# Vectores y Listas

Carlos E. Alvarez[1].

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

2019-II

# Vectores

## Inicialización

```
#include <iostream>
#include <vector>

int main(){
  std::vector<int> first;  //empty vector of ints
  std::vector<int> second(4, 100); //four ints (value 100)
  std::vector<int> third(second);  //a copy of third

  //the iterator constructor can also be used to
  // construct from arrays:
  int myints[] = {16, 2, 77, 29};
  std::vector<int> fourth(myints, myints +
            sizeof(myints) / sizeof(int));
```

# Vectores

```
  std::cout << "The contents of fourth are:";
  for (unsigned int i = 0; i < fourth.size(); ++i)
    std::cout << ' ' << fourth[i];
  std::cout << '\n';

  return 0;
}
```

# Vectores

## pop_back / push_back

```
int main(){
  std::vector<int> myvector;
  int sum(0);

  myvector.push_back(100);
  myvector.push_back(200);
  myvector.push_back(300);

  while (!myvector.empty()) {
    sum += myvector.back();
    myvector.pop_back();
  }
  std::cout << "The elements of myvector add up to "
            << sum << '\n';

  return 0;
}
```

# Vectores

## Operador [ ]

```
int main(){
  std::vector<int> myvector(10);
  unsigned int sz = myvector.size();

  // assign some values:
  for (unsigned int i = 0; i < sz; i++)
    myvector[i] = i;

  std::cout << "myvector contains:";
  for (unsigned int i = 0; i < sz; i++)
    std::cout << ' ' << myvector[i];
  std::cout << '\n';

  return 0;
}
```

# Vectores

Ejercicios:

1. Cree un vector de float vacío y llénelo con los múltiplos de $\pi$ del 1 al 10. Imprima el resultado
2. Cree una función que reciba dos vectores de int y retorne la suma de estos en otro vector
3. Cree un vector de double con 10 elementos y vacíelo

# Listas

## Inicialización

```
#include <iostream>
#include <list>

int main(){
  std::list<int> first;   //empty list of ints
  std::list<int> second(4, 100);//four ints (value 100)
  std::list<int> third(second); // a copy of third

  // the iterator constructor can also be used to
  // construct from arrays:
  int myints[] = {16, 2, 77, 29};
  std::list<int> fourth(myints, myints +
                  sizeof(myints) / sizeof(int));
  return 0;
}
```

# Listas

## front / back

```
int main(){
  std::list<int> mylist;

  mylist.push_back(77);
  mylist.push_back(22);
  // now front equals 77, and back 22

  mylist.front() -= mylist.back();

  std::cout << "mylist.front() is now "
            << mylist.front() << '\n';

  return 0;
}
```

# Listas

## empty

```
int main (){
  std::list<int> mylist;
  int sum(0);

  for (int i = 1;i <= 10;++i)
    mylist.push_back(i);

  while (!mylist.empty()) {
    sum += mylist.front();
    mylist.pop_front();
  }

  std::cout << "total: " << sum << '\n';

  return 0;
}
```