Università della Calabria

Corso di Laurea in Ingegneria Elettronica

Programmazione Orientata agli Oggetti

A.A. 2006/2007

Esempi di strutture dati: code e pile

Paolo Trunfio *

* DEIS, Università della Calabria – http://si.deis.unical.it/~trunfio

Una classe Coda

```
// Coda.h
#include <assert.h>
#include <iostream>
using namespace std;
class Coda {
  private:
    class Nodo {
      public:
        int valore;
        Nodo *successivo;
        Nodo (int val) {
          valore = val;
          successivo = 0;
        }
    };
    Nodo *primo;
    Nodo *ultimo;
    int size;
```

```
struct Nodo {
  int valore;
  Nodo *successivo;
  Nodo (int val) {
    valore = val;
    successivo = 0;
  }
};
```

Una classe Coda

```
public:
  Coda () {
    primo = 0;
    ultimo = 0;
    size = 0;
  }
  void inserisci (int val) {
    Nodo *nodo = new Nodo(val);
    assert(nodo != 0);
    if (size == 0) {
      primo = nodo;
      ultimo = nodo;
    }
    else { // inserisce in coda
      (*ultimo).successivo = nodo;
      ultimo = nodo;
    }
    size++;
  }
```


Una classe Coda

```
  int rimuovi () {
    assert(size != 0);
    Nodo *nodo = primo;
    int val = (*nodo).valore;
    if (size == 1) {
      primo = 0;
      ultimo = 0;
    }
    else {
      primo = (*nodo).successivo;
    }
    delete nodo;
    size--;
    return val;
  }
```

Una classe Coda

```
    int getSize() const {
      return size;
    }

    void stampa() const {
      Nodo *nodo = primo;
      cout << "[ ";
      while (nodo != 0) {
        cout << (*nodo).valore << " ";
        nodo = (*nodo).successivo;
      }
      cout << "]" << endl;
    }

};
```


Una classe Coda

```
#include "Coda.h"
#include <iostream>
using namespace std;

main () {
  Coda c;
  for (int i=0; i<10; i++) {
    c.inserisci(i);
    cout << "Inserito: " << i << " -> Coda = ";
    c.stampa();
  }
  while (c.getSize() > 0) {
    int x = c.rimuovi();
    cout << "Rimosso : " << x << " -> Coda = ";
    c.stampa();
  }
  system("pause");
}
```

Una classe Coda

```
#include "Coda.h"
#include <iostream>
using namespace std;

main () {
  Coda *c = new Coda;
  for (int i=0; i<10; i++) {
    c->inserisci(i);
    cout << "Inserito: " << i << " -> Coda = ";
    c->stampa();
  }
  while (c->getSize() > 0) {
    int x = c->rimuovi();
    cout << "Rimosso : " << x << " -> Coda = ";
    c->stampa();
  }
  system("pause");
  delete c;
}
```


Una classe Coda

```
Inserito: 0 -> Coda = [ 0 ]
Inserito: 1 -> Coda = [ 0 1 ]
Inserito: 2 -> Coda = [ 0 1 2 ]
Inserito: 3 -> Coda = [ 0 1 2 3 ]
Inserito: 4 -> Coda = [ 0 1 2 3 4 ]
Inserito: 5 -> Coda = [ 0 1 2 3 4 5 ]
Inserito: 6 -> Coda = [ 0 1 2 3 4 5 6 ]
Inserito: 7 -> Coda = [ 0 1 2 3 4 5 6 7 ]
Inserito: 8 -> Coda = [ 0 1 2 3 4 5 6 7 8 ]
Inserito: 9 -> Coda = [ 0 1 2 3 4 5 6 7 8 9 ]
Rimosso : 0 -> Coda = [ 1 2 3 4 5 6 7 8 9 ]
Rimosso : 1 -> Coda = [ 2 3 4 5 6 7 8 9 ]
Rimosso : 2 -> Coda = [ 3 4 5 6 7 8 9 ]
Rimosso : 3 -> Coda = [ 4 5 6 7 8 9 ]
Rimosso : 4 -> Coda = [ 5 6 7 8 9 ]
Rimosso : 5 -> Coda = [ 6 7 8 9 ]
Rimosso : 6 -> Coda = [ 7 8 9 ]
Rimosso : 7 -> Coda = [ 8 9 ]
Rimosso : 8 -> Coda = [ 9 ]
Rimosso : 9 -> Coda = [ ]
```

Una classe Pila

```
// Pila.h
#include <assert.h>
#include <iostream>
using namespace std;
class Pila {
  private:
    class Nodo {
      public:
        int valore;
        Nodo *successivo;
        Nodo (int val) {
          valore = val;
          successivo = 0;
        }
    };
    Nodo *primo;
    Nodo *ultimo;
    int size;
```

come nella classe Coda

Una classe Pila

```
  public:
    Pila () {
      primo = 0;
      ultimo = 0;
      size = 0;
    }
    void inserisci (int val) {
      Nodo *nodo = new Nodo(val);
      assert(nodo != 0);
      if (size == 0) {
        primo = nodo;
        ultimo = nodo;
      }
      else { // inserisce in testa
        (*nodo).successivo = primo;
        primo = nodo;
      }
      size++;
    }
```

