unsigned long long int’;



```
// Definition of the map<map::key_type,map::mapped_type>; map<unique_key,payload_data>
//   The largest size payload data type in C++ (beside GMP)
std::map<unsigned long long int, unsigned long long int> map_ulli;
std::map<unsigned long long int, unsigned long long int>::iterator iter_map_ulli; // iterator
```

- Solution used to store strings which symbolize points’ coordinates;
- Useful to compute the coefficients within the polynomial regression;



```
//
// Definition of the map<map::key_type,map::mapped_type>; map<unique_key,payload_data>
//   The largest size payload data type in C++ (beside GMP)
// This is a one-dimensional array of strings 'on steroids' --> map_1dim_string
typedef std::map<unsigned long long int, std::string> map_1dim_string;
typedef std::map<unsigned long long int, std::string>::iterator iter_map_1dim_string; // iterator
// This is a two-dimesional array of strings 'on steroids' --> map_2dim_string
// Remark:
//  a. with respect to a matrix, it may be considered either a vector of lines
//     or a vector of columns, depending on the way the map_2dim_string is
//     initialised, i.e. loaded with the elements of the matrix, either
//     along the lines i.e. line after line, or along columns, i.e column
//     after column;
//  b. calculi in GMP are performed locally, by converting the string element
//     of the matrix in the appropriate mpl_t type and then performing the
//     current calculus
typedef std::map<unsigned long long int, std::map<unsigned long long int, std::string>> map_2dim_string;
typedef std::map<unsigned long long int, std::map<unsigned long long int, std::string>>::iterator iter_map_2dim_string; // iterator
//
map_2dim_string map_2dim_string_csv_content; // The matrix of strings defined as a 2dim map

```

<<< Table of contents

Conclusion →

Conclusion

Usefulness

- Modeling complex phenomena requires advanced concepts & instruments;
- Data integration and knowledge acquisition require computer based original software;
- Hybrid research approaches use composite models with deeply integrated modules;
- Facile and rapid development of software components is paramount;
- Various software libraries: solvers, interfaces, RNGs, data persistency etc.;
- “Failing to prepare, you are preparing to fail” → software development strategy;
- Strategy is valuable, being confirmed over the past 38 years in many R&D projects;

Progresses in the last 3 years

- Header files approach → class based programming, i.e. (atomization → integration);
- Class templates based programming = generic programming;
- Standard Template Library in C++: containers, iterators, algorithms, functions;
- Collections in Java (not presented in this paper);

Accomplishments

- Modern know how in the development of new software components for hybrid models
- Updated libraries and new libraries based on the aforementioned progresses;
- New computer based models in data science;

<<< Table of contents

Acknowledgement

The ideas presented in this scientific paper are inspired by the results of the ID1223 - "***Computer Aided Advanced Studies in Applied Elasticity from an Interdisciplinary Perspective***" Scientific Research Project, under the supervision of the National University Research Council (CNCSIS), Romania, 2007-2010[7] and by the follow-up scientific research project "***Mathematical Models for Inter-Domain Approaches with Applications in Engineering and Economy***", under the supervision of the National Authority for Scientific Research, Romania, 2010-2012[8].

<<< Table of contents

References

[1] Oanta, E., Nicolescu, B., “A Versatile PC-Based Method for the Processing of the Large Matrices”, Proc. DETC99/CIE ‘1999 ASME Design Engineering Technical Conference’, September 12-15, 1999, Las Vegas, Nevada, ISBN 9780791819722, Vol 2, paper 9059, pp 457-464, https://doi.org/10.1115/DETC99/CIE-9059 (1999).

[2] Oanta, E., “Hybrid modeling in mechanical engineering”, Habilitation thesis, September 21 2018, Doctoral School of Mechanical Engineering, Constanta Maritime University, https://cmu-edu.eu/blog/2018/09/18/sustinere-teza-de-abilitare-conf-univ-dr-ing-emil-m-oanta-21-09-2018/, Constanta (2018).

[3] Oanta, E., Pescaru, A., Lazaroiu, G., “General data structure for the dynamic memory allocation in the development of the computer based models in engineering”, Proc. SPIE 10977, Advanced Topics in Optoelectronics, Microelectronics, and Nanotechnologies IX, eISSN: 1996-756X, ISSN: 0277-786X, ISBN: 978-1-5106-2614-0; 1097721 (31 December 2018); doi: 10.1117/12.2324291; https://doi.org/10.1117/12.2324291 (2018).

[4] * * *, Standard Template Library, https://en.wikipedia.org/wiki/Standard_Template_Library (18 July 2022).

[5] Sabau, A., “Transient Regimes Analysis for a Diesel Engine”, Book Series: Advanced Materials Research, Volume 837, Page 471-476, DOI 10.4028/www.scientific.net/AMR.837.471, 2014, ISSN 1022-6680, ISBN 978-3-03785-929-2 (2014).

[6] Sabau, A., “Analysis of flow in fluidized bed of particles”, Book Series: IOP Conference Series-Materials Science and Engineering, Volume 916, DOI 10.1088/1757-899X/916/1/012097, 2020, ISSN 1022-6680, ISBN 978-3-03785-929-2 (2020).

[7] Oanță, E., & all, “Computer Aided Advanced Studies in Applied Elasticity from an Interdisciplinary Perspective”, Research Project ID1223, under the supervision of the National University Research Council (CNCSIS), Romania, 2007-2010.

[8] Oanta, E., Panait, C., Lepadatu, L., Tamas, R., Constantinescu, M. et. all., Mathematical Models for Inter-Domain Approaches with Applications in Engineering and Economy, MIEC2010 - Bilateral Romania-Moldavia Scientific Research Project, 2010-2012, under the supervision of the National Authority for Scientific Research, Romania.

<<< Table of contents

Questions?

“Failing to prepare, you are preparing to fail”   Benjamin Franklin

<<< Table of contents