Plantillas (Templates)

Carlos E. Alvarez[1].

[1]Dep. de Matemáticas aplicadas y Ciencias de la Computación, Universidad del Rosario

2019-II

Polimorfismo

Función `max` para diferentes tipos de datos

int

```
int GetMax(int x, int y) {
  return (x > y) ? x : y;
}
```

double

```
double GetMax(double x, double y) {
  return (x > y) ? x : y;
}
```

¡¡¡REDUNDANTE!!!

Plantillas

Nos permiten definir funciones y estructuras a las que el tipo de dato se les presenta al momento de compilar

template

```
template <typename T>
T GetMax(T x, T y) {
  return (x > y) ? x : y;
}
```

Plantillas

Uso

```
int main() {
  int i=5, j=6, k;
  long l=10, m=5, n;

  k = GetMax<int>(i,j);
  n = GetMax<long>(l,m);

  cout << k << endl;
  cout << n << endl;
  return 0;
}
```

Convirtiendo CharList en la plantilla MyList

mylist.hpp

```
template <typename T>
class MyList {
private:
  struct Cell {
    T ch;
    Cell *next;
  };

  Cell *back; //pointer to end of list

public:
  MyList() : back(nullptr) {}
  ~MyList();

  bool empty();
  void push_back(T c); //create element at end of list
  void pop_back(); //delete element at end of list
};
```

Convirtiendo CharList en la plantilla MyList

mylist.hpp

```
template <typename T>
MyList<T>::~MyList() {
  for(back; back != nullptr; back = back->next){
    delete back;
    count--;
  }
}

template <typename T>
bool MyList<T>::empty() {
  return back == nullptr;
}
```

Convirtiendo CharList en la plantilla MyList

mylist.hpp

```
template <typename T>
void MyList<T>::push_back(T c) {
  Cell *newCell = new Cell;
  newCell->ch = c;
  if(empty())
    newCell->next = nullptr;
  else
    newCell->next = back;
  back = newCell;
  cout << "pushing " << back->ch << endl;
}
```

Convirtiendo CharList en la plantilla MyList

mylist.hpp

```
template <typename T>
void MyList<T>::pop_back() {
  Cell *cursor;
  if(back != nullptr){
    cursor = back->next;
    cout << "popping " << back->ch << endl;
    delete back;
    back = cursor;
  }
}
```