Esempio di gestione di una CODA mediante Array oppure *container Queue*

prof.ssa P.Grandillo

ARRAY — *coda_array.ccp*

```
#include <iostream>
#include <fstream>
  using namespace std;

#define DIM 10
   int Testa=0, Coda=-1;
   int  FILA  [DIM];          //-- dichiarazione  array FILA

   ifstream finp  ("in_coda.txt");

void scriviCoda () {     //-- stampa tutti gli elementi nella Coda
    cout<<"\n\nin   FILA   persone : ";
    for(int i=0; i<=Coda; i++)
        cout << FILA [i]<< ' ';     //-- elemento nella Coda
}
int Push  (int per){
    if (Coda == DIM-1){       //-- errore: array ormai pieno
        cout << "\nFILA piena, persona: "<< per;
        cout <<" non inserita\n";
        return -1;
    }
    Coda++;
    FILA [Coda]  = per;          //-- inserimento nella Coda
    return 0;
}
int Pop  (int& perT){
    if (Coda == -1) {            //-- errore: Coda vuota
        cout<< "\n\nFILA vuota, nessuno da servire";
        return -1;     } //FINE if
    scriviCoda ();                    //-- OK, Coda non vuota
    perT  = FILA[Testa];      //-- salvato in perT ultimo elem.
    for (int i=1; i<=Coda; i++)
        FILA[i-1] = FILA[i];   //-- shift Coda a sinistra
    Coda --;          //-- aggiornam. indice Coda a penult.elem.
    return 0;   } //FINE funzione Pop
```

container QUEUE — *coda_queue.ccp*

```
#include <iostream>
#include <fstream>
  using namespace std;

#include <queue>
  queue <int> FILA;          //-- dichiarazione  queue FILA

  ifstream finp  ("in_coda.txt");

void scriviCoda () {
    cout << "\n\nin FILA persone: ";        //-- stampa:
    cout << FILA.size();     //-- numero di elementi nella coda
    cout << " - prima: "   << FILA.front();
    cout << " - ultima: " << FILA.back();
         //-- primo e ultimo elemento, unici elementi accessibili
}


int Push  (int per){

    FILA.push (per);                 //-- inserimento nella Coda
    return 0;
}

int Pop  (int& perT){

    if ( FILA.empty() ) {       //-- errore: Coda vuota
        cout<< "\n\nFILA vuota, nessuno da servire";
        return -1;
    }
    scriviCoda ();                    //-- OK, Coda non vuota
    perT  = FILA.front();    //-- salvato in perT l'ultimo elem.
    FILA.pop();              //-- rimosso primo elem. dalla Coda
    return 0;
}
```

Esempio di gestione di una CODA mediante Array oppure *container Queue*


```
int main(){
  int PER, perCoda;   //-- PER persona letta da file
  int N, esito=0;     //-- perCoda prelevata da array FILA

    finp >> N;

    for (int i=0; i<N && esito==0; i++){
        finp >> PER;
        if (PER > 0)                  //-- arriva persona
            esito = Push(PER);        //-- inserimento in Coda
        else {                                //-- parentesi chiusa
            esito = Pop (perCoda);    //- prelievo da Coda
            if (esito == 0)           //-- e controllo
              cout <<"\nservita la persona: "<<perCoda;
        }
    }
    if ( esito == 0)
      if (Coda > -1) {
        cout <<"\n\n errore: Fila ancora piena";
        scriviCoda ();
      }
      else {
        cout <<"\n\n OK - ";
        cout <<"non ci sono altre persone in Fila";
      }
    cout << endl;
    finp.close();
}
```

per approfondimenti consultare :

http://www.cplusplus.com/reference/queue/queue/

```
int main(){
  int PER, perCoda;
  int N, esito=0;

    finp >> N;

    for (int i=0; i<N && esito==0; i++) {
       finp >> PER;
       if (PER > 0)           //---arriva persona
          esito = Push(PER);
       else {                 //---esce persona
          esito = Pop (perCoda);
          if (esito == 0)
             cout <<"\nservita la persona: "<<perCoda;
       }
    }
    if (esito == 0)
      if (FILA.size() > 0) {
        cout <<"\n\n errore: Fila ancora piena";
        scriviCoda ();
      }
      else {
        cout <<"\n\n OK - ";
        cout <<"non ci sono altre persone in Fila";
      }
    cout << endl;
    finp.close();
}
```

*NOTA: con il container queue non è possibile accedere alle informazioni contenute nella Coda, ma solo al primo – **metodo** front() – e all'ultimo elemento inserito – metodo **metodo** top()*

OUTPUT con *coda_array.exe*	**OUTPUT con *coda_queue.exe***

caso 1 – file di input corretto → 10 **1 2 -1 3 4 -1 5 -1 -1 -1**

```
Prompt dei comandi
>>coda_array.exe


in  FILA  persone : 1 2
servita la persona: 1

in  FILA  persone : 2 3 4
servita la persona: 2

in  FILA  persone : 3 4 5
servita la persona: 3

in  FILA  persone : 4 5
servita la persona: 4

in  FILA  persone : 5
servita la persona: 5

 OK - non ci sono altre persone in Fila

>>
```

```
Prompt dei comandi
>>coda_queue.exe


in FILA persone: 2 - prima: 1 - ultima: 2
servita la persona: 1

in FILA persone: 3 - prima: 2 - ultima: 4
servita la persona: 2

in FILA persone: 3 - prima: 3 - ultima: 5
servita la persona: 3

in FILA persone: 2 - prima: 4 - ultima: 5
servita la persona: 4

in FILA persone: 1 - prima: 5 - ultima: 5
servita la persona: 5

 OK - non ci sono altre persone in Fila

>>
```

caso 2 – file di input errato → 6 **1 2 -1 -1 -1 3**
(chiamata terza persona, non ancora in fila)

```
Prompt dei comandi
>>coda_array.exe


in  FILA  persone : 1 2
servita la persona: 1

in  FILA  persone : 2
servita la persona: 2

FILA vuota, nessuno da servire

>>
```

```
Prompt dei comandi
>>coda_queue.exe


in FILA persone: 2 - prima: 1 - ultima: 2
servita la persona: 1

in FILA persone: 1 - prima: 2 - ultima: 2
servita la persona: 2

FILA vuota, nessuno da servire

>>
```

caso 3 – file di input errato (chiamate inferiori) → 6 **1 2 3 4 -1 -1**

```
Prompt dei comandi
>>coda_array.exe


in  FILA  persone : 1 2 3 4
servita la persona: 1

in  FILA  persone : 2 3 4
servita la persona: 2

 errore: Fila ancora piena

in  FILA  persone : 3 4

>>
```

```
Prompt dei comandi
>>coda_queue.exe


in FILA persone: 4 - prima: 1 - ultima: 4
servita la persona: 1

in FILA persone: 3 - prima: 2 - ultima: 4
servita la persona: 2

 errore: Fila ancora piena

in FILA persone: 2 - prima: 3 - ultima: 4

>>
```