C++ Programming

Templates

M1 Math
Michail Lampis
Michail.lampis@dauphine.fr

Templates

- Sometimes we write a piece of code to handle a certain type of data (e.g. int)
- The exact same piece of code may still work for other types of data (e.g. double)
- Solution 1: we could copy our functions, change all int to double
- Solution 2: Templates

Find max

- Consider the following code:

```
int find_max(int *data, int size){
    int tmp = data[0];
    for(int i=1; i<size; i++)
    if( data[i]>tmp) tmp = data[i];
    return tmp;
}
```

- Change this to work with doubles...

Find max

- Consider the following code:

```
double find_max(double *data, int size){
    double tmp = data[0];
    for(int i=1; i<size; i++)
    if( data[i]>tmp) tmp = data[i];
    return tmp;
}
```

- The exact same thing works for any type that supports the > operation...

Templates

- Templates allow us to define functions with “Type variables”
  - The type of some parameters/variables is not pre-defined
  - It depends on how the function is called
- Syntax:

  template <typename T> …
  - The … is a function definition where we can use T as if it were a real type

Example

```
template <typename T> //Pretend T is a type!
T find_max(T* data, int size)
{
        T tmp = data[0];
        for(int i=0;i<size;i++)
                if(data[i]>tmp) tmp = data[i];
        return tmp;
}
```

Example

```
int a[ ] = { 2, 3 , 5 ,1 ,6 };
double b[ ] = { 2.1,3.2, 4.1, 5.7, 1.2};
cout << find_max(a,5) << endl;
cout << find_max(b,5) << endl;
```

- Two versions of find_max( ) are compiled
  - One for int, one for double...

Class Templates

- More generally we can make generic classes
- Example: Linked lists

```
struct Node {
    int data;
    struct Node * next;
};
```

- Why should this only work for int?

Class Templates

```
template <typename T>
struct Node {
        T data;
        struct Node *next;
};
```

- This defines a generic linked list node type

Using template classes

- Define an object of a template class

```
Node<int> mynode;
mynode.data = 5;
mynode.next = NULL;
```

Example

```
Node<int> *p = new Node<int>;
p->data = 2;
p->next = 0;

Node<int> *p2 = new Node<int>;
p2->data = 3;
p2->next = p;
```

The STL

- Thanks to templates, the C++ standard library contains many functions/data structures which can easily work with any type
- → Standard Template Library
- Example: vector class
  - vector<int> x;
  - vector<double> y;
- Too much material to cover in this class…

Binary trees

- For the last TD exercise you will have to program a class that looks a bit like a linked list
  - But, every node has two “next” nodes (called its children)
- Try to make this generic, using templates
  - The logic is always the same for any data type you use