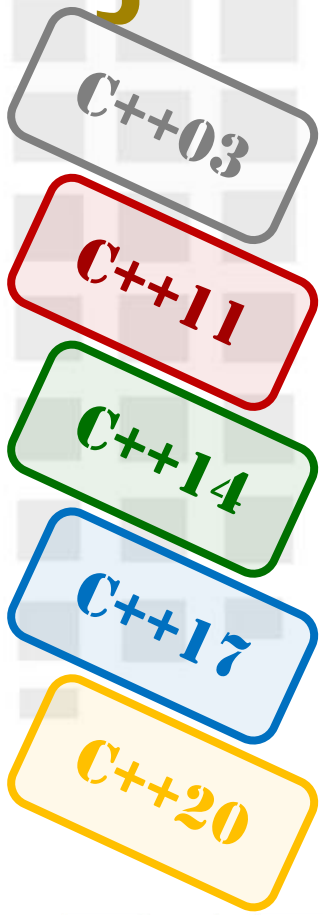
Institut Mines-Télécom
Didier GUERIOT (Dpt. ITI)
Johanne VINCENT (Dpt. INFO)
Reda BELLAFQIRA (Dpt. ITI)
Didier.Gueriot@imt-atlantique.fr
Johanne.Vincent@imt-atlantique.fr
Reda.Bellafqira@imt-atlantique.fr

Advanced C++ programming



- **Introduction to C++**
  - Slot 1
  - → C++: from C and beyond
  - → Classes, objects and lifetime (vs. JAVA)
  - → Oriented-Object Programming (inheritance, polymorphism)
- **Memory management & object manipulation**
  - → References, « copy » / « move » object construction
  - → Overloading operators
- **Template vs OO programming**
  - → Template functions and classes
- **The Standard Template Library**
  - → Containers, iterators and algorithms
  - → Using sequence & associative containers …
- **Smart pointers (STL & Boost)**

Resources

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom


« *Effective STL* » - Meyers – Addison Wesley


« *Effective C++* » - Meyers – Addison Wesley


« *More Effective C++* » - Meyers – Addison Wesley


« *C++ Coding Standards: 101 rules, guidelines and best practices* »
Herb Sutter – Addison Wesley


« *Modern C++ Design: Generic Programming and Design Patterns Applied* »
Andrei Alexandrescu – Addison Wesley


C++ bible 1 - **`http://www.cplusplus.com`**


C++ bible 2 - **`http://en.cppreference.com`**

Bookmark these !!!


Boost – `http://www.boost.org`

Why C++ ?

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- **Core libraries often implemented in C++**
  - ITK : « Insight Segmentation and Registration Toolkit »
    - Image processing (medical flavor)
  - OTB : « ORFEO Toolbox »
    - Algorithms : image processing, segmentation, classification, ...
    - Remote sensing main target (radar, satellite, ...)
  - QuantLib : Library for Quantitative Finance
    - Modeling, simulation, risk assessment, ...
- **Even as user, some concepts / tools are required**
  - « templates », « smart pointers », ...
  - Libraries : STL, Boost

Once upon a time ...

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- First OOP (Object Oriented Programing) language ⇨ *Simula*

