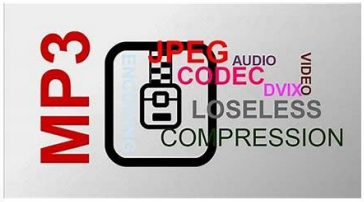
ITIS-LS “Francesco Giordani” Caserta

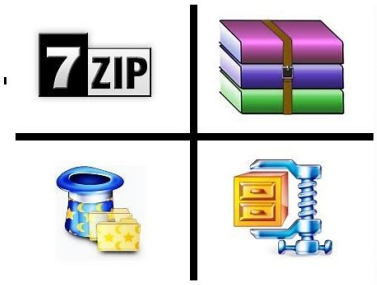
prof. Ennio Ranucci

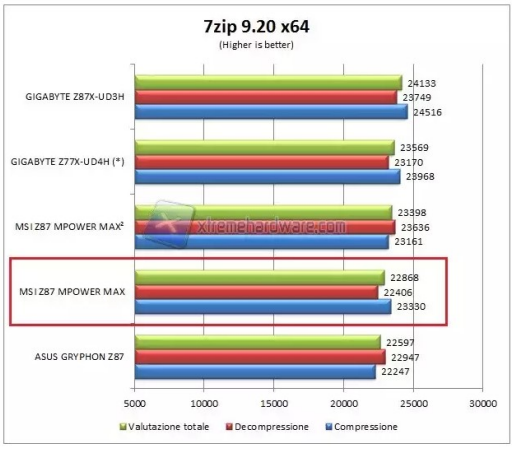
a.s. 2020-2021

La compressione dati senza perdita (o compressione dati lossless)

