UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI **MATEMATICA E INFORMATICA**

# Programmazione ad Oggetti

Carmine Dodaro

Anno Accademico 2019/2020

# Templates

## A cosa servono?

- Permettono di creare diverse funzioni o classi che condividono funzionalità
- Permettono di effettuare operazioni identiche su diversi tipi di dati

## Come si usano

Si devono aggiungere delle keywords prima di implementare classi o funzioni

- template<class nome1, class nome2> oppure
- template<typename nome1, typename nome2>

# Esempio (1)

Stampiamo un array di interi, un array di char e un array di string.

```
void printArrayInt(const int* array, int size) {
  for(int i=0;i < size; i++)
    cout << array[i] << endl;
}

void printArrayChar(const char* array, int size) {
  for(int i=0;i < size; i++)
    cout << array[i] << endl;
}

void printArrayString(const string* array, int size) {
  for(int i=0;i < size; i++)
    cout << array[i] << endl;
}
```

# Esempio (2)

I template permettono di generalizzare il codice.

```
template<class T>
void printArray(const T* array, int size) {
  for(int i=0;i < size; i++)
    cout << array[i] << endl;
}
```

# Template e classi

## PairInt.h

```
#ifndef PAIRINT_H
#define PAIRINT_H

class PairInt {

    public:
        PairInt(int f, int s);
        int getFirst() const;
        int getSecond() const;
        void setFirst(int f);
        void setSecond(int s);

    private:
        int first;
        int second;
};

#endif
```

## PairInt.cpp

```
#include "PairInt.h"

PairInt::PairInt(int f, int s) {
    first = f;
    second = s;
}

int PairInt::getFirst() const {
    return first;
}

int PairInt::getSecond() const {
    return second;
}

void PairInt::setFirst(int f) {
    first = f;
}

void PairInt::setSecond(int s) {
    second = s;
}
```

# Template e classi

## Pair.h

```
#ifndef PAIR_H
#define PAIR_H

template<class T>
class Pair {

  public:
    Pair(T f, T s);
    T getFirst() const;
    T getSecond() const;
    void setFirst(T f);
    void setSecond(T s);

  private:
    T first;
    T second;
};

template<class T>
Pair<T>::Pair(T f, T s) {
  first = f;
  second = s;
}
```

## Pair.h: Implementazione nel file .h

```
template<class T>
T Pair<T>::getFirst() const {
  return first;
}

template<class T>
T Pair<T>::getSecond() const {
  return second;
}

template<class T>
void Pair<T>::setFirst(T f) {
  first = f;
}

template<class T>
void Pair<T>::setSecond(T s) {
  second = s;
}
#endif
```

# Template e classi

## main.cpp

Come usare le classi template:

```
#include "PairInt.h"
#include "Pair.h"
#include <iostream>
#include <string>
using namespace std;

int main() {
    PairInt p1(1,2);
    Pair<int> p2(1,2);
    Pair<string> p3("ciao","mondo");

    cout << p1.getFirst() << endl;
    cout << p2.getFirst() << endl;
    cout << p3.getFirst() << " " << p3.getSecond() << endl;
}
```