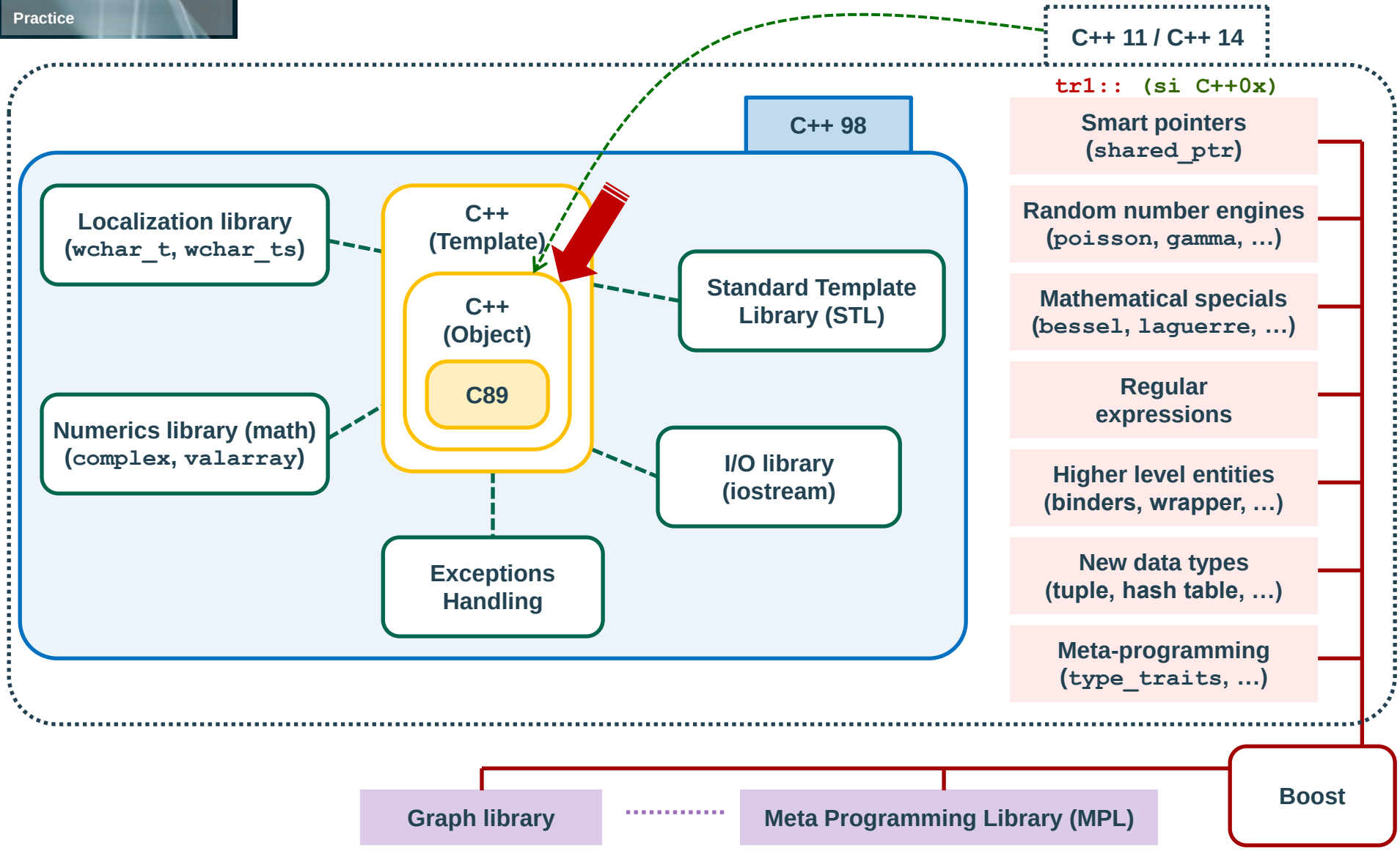
- Imperative procedural language supporting structured programing ⇨ *C language*
  - Denis Richie & Brian Kernighan (*Bell Labs*)
  - Compiled, low-level system access (memory, speed) ⇨ UNIX system
  - ANSI standard since 1989 (C ANSI)

- Early stages of *C++ language*
  - Bjarne Stroustrup (*Bell Labs* ...) ⇨ provide Simula classes to C
  - 1979 ⇨ "C with classes" (acting more like a C pre-compiler)

- *C++* genesis
  - "Named by" Rick Mascitti (coming from the ++ operator)
  - 1982 – 1985 - 1989 : start – first commercial release – normalization
  - "C++98" : first C++ standard only in 1998

Core C++ and beyond

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom



Back to C language C (1/4)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- Basic types & variables

```
[const]     [signed][unsigned]     [short][long]     int          nom_var = init_val;
            [signed][unsigned]     [short][long]     char
            [signed][unsigned]     [short][long]     float
            [signed][unsigned]     [short][long]     double
```



```
short int       i = -1;
int             j = 19, k = +123;
char            reponse = 'N';
unsigned char   byte;

byte = 255;
```

```
float       x = 3.0;
float       y = .5;
double      z = +1.3e-6;
```