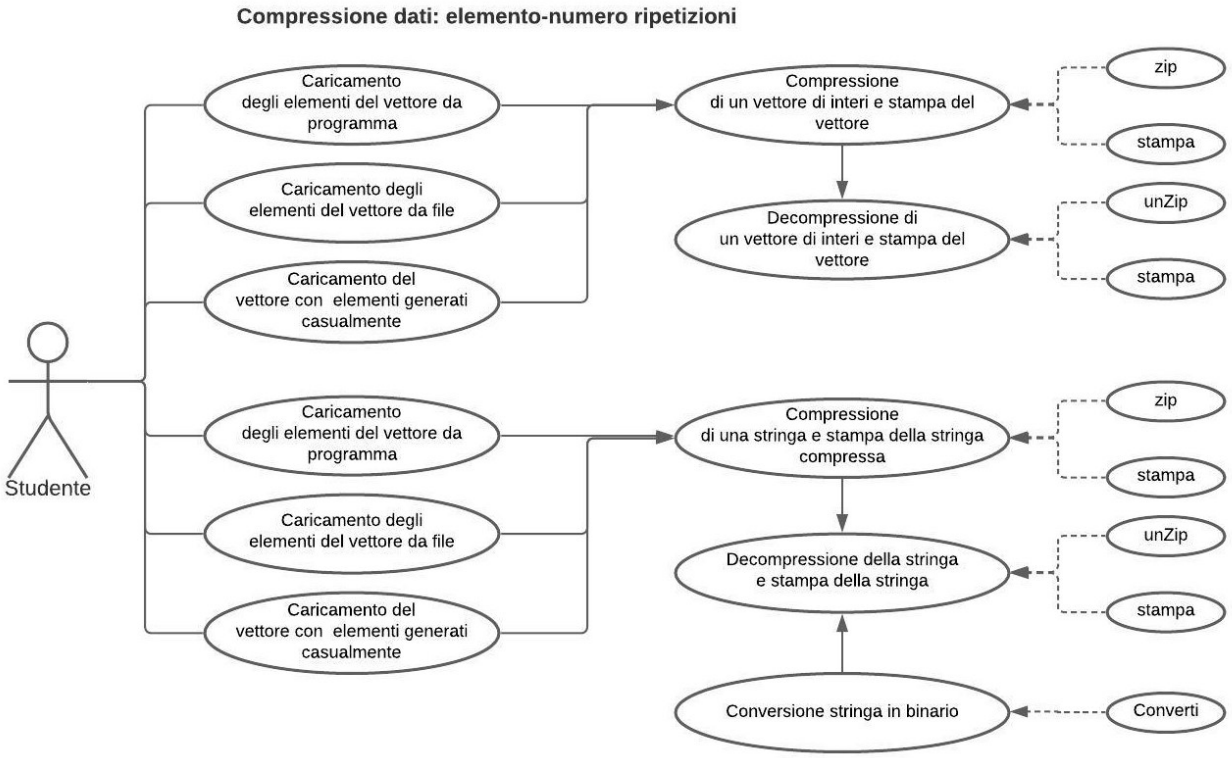
Codifica C++ e C#







Compressione lossness, in informatica è una classe di algoritmi che non porta alla perdita di alcuna parte dell'informazione originale durante la fase di compressione/decompressione dei dati stessi.
Gli algoritmi di compressione *lossless* non possono sempre garantire che ogni insieme di dati in input diminuisca di dimensione dopo l'esecuzione dell'algoritmo.
Molti applicativi che utilizzano la compressione lossless prevedono di lasciare invariati gli insiemi di dati la cui dimensione sia aumentata dopo la compressione.
Un algoritmo di compressione lossless di un insieme di dati potrebbe essere "elemento-numero di ripetizioni".



Implementazione che utlizza un vettore

```
#include <iostream>
#include <string>
using namespace std;
int vet[100] = {10,4,5,5,5,18,18,15,15,15,15,4,4,4,4};
int vet2[100]= {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
int dimLog=14;
int dimLogZip;
int vetZip[200];
int vetUnZip[100];
void zip(int vetPar[100],int dimLogPar);
void stampaVet(int vetPar[100],int dimLogPar);
void stampaVetZip(int vetPar[100],int dimLogPar);
void unZip(int vetPar[],int dimLogPar);
int main ()
 {
   zip(vet,dimLog);
   stampaVet(vet,dimLog);
   cout<<endl<<"dimensione vettore prima della compressione: "<<dimLog+1<<endl;
   stampaVetZip(vetZip,dimLogZip);
   cout <<endl<< "dimensione vettore compresso: "<<dimLogZip<<endl;
   unZip(vetZip,dimLogZip);
   cout <<"Vettore decompresso: "<<endl;
   stampaVet(vetUnZip,dimLog);
   return 0;
}
void stampaVet(int vetPar[100],int dimLogPar)
```

```
{
  for (int i=0; i<=dimLogPar;i++)
  {
    cout<<vetPar[i]<<" ";
  }
}
void stampaVetZip(int vetPar[100],int dimLogPar)
{
   for (int i=0; i<dimLogPar;i=i+2)
  {
     cout<<vetPar[i]<<"/"<<vetPar[i+1]<<" ";
  }
}
void zip(int vetPar[],int dimLogPar)
{
 int contatore,j=0,k=0;
  for (int i=0; i<=dimLogPar;i++)
    {
     contatore=0;
     j=i+1;
     while (vetPar[i]==vetPar[j])
      {
        contatore++;
        j++;
      }
     vetZip[k]=vetPar[i];
     vetZip[k+1]=contatore+1;
     k=k+2;
```

```
      i=j-1;
    }
  dimLogZip=k;
}
void unZip(int vetPar[],int dimLogPar)
{
  int k=0;
  for (int i=0; i<=dimLogPar;i=i+2)
    {
     for (int j=1;j<=vetPar[i+1];j++)
        {
         vetUnZip[k]=vetPar[i];
         k++;
        }
    }
}
```

