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

struct Atleti {
 string NOME;
 int  punti;
};

int main(){
  ofstream fout_b ("classifica_st.dat", ios::binary);
  Atleti atleta;
  string NOME;
  int  punti;

  cout<<endl;
  cout<<"sizeof(NOME)  ="<<sizeof(NOME)<<endl;
  cout<<"sizeof(atleta)="<<sizeof(atleta)<<endl;
  cout<<"indir.atletaP ="<< &atleta.punti <<endl;
  cout<<"indir.atletaN ="<< &atleta.NOME  <<endl;
  cout<<"indir.atleta  ="<< &atleta       <<endl;

  for (int i=0; i<4; i++){
      cout << "\natleta? ";
      cin >> NOME >> punti;

      atleta.NOME  = NOME;
      atleta.punti  = punti;

      fout_b.write( (char *) &atleta, sizeof(Atleti) );
  }
  fout_b.close();
  return 0;
}
```

Il programma ***classifica_string.cpp*** è simile a *classifica_txt_bin.cpp*, ma per le stringhe dei nomi utilizza oggetti della **classe *string* del C++**, invece di usare gli array di caratteri ***del C***.

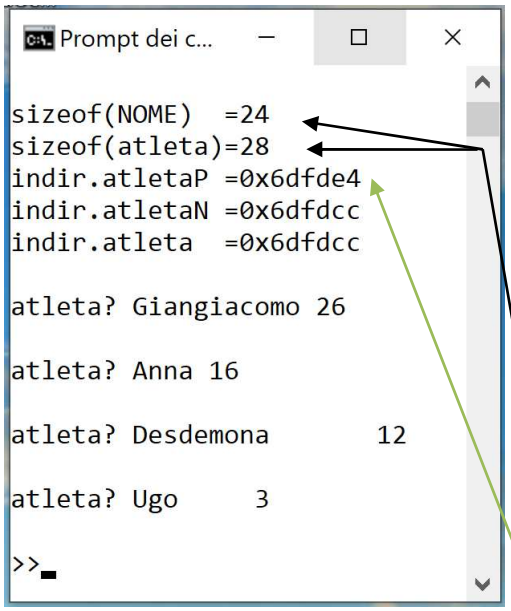
In **C** per gli array di char vanno usate le funzioni di copia (*strcpy*), confronto (*strcmp*) e concatenazione (*strcat*).

In **C++** per la classe *string* sono definiti i **più comodi operatori** di: assegnamento (=), confronto (<, >=,…) e "somma" (l'operatore **+** effettua la concatenazione di 2 stringhe).

In **C**, una stringa è semplicemente un array di caratteri che include sempre uno zero binario come elemento finale (indicato in C/C++ con '\0' detto NULL o terminatore di stringa).

In **C++** le stringhe riducono la possibilità per lo sviluppatore di fare gli errori di programmazione del C distruttivi (come superare le dimensioni dell'array), ma nascondono la rappresentazione fisica dei caratteri che contengono: **una stringa C++** oltre al suo **contenuto**, **memorizza anche** la **locazione di partenza** in memoria e la **lunghezza in caratteri** del suo contenuto.

INPUT da tastiera

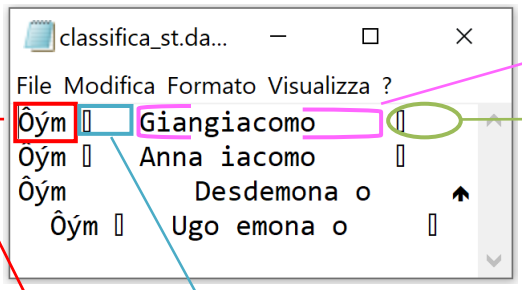


In **C**, ogni array di caratteri occupa una **regione fisica unica di memoria**. In **C++**, oggetti stringa individuali possono o no occupare regioni fisiche uniche di memoria. Inoltre, l'esatta implementazione della memoria per una classe *string* non è definita dallo standard del C++ (ogni compilatore la implementa in modo diverso).