- Functions

```
returned_type function_name ( parameters )
{

   local definitions ;

   statements ;

   return value_to_return ;

}
```



```
double carre( double x )
{

   double   retour;

   retour = x*x;

   return retour;

}
```

Back to C language C (2/4)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- Program startup ... where to start with ?
  - THE « **main** » function

```
int main( int argc, char * argv[] )
{
   /* Déclarations, instructions, appels
      de fonctions et méthodes */
   return 0;
}
```

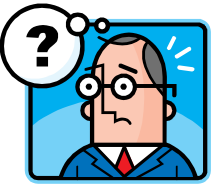
```
#include <iostream>

int main (int argc , char* argv[]) {
  std::cout << "Hello Laurent" << std::endl ;
  return 0 ;
}
```

Binary executable generation : "prog" ...
(How to make such a "prog" will be seen later)

Runtime :

```
> prog
Hello Laurent
```

- How can the program retrieve the data given (here "Jean-Marc") when called from the command line ?




```
> prog Jean-Marc
Hello Jean-Marc
```

Back to C language (3/4)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- C operators

```
+= -= *=
/= %=
```

Logical :
```
! && ||
```

Arithmetic (*binary too*) :
```
+ - * / %
```

Comparison :
```
== != < <= > >=
```

Unary :
```
++ -- & * sizeof()
```

- C++ operators
  - Dynamic storage : ~~`malloc()`~~ ~~`free()`~~ ⇨ `new delete`
  - Type conversion : ~~`(new_type) var ;`~~ ⇨ `static_cast<>()`
  - I/O & streams :

```
printf("%s : %d","value",12);  ⇨  std::cout << "value : " << 12;
```

Most of operators can be redefined: how ? … coming soon !

Back to C language (4/4)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- Flow control

```
if ( condition )
   statements_true
[else
   statements_false]
```

```
while ( condition )
   statements
```

```
for ( init_statement ; condition ; iter_expression )
   statements
```

```
double mediane( double * sTab, int nVal )
{
   double      med;

   if ( (nVal % 2) == 0 )
   {
      int   sup = nVal/2;
      med = ( sTab[sup-1] + sTab[sup] )/2;
   }
   else
      med = sTab[ static_cast<int>(nVal/2) ];

   return med;
}
```

```
double sommeP( double * sConv, int nVal, double eps )
{
   double   somme, sauve = 0;
   int      i;

   somme = sConv[ 1 ];
   i = 2;
   while ( (i<nVal) && (fabs(somme-sauve) > eps) ) {
      sauve = somme;
      somme += sConv[ i++ ];
   }

   return somme;
}
```

```
int sommeEntiers( int N )
{

   int   somme = 0, i;

   for ( i = 1; i <= N; i++ )
      somme += i;

   return somme;

}
```

Some comments ? (1/2)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- C or C++ style ?
  - Single or multi-line

```
/*
   Ceci est un commentaire
   sur plusieurs lignes
*/

// Ceci est un commentaire de ligne ...
int   var; // ... ou de fin de ligne
```

- External tools ("doxygen", ...)
  - Structured and richer documentation (up to you !)
  - Automatic HTML, LaTeX exports from the C++ code, ...

MaClasse.h

```
/*! \file fileName.h
 *  Documente le fichier complet
 *  \author Moi
 *  \date 20/10/2009
 */

/*! \class MaClasse
 *  Documente la classe suivante
 *  \brief Description brève
 */

class MaClasse
{
private :
   int   attribut;    //!< Documente l'attribut
public :
   MaClasse();        //!< Documente le constructeur
   ~MaClasse();       //!< Documente le destructeur
   //! Documente la méthode
   int uneMethode( int p );
};
```

MaClasse.cpp

```
#include "MaClasse.h"

//! Constructeur de la class MaClasse
MaClasse::MaClasse() {}

//! Destructeur de la class MaClasse
MaClasse::~MaClasse() {}

/*! Description de la méthode
 *  \param p    Un paramètre de type entier
 *  \return     Valeur de p + 1
 */
int MaClasse::uneMethode( int p )
{
   return p + 1;
}
```

