Università della Calabria

Corso di Laurea in Ingegneria Elettronica

Programmazione Orientata agli Oggetti

A.A. 2006/2007

Ricorsione

Paolo Trunfio *

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

Calcolo iterativo del fattoriale

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

unsigned long fattoriale (unsigned long n) {
  unsigned long fatt = 1;
  for (unsigned long x = n; x > 1; x--)
    fatt *= x;
  return fatt;
}

int main () {
  unsigned long n = 6;
  unsigned long fatt = fattoriale(n);
  cout << n << "! = " << fatt << endl;  // 6! = 720
  system("pause");
}
```

Calcolo ricorsivo del fattoriale

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

unsigned long fattoriale (unsigned long n) {
  if (n == 0 || n == 1)
    return 1;
  else
    return n*fattoriale(n-1);
}

int main () {
  unsigned long n = 6;
  unsigned long fatt = fattoriale(n);
  cout << n << "! = " << fatt << endl;  // 6! = 720
  system("pause");
}
```


Merge sort (1)

```
void merge(int v[], int inizio, int medio, int fine);

void mergeSort (int v[], int inizio, int fine) {
  if (inizio < fine) {
    int medio = (inizio + fine)/2;
    mergeSort (v, inizio, medio);
    mergeSort (v, medio+1, fine);
    merge (v, inizio, medio, fine);
  }
}

int main () {
  int dim = 7;
  int v[] = {11, -3, 9, 22, 6, 41, -9};
  mergeSort (v, 0, dim-1);
  for (int i = 0; i < dim; i++)
    cout << v[i] << " ";
  cout << endl;
  system("pause");
}
```

Merge sort (2)

```
void merge(int v[], int inizio,
                int medio, int fine) {
  int dim = fine-inizio+1;
  int temp[dim];
  int x = 0;
  int i = inizio;
  int j = medio+1;
  while (i <= medio && j <= fine ) {
    if (v[i] < v[j]) {
      temp[x] = v[i];
      i++;
    }
    else {
      temp[x] = v[j];
      j++;
    }
    x++;
  }
  while (i <= medio) {
    temp[x] = v[i];
    i++;
    x++;
  }
  while (j <= fine) {
    temp[x] = v[j];
    j++;
    x++;
  }
  for (x = 0; x < dim; x++) {
    v[x+inizio] = temp[x];
  }
}
```


Merge sort con template (1)

```
template <class T>
void merge(T v[], int inizio, int medio, int fine);

template <class T>
void mergeSort (T v[], int inizio, int fine) {
  if (inizio < fine) {
    int medio = (inizio + fine)/2;
    mergeSort (v, inizio, medio);
    mergeSort (v, medio+1, fine);
    merge (v, inizio, medio, fine);
  }
}

template <class T>
void merge(T v[], int inizio, int medio, int fine) {
  int dim = fine-inizio+1;
  T temp[dim];
  …come nel lucido precedente…
}
```

Merge sort con template (2)

```
template <class T> void stampaArray (T v[], int dim) {
  for (int i=0; i<dim; i++)
    cout << v[i] << "  ";
  cout << endl;
}

int main () {
  int dim1 = 10;
  int v1[] = {11, -3, 9, 22, 6, 1, 41, -9, 19, 8};
  mergeSort (v1, 0, dim1-1);
  stampaArray(v1, dim1);
  int dim2 = 4;
  string v2[] = {"meccanica", "informatica", "civile", "elettronica"};
  mergeSort (v2, 0, dim2-1);
  stampaArray (v2, dim2);
  system("pause");
}
```

-9 -3 1 6 8 9 11 19 22 41

civile elettronica informatica meccanica