Una classe Pila

```
    int rimuovi () {
      ...come nella Coda...
    }

    int getSize() const {
      ...come nella Coda...
    }

    void stampa() const {
      ...come nella Coda...
    }

};
```


Una classe Pila

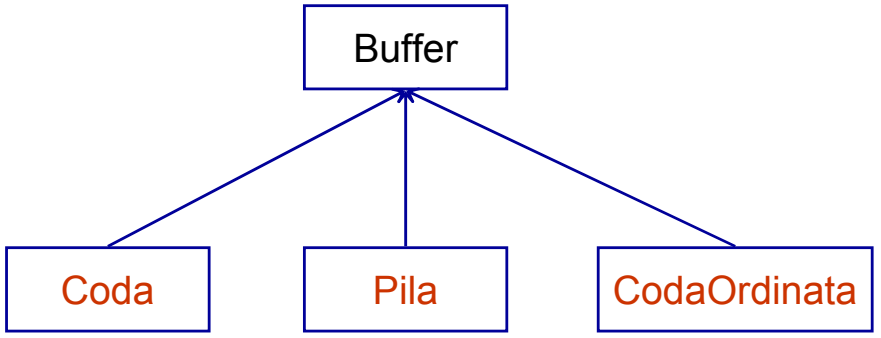
```
#include "Pila.h"
#include <iostream>
using namespace std;

main () {
  Pila *p = new Pila;
  for (int i=0; i<10; i++) {
    p->inserisci(i);
    cout << "Inserito: " << i << " -> Pila = ";
    p->stampa();
  }
  while (p->getSize() > 0) {
    int x = p->rimuovi();
    cout << "Rimosso : " << x << " -> Pila = ";
    p->stampa();
  }
  system("pause");
  delete p;
}
```

Una classe Pila

```
Inserito: 0 -> Pila = [ 0 ]
Inserito: 1 -> Pila = [ 1 0 ]
Inserito: 2 -> Pila = [ 2 1 0 ]
Inserito: 3 -> Pila = [ 3 2 1 0 ]
Inserito: 4 -> Pila = [ 4 3 2 1 0 ]
Inserito: 5 -> Pila = [ 5 4 3 2 1 0 ]
Inserito: 6 -> Pila = [ 6 5 4 3 2 1 0 ]
Inserito: 7 -> Pila = [ 7 6 5 4 3 2 1 0 ]
Inserito: 8 -> Pila = [ 8 7 6 5 4 3 2 1 0 ]
Inserito: 9 -> Pila = [ 9 8 7 6 5 4 3 2 1 0 ]
Rimosso : 9 -> Pila = [ 8 7 6 5 4 3 2 1 0 ]
Rimosso : 8 -> Pila = [ 7 6 5 4 3 2 1 0 ]
Rimosso : 7 -> Pila = [ 6 5 4 3 2 1 0 ]
Rimosso : 6 -> Pila = [ 5 4 3 2 1 0 ]
Rimosso : 5 -> Pila = [ 4 3 2 1 0 ]
Rimosso : 4 -> Pila = [ 3 2 1 0 ]
Rimosso : 3 -> Pila = [ 2 1 0 ]
Rimosso : 2 -> Pila = [ 1 0 ]
Rimosso : 1 -> Pila = [ 0 ]
Rimosso : 0 -> Pila = [ ]
```


Una superclasse Buffer



Una superclasse Buffer

```
// Buffer.h
#include <assert.h>
#include <iostream>
using namespace std;
class Buffer {
  protected: // <-- Nota
    class Nodo {
      public:
        int valore;
        Nodo *successivo;
        Nodo (int val) {
          valore = val;
          successivo = 0;
        }
    };
    Nodo *primo;
    Nodo *ultimo;
    int size;
```


Una superclasse Buffer

```
  public:
    Buffer () {
      primo = 0;
      ultimo = 0;
      size = 0;
    }
    virtual void inserisci (int val) = 0;
    int rimuovi () {
      ...come nella Coda...
    }
    int getSize() const {
      ...come nella Coda...
    }
    void stampa() const {
      ...come nella Coda...
    }
};
```

Buffer: Coda

```
class Coda: public Buffer {
  public:
    void inserisci (int val) {
      Nodo *nodo = new Nodo(val);
      assert(nodo != 0);
      if (size == 0) {
        primo = nodo;
        ultimo = nodo;
      }
      else {
        (*ultimo).successivo = nodo;
        ultimo = nodo;
      }
      size++;
    }
};
```