Some comments ? (2/2)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

MaClasse.h

```
/*! \file fileName.h
 *  Documente le fichier complet
 *  \author Moi
 *  \date 20/10/2009
 */

/*! \class MaClasse
 *  Documente la classe suivante
 *  \brief Description brève
 */

class MaClasse
{
private :
   int   attribut;    //!< Documente l'attribut
public :
   MaClasse();        //!< Documente le constructeur
   ~MaClasse();       //!< Documente le destructeur
   //! Documente la méthode
   int uneMethode( int p );
};
```

doxygen …

doxywizard …

MaClasse.cpp

```
#include "MaClasse.h"

//! Constructeur de la class MaClasse
MaClasse::MaClasse() {}

//! Destructeur de la class MaClasse
MaClasse::~MaClasse() {}

/*! Description de la méthode
 *  \param p    Un paramètre de type entier
 *  \return     Valeur de p + 1
 */
int MaClasse::uneMethode( int p )
{
   return p + 1;
}
```

Page principale | Classes | Fichiers

Liste des classes | Membres de classe

Référence de la classe MaClasse

Description brève. Plus de détails...

#include <MaClasse.h>

Liste de tous les membres

Fonctions membres publiques

MaClasse ()
Documente le constructeur.

int **uneMethode** (int p)
Documente la méthode.

~MaClasse ()
Destructeur de la class **MaClasse**.

Attributs privés

int **attribut**
Documente l'attribut.

Description détaillée

Documente la classe suivante

Documentation des constructeurs et destructeur

MaClasse ( )

Constructeur de la class **MaClasse**.

~MaClasse ( )

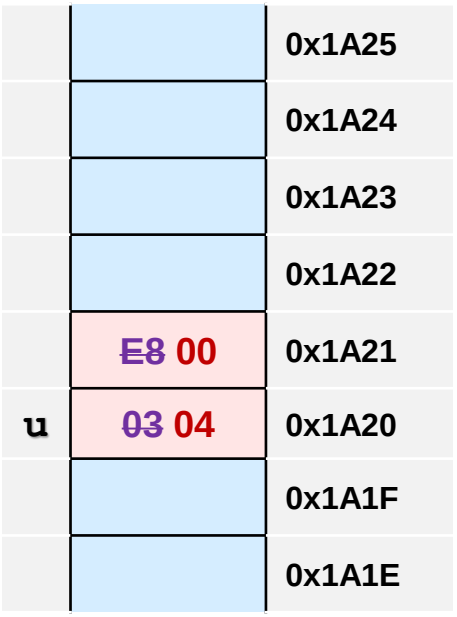
Variables (1/2)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- **What is a variable ?**
  - Specific area in computer memory, containing an entity having a type (integer, float, ...)
    - location : `&` operator (memory address)
    - size : `sizeof` operator (how many bytes ?)
  - Variable name = area ID (it gives the programmer an easy way to access this area)

```
// u is an integer type variable (2 bytes)
int u = 1000 ;       // 3*256+232 (03*FF+E8)

// Display variable "u" features
cout << u << endl ;            // contents
cout << sizeof(u) << endl ;    // size
cout << &u << endl ;           // location
cout << hex << &u << endl ;

// Write access to variable u
u = 1024 ;           // 4*256+0 (04*FF+0)
```

```
1000
2
6688
1A20
```



Variables (2/2)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

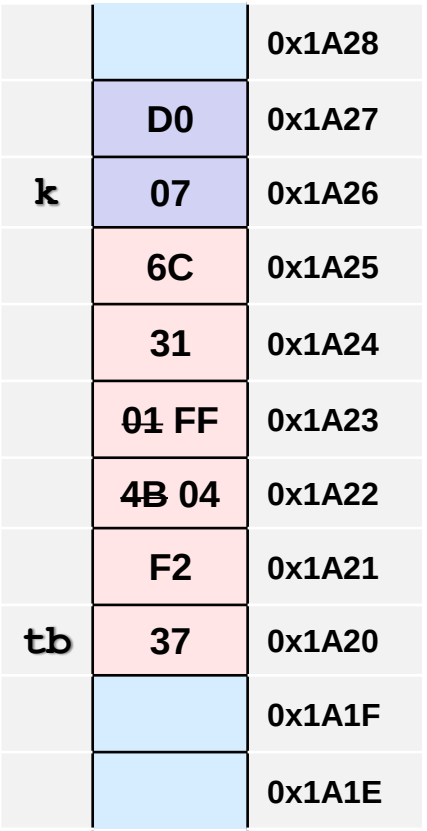
- **What is an "array" ?**
  - Specific area in machine memory containing several consecutive items having the same type

```
// tb is a 3-integer array (6 bytes in memory)
int tb[3] ;
```