Implementazione che utlizza una stringa

```
#include <iostream>
#include <string>
#include <sstream>
#include <stdlib.h>
using namespace std;
string str = "10000111111111111111111111110010011111";
string strZip="";
string strUnZip;
int dimStr;
string numeroRipetizioni;
void zip(string strPar);
void unZip(string strPar);
string toString(int n)
{
   ostringstream temp;
   temp<<n;
   return temp.str();
}
int toInt(string strPar)
{
   int dim=strPar.length();
   int result=0;
   for(int i=0; i<dim; i++)
     {
       result = result * 10 + ( strPar[i] - '0' );
     }
```

```
    return result;
}
int main ()
 {
   dimStr=str.length();
   cout<<str<<endl;
   cout<<"dimensione vettore prima della compressione: "<<dimStr<<endl;
   zip(str);
   cout<<strZip<<endl;
   int dimStrZip=strZip.length();
   cout <<"dimensione vettore compresso: "<<dimStrZip<<endl;
   unZip(strZip);
   cout <<"Stringa decompressa: "<<strUnZip<<endl;
   return 0;
}

void zip(string strPar)
{
  int contatore,j=0;
   for (int i=0; i<=dimStr-1;i++)
     {
       contatore=0;
       j=i+1;
       while (strPar[i]==strPar[j])
        {
          contatore++;
          j++;
        }
```

```
      contatore++;
      numeroRipetizioni= toString(contatore);
      strZip=strZip+strPar[i]+"#"+numeroRipetizioni+'#';
      i=j-1;
     }

}

void unZip(string strPar)
{
  strUnZip="";
  int dim=strPar.length()-1;
  int numeroRipetizioni;
  string ripetizioni;
  string elemento;
  for (int i=0;i<dim;i++)
    {
      elemento="";
      while (strPar[i]!='#')
        {
          elemento=elemento+strPar[i];
          i++;
        }
      i++;
      ripetizioni="";
      while (strPar[i]!='#')
        {
          ripetizioni=ripetizioni+strPar[i];
```

```
        i++;
      }
    numeroRipetizioni=toInt(ripetizioni);
    for(int k=1;k<=numeroRipetizioni;k++)
    {
      strUnZip=strUnZip+elemento;
    }
  }
}
```

Implementazione che trasforma la stringa di caratteri in una stringa binaria

```
#include <iostream>
#include <string>
#include <sstream>
#include <stdlib.h>
using namespace std;
string str; //= "10000111111111111111111111111110010011111";
string strZip="";
string strUnZip;
int dimStr;
string numeroRipetizioni;
string convertiStrBin(string strPar);
void zip(string strPar);
void unZip(string strPar);
string toString(int n)
{
   ostringstream temp;
   temp<<n;
   return temp.str();
}
int toInt(string strPar)
{
   int dim=strPar.length();
   int result=0;
   for(int i=0; i<dim; i++)
     {
       result = result * 10 + ( strPar[i] - '0' );
     }
```

```
    return result;
}
int main ()
 {
   str=convertiStrBin("abbbbbccccccc");
   //str="abbbbbccccccc";
   dimStr=str.length();
   cout<<str<<endl;
   cout<<"dimensione vettore prima della compressione: "<<dimStr<<endl;
   zip(str);
   cout<<strZip<<endl;
   int dimStrZip=strZip.length();
   cout <<"dimensione vettore compresso: "<<dimStrZip<<endl;
   unZip(strZip);
   cout <<"Stringa decompressa: "<<strUnZip<<endl;
   return 0;
}
string convertiStrBin(string strPar)
{
    string strBin="";
    char car;
    string carBin;
    int n;
    char buffer[255];
    int dim=strPar.length();
    for(int i=0; i<dim; i++)
      {
        car=strPar[i];
```

```
        n=(int)car;
        carBin=itoa(n,buffer,2);
        strBin=strBin+carBin;
      }
     return strBin;
}
void zip(string strPar)
{
  int contatore,j=0;
   for (int i=0; i<=dimStr-1;i++)
     {
       contatore=0;
       j=i+1;
       while (strPar[i]==strPar[j])
        {
          contatore++;
          j++;
        }
       contatore++;
       numeroRipetizioni= toString(contatore);
       strZip=strZip+strPar[i]+"#"+numeroRipetizioni+'#';
       i=j-1;
     }
}
void unZip(string strPar)
{
   strUnZip="";
   int dim=strPar.length()-1;
```

```
  int numeroRipetizioni;
  string ripetizioni;
  string elemento;
  for (int i=0;i<dim;i++)
    {
      elemento="";
      while (strPar[i]!='#')
        {
          elemento=elemento+strPar[i];
          i++;
        }
      i++;
      ripetizioni="";
      while (strPar[i]!='#')
        {
          ripetizioni=ripetizioni+strPar[i];
          i++;
        }
      numeroRipetizioni=toInt(ripetizioni);
      for(int k=1;k<=numeroRipetizioni;k++)
      {
        strUnZip=strUnZip+elemento;
      }
    }
}
```