Buffer: Pila

```
class Pila: public Buffer {
  public:
    void inserisci (int val) {
      Nodo *nodo = new Nodo(val);
      assert(nodo != 0);
      if (size == 0) {
        primo = nodo;
        ultimo = nodo;
      }
      else {
        (*nodo).successivo = primo;
        primo = nodo;
      }
      size++;
    }
};
```

Buffer: CodaOrdinata

```
class CodaOrdinata: public Buffer {
 public:
  void inserisci (int val) {
   Nodo *nodo = new Nodo(val);
   assert(nodo != 0);
   if (size == 0) {
    primo = nodo;
    ultimo = nodo;
   }
   else {
    Nodo *corrente = primo;
    Nodo *precedente = 0;
    while ( corrente != 0 && (*corrente).valore < val ) {
     precedente = corrente;
     corrente = (*corrente).successivo;
    }
```


Buffer: CodaOrdinata

```
    if (precedente == 0) { // inserisce in testa
     (*nodo).successivo = corrente;
     primo = nodo;
    }
    else if (corrente == 0) { // inserisce in coda
     (*ultimo).successivo = nodo;
     ultimo = nodo;
    }
    else {
     (*precedente).successivo = nodo;
     (*nodo).successivo = corrente;
    }
   }
   size++;
  }
};
```

CodaOrdinata

```
#include "Buffer.h"
#include <iostream>
using namespace std;

main () {
  Buffer *b = new CodaOrdinata;
  for (int i=1; i<=6; i++) {
    int x; cout << "Valore da inserire: "; cin >> x;
    b->inserisci(x);
    cout << "Inserito: " << x << " -> Buffer = ";
    b->stampa();
  }
  while (b->getSize() > 0) {
    int x = b->rimuovi();
    cout << "Rimosso : " << x << " -> Buffer = ";
    b->stampa();
  }
  system("pause");
  delete b;
}
```


CodaOrdinata

```
Valore da inserire: 4
Inserito: 4 -> Buffer = [ 4 ]
Valore da inserire: 2
Inserito: 2 -> Buffer = [ 2 4 ]
Valore da inserire: 8
Inserito: 8 -> Buffer = [ 2 4 8 ]
Valore da inserire: 5
Inserito: 5 -> Buffer = [ 2 4 5 8 ]
Valore da inserire: 4
Inserito: 4 -> Buffer = [ 2 4 4 5 8 ]
Valore da inserire: 0
Inserito: 0 -> Buffer = [ 0 2 4 4 5 8 ]
Rimosso : 0 -> Buffer = [ 2 4 4 5 8 ]
Rimosso : 2 -> Buffer = [ 4 4 5 8 ]
Rimosso : 4 -> Buffer = [ 4 5 8 ]
Rimosso : 4 -> Buffer = [ 5 8 ]
Rimosso : 5 -> Buffer = [ 8 ]
Rimosso : 8 -> Buffer = [ ]
```

Una classe Coda con i template

```
// CodaT.h
#include <assert.h>
#include <iostream>
using namespace std;
template<class Tipo>
class Coda {
  private:
    class Nodo {
      public:
        Tipo valore;
        Nodo *successivo;
        Nodo (const Tipo &val) {
          valore = val;
          successivo = 0;
        }
    };
    Nodo *primo;
    Nodo *ultimo;
    int size;
```


Una classe Coda con i template

```
  public:
    Coda () {
      primo = 0;
      ultimo = 0;
      size = 0;
    }
    void inserisci (const Tipo &val) {
      Nodo *nodo = new Nodo(val);
      assert(nodo != 0);
      if (size == 0) {
        primo = nodo;
        ultimo = nodo;
      }
      else {
        (*ultimo).successivo = nodo;
        ultimo = nodo;
      }
      size++;
    }
```

Una classe Coda con i template

```
    Tipo rimuovi () {
      assert(size != 0);
      Nodo *nodo = primo;
      Tipo val = (*nodo).valore;
      if (size == 1) {
        primo = 0;
        ultimo = 0;
      }
      else {
        primo = (*nodo).successivo;
      }
      delete nodo;
      size--;
      return val;
    }
```


Una classe Coda con i template

```
    int getSize() const {
      return size;
    }

    void stampa() const {
      Nodo *nodo = primo;
      cout << "[ ";
      while (nodo != 0) {
        cout << (*nodo).valore << " ";
        nodo = (*nodo).successivo;
      }
      cout << "]" << endl;
    }

};
```

Una classe Coda con i template

```
#include "CodaT.h"
#include <iostream>
using namespace std;

main () {
  Coda<int> *c = new Coda<int>;
  for (int i=0; i<10; i++) {
    c->inserisci(i);
    cout << "Inserito: " << i << " -> Coda = ";
    c->stampa();
  }
  while (c->getSize() > 0) {
    int x = c->rimuovi();
    cout << "Rimosso : " << x << " -> Coda = ";
    c->stampa();
  }
  system("pause");
  delete c;
}
```