  - How to access just one item ?
    - array name (= variable name)
    - offset (= item number starting from 0)

```
// Display array "tb" features
cout << hex << tb[0] << endl ;  // 1st item
cout << hex << tb[1] << endl ;  // 2nd item
cout << hex << tb[2] << endl ;  // 3rd item
cout << hex << tb[3] << endl ;  // 4th item ?????

// Write access to "tb" items
tb[1] = 1024 ;       // 4*256+0 (04*FF+0)
```

37F2
4B01
316C
07D0



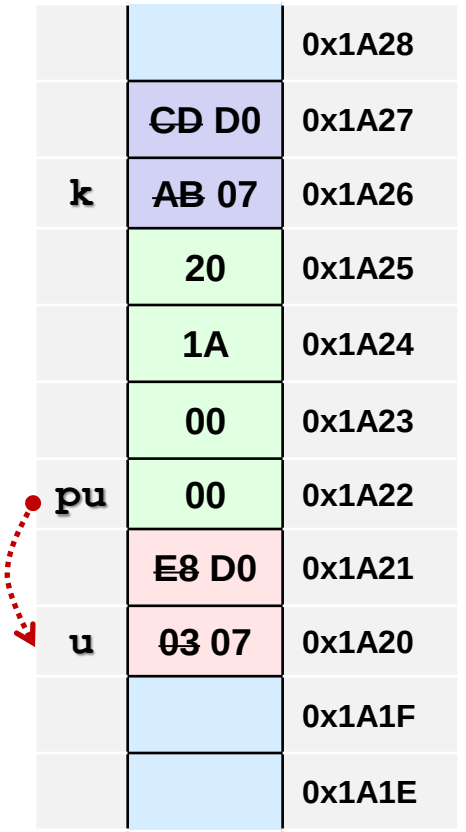
Pointers ? (1/3)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

```
// u et k are integer variables
//   (2 bytes each)
int u = 1000 ;       // 0x03E8
int k = 43981 ;      // 0xABCD
```

- **What is a pointer ?**
  - Type `T*` variable containing the address of :
    - one variable of type `T`
    - an area of several consecutive type `T` variables (~tableau)

```
// pu is a variable whose type is « pointer to integer » and
// initialized with the address of variable u ⇒ « pu points to u »
int* pu = &u ;
```



  - Access to the variable ? (dereferencing)

```
// Read access
k = *pu ;       // get the integer located at the address stored in
                // variable pu (here, this is 1000, the value of u)
                // and assign it to variable k

// Write access
*pu = 2000 ;    // write the value 2000 (0x07D0) in memory,
                // at the address stored in variable pu
                // (here, at location 0x1A20)
```

Pointers ? (2/3)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

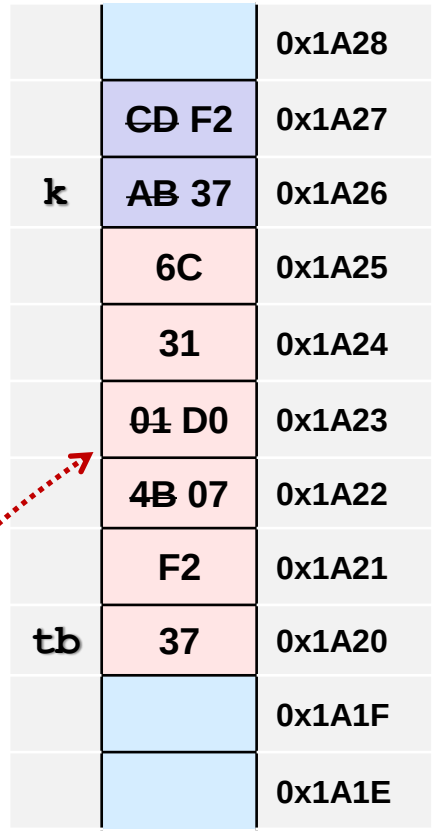
```
// k is an integer variable
//   (2 bytes)
int k = 43981 ;      // 0xABCD
```