Caricamento da file e scrittura in file

```
#include <iostream>
#include <string>
#include <sstream>
#include <stdlib.h>
#include <fstream>

using namespace std;
string str; //= "10000111111111111111111111110010011111";
string strZip="";
string strUnZip;
int dimStr;
string numeroRipetizioni;
string convertiStrBin(string strPar);
void zip(string strPar);
void unZip(string strPar);
void caricaStrFileTesto(char nomeFisicoPar[]);
void salvaStrFileTestoCompresso(char nomeFisicoPar[]);
void stampaFileCompresso(char nomeFisicoPar[]);
void creaStrFileTesto(char nomeFisicoPar[]);
string toString(int n)
{
   ostringstream temp;
   temp<<n;
   return temp.str();
}
int toInt(string strPar)
{
```

```
    int dim=strPar.length();
    int result=0;
    for(int i=0; i<dim; i++)
      {
        result = result * 10 + ( strPar[i] - '0' );
      }
    return result;
}
int main ()
 {
   //str=convertiStrBin("abbbbbccccccc");
   //str="abbbbbccccccc";
//creaStrFileTesto("car.txt");
   caricaStrFileTesto("caratteri.txt");
   dimStr=str.length();
   cout<<str<<endl;
   cout<<"dimensione stringa prima della compressione: "<<dimStr<<endl;
   zip(str);
   cout<<strZip<<endl;
   salvaStrFileTestoCompresso("caratteriCompresso.txt");
   cout<<"file compresso"<<endl;
   stampaFileCompresso("caratteriCompresso.txt");
   cout<<endl;
   int dimStrZip=strZip.length();
   cout <<"dimensione vettore compresso: "<<dimStrZip<<endl;
   unZip(strZip);
   cout <<"Stringa decompressa: "<<strUnZip<<endl;
   return 0;
```

```
}
string convertiStrBin(string strPar)
{
   string strBin="";
   char car;
   string carBin;
   int n;
   char buffer[255];
   int dim=strPar.length();
   for(int i=0; i<dim; i++)
    {
      car=strPar[i];
      n=(int)car;
      carBin=itoa(n,buffer,2);
      strBin=strBin+carBin;
    }
    return strBin;



}

void zip(string strPar)
{
  int contatore,j=0;
   for (int i=0; i<=dimStr-1;i++)
    {
      contatore=0;
      j=i+1;
```

```
      while (strPar[i]==strPar[j])
       {
         contatore++;
         j++;
       }
      contatore++;
      numeroRipetizioni= toString(contatore);
      strZip=strZip+strPar[i]+"#"+numeroRipetizioni+"#";
      i=j-1;
     }

}

void unZip(string strPar)
{
   strUnZip="";
   int dim=strPar.length()-1;
   int numeroRipetizioni;
   string ripetizioni;
   string elemento;
   for (int i=0;i<dim;i++)
     {
       elemento="";
       while (strPar[i]!='#')
         {
           elemento=elemento+strPar[i];
           i++;
         }
```

```
        i++;
        ripetizioni="";
        while (strPar[i]!='#')
         {
           ripetizioni=ripetizioni+strPar[i];
           i++;
         }
        numeroRipetizioni=toInt(ripetizioni);
        for(int k=1;k<=numeroRipetizioni;k++)
        {
          strUnZip=strUnZip+elemento;
        }
      }
}

void caricaStrFileTesto(char nomeFisicoPar[])
{
  ifstream in(nomeFisicoPar);
  char ch; str="";
  while(!in.eof())
  {
    in>> ch;
    str=str+ch;
  }
  in.close();
}
void salvaStrFileTestoCompresso(char nomeFisicoPar[])
{
```

```
    ofstream out(nomeFisicoPar);
    out<<strZip;
    out.close();
}
void stampaFileCompresso(char nomeFisicoPar[])
{
  ifstream in(nomeFisicoPar);
  char ch;
  while(!in.eof())
  {
    in>> ch;
    if (in.eof()==false) cout<<ch;
  }
  in.close();
}
void creaStrFileTesto(char nomeFisicoPar[])
{
   string strInput;
   ofstream out(nomeFisicoPar);
   cout<<"inserire la stringa: ";
   cin>>strInput;
   out<<strInput;
   out.close();
 }
```