- **What is a pointer ?**
  - Type `T*` variable containing the address of :
    - one variable of type `T`
    - an area of several consecutive type `T` variables (~tableau)

```
// tb is a 3-integer array
int tb[3] ;

// tb may be seen as a "pointer to an integer" (int*) initialized with
// the address of the 3-integer area
```

  - Access to the area ? (dereferencing)

```
// Read access
k = *(tb+0) ;  // get the array tb 1st item value and assign it to
               // variable k

// Write access
*(tb+1) = 2000 ;   // set the array tb 2nd item value to 2000 (0x07D0)
```

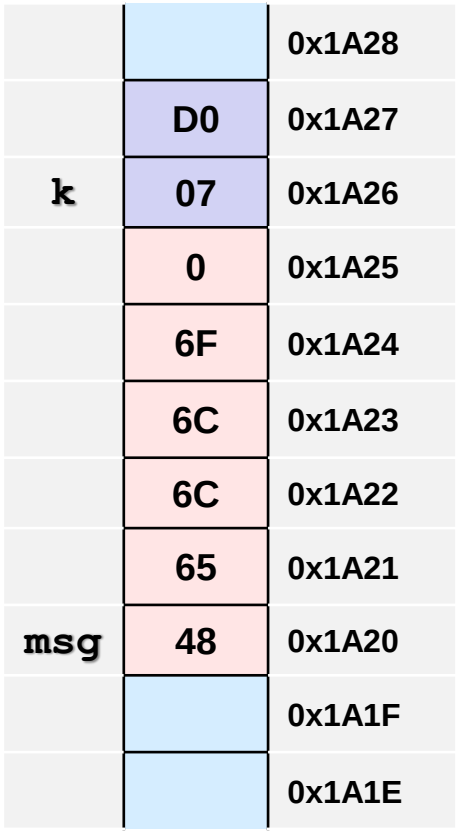


Pointers ? (3/3)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- C-style strings (also available in C++)
  - Characters array ended by character \0

```
// msg is a 6-character array (6 bytes)
char msg[6] = "Hello" ;
// OR
char msg[6] = {'H', 'e', 'l', 0x6C, 'o', '\0'} ;
// OR
char msg[] = "Hello" ;
// OR
char* msg = "Hello" ;
```



  - Code ASCII 0 character is used by all C string functions to know where the string ends (`strlen`, `strcpy`, ...)

- C++ strings
  - Classe `std::string`

Function call mechanism (1/2)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- Through the stack
  - Provide parameter values
    - passed by value (a copy)
    - lifetime (only the function scope)
  - Output the result


```
// Function f definition
double f (int v) {
  return v / 2.0 ;
}

int main (int argc , char* argv[]) {
  int m     = 10 ;
  double dm = 0.0 ;

  dm = f(m) ;    // Calling f

  return 0 ;
}
```

2 3 1 5 4

(1)	
dm	0.0
m	10

(2)	
v	10
	∅
dm	0.0
m	10

(3)	
v	10
	∅ 5.0
dm	0.0
m	10

(4)	
	5.0
dm	0.0
m	10

(5)	
dm	5.0
m	10

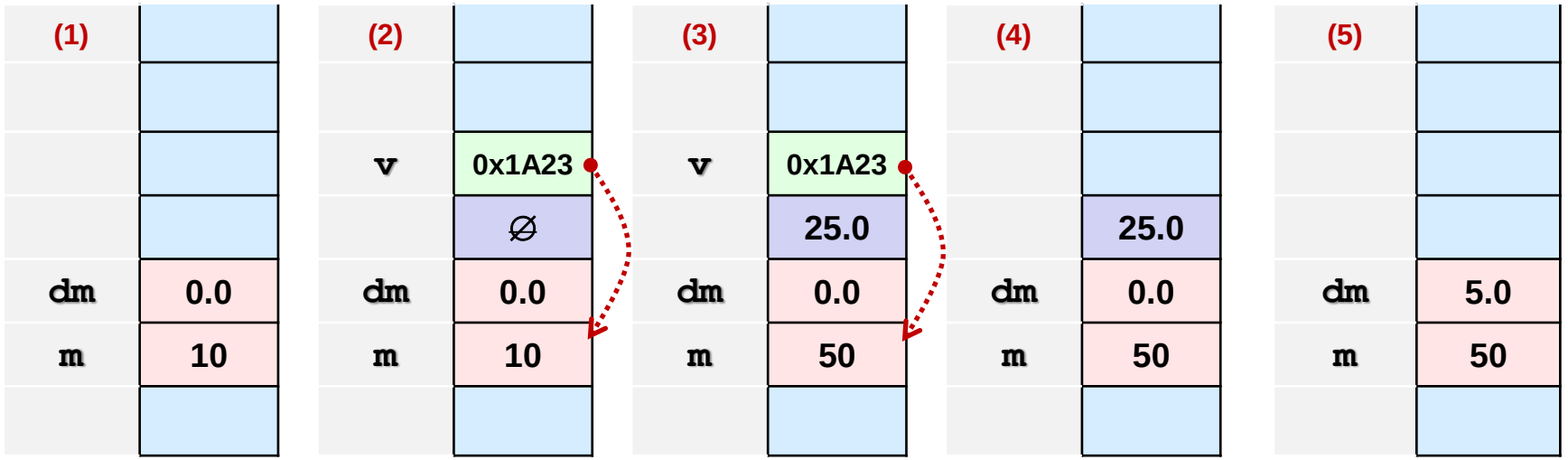
Function call mechanism (2/2)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- **Read/write parameters?**
  - Pass the address of the variable to be modified
    - The address is copied (by **value**)
    - Dereferencing the pointer allow accessing to the variable

2
3
1
5
4

```
// Function f definition
double f (int* v) {
   *v = 50 ;
   return (*v) / 2.0 ;
}

int main (int argc , char* argv[]) {
   int m      = 10 ;
   double dm = 0.0 ;

   dm = f(&m) ;    // Calling f

   return 0 ;
}
```



Build a binary executable(1/3)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- C++ files organization
  - Header files (« .h ») :
    - Declaration : types, constants, functions,
    - Files to be included when the declared entities are needed.
  - Implementation files (« .cpp ») :
    - Definition : global variables, functions contents,
    - Files to be compiled to produce intermediate binaries and finally the binary executable.
- Only one entry point for one program
  - The « main » function

```
int main (int argc , char* argv[]) {
  ...  // Program …
  return 0 ;
}
```

Build a binary executable (2/3)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- C++ preprocessor called prior to compiling
  - First step when building a binary
  - Provides directives but only at a textual level
- Some directives :
  - **`#define`** : macros or textual symbol definition
  - **`#include`** : allow files inclusion
  - ...

Build a binary executable (3/3)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

TP 1.3

main.cpp

```
#include <iostream>
#include <cmath>

int main (int argc, char* argv[])
{
  std::cout << "Hello World" << std:endl ;
  std::cout << "Résultat = " << std::cos(0) << std:endl ;

  return 0 ;
}
```

math.h

```
namespace std {
…
   double cos (double) ;
…
}
```

libm.so

01010110111
10100010010
01010000101

1

Compiling:

```
g++ -c main.cpp
```

main.o

0101011
1010001
0101000

2

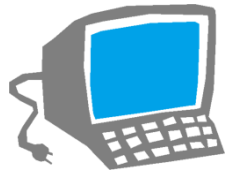
Linking:

```
g++ -o mon_prog main.o -lm
```

mon_prog

01010110111
10100010010
01010000101

```
-std=c++17
-Wall
```



Practice

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- **Building your first own program written in C++**
  - Compiling
  - Linking
  - Running
  - "Make" the process automatic ...

- **Bonus : using the command line !**
  - The command line arguments ?
  - Practical use of the command line ...