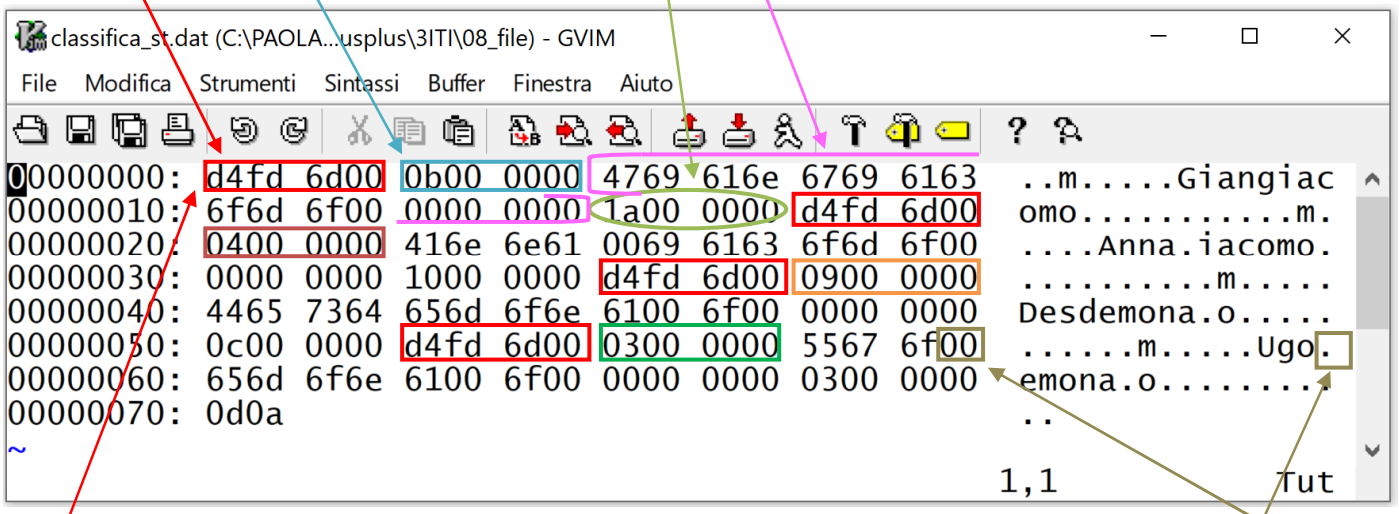
In questo esempio, sviluppato con *Code::Blocks* e compilatore *mingw32-g++.exe*, l'oggetto NOME di classe *string* occupa **24** byte, anche nella struttura **atleta** che occupa **28** byte (compresi i **4** byte del campo **punti**):

6DFDCC è l'indirizzo di partenza in memoria centrale della struttura **atleta** e del suo primo campo, la stringa NOME; esattamente dopo 24 byte inizia il campo **punti**, all'indirizzo **6DFDE4**.

OUTPUT su file binary



Nel file binary **classifica_st.dat** ogni record **atleta** di **28** byte è formato da: **8** byte **iniziali** seguiti da **16** byte per il **nome** e poi da **4** byte per **punti** (dato di tipo ***int***). Gli **8** byte iniziali sono composti da **4** byte per un **indirizzo** e **4** byte per la **lunghezza in caratteri** del **nome** (**0b**, cioè 11 lunghezza di *Giangiacomo*, **03** quella di *Ugo*).



NOTA: dopo ogni **nome** viene comunque inserito il terminatore di stringa (NULL cioè **'\0'**)

6DFDD4 è l'**indirizzo** di partenza dei caratteri del **nome** (è la posizione del carattere *G* di *Giangiacomo*)
Infatti aggiungendo **8** a **6DFDCC** (indirizzo di partenza della stringa) si ottiene **6DFDD4**, quindi i **4** byte iniziali sono il **puntatore** a **nome** e i successivi **4** byte sono un intero per la **lunghezza in caratteri**.

(https://www.micc.unifi.it/bertini/download/programmazione/TICPP-2nd-ed-Vol-two-printed.pdf p.27-28)

Questo comportamento apparentemente anomalo del programma (rispetto a *classifica_txt_bin.cpp* che per NOME usa un array di char) è dovuto al fatto che **non è possibile salvare oggetti C++ su file binary** come si farebbe con una struttura dati del C.

Il metodo **write** consente il salvataggio su file in binario di **dati elementari** (char, int, float, double) e di strutture composte dai precedenti (come gli **array di char**); **write** richiede il passaggio di un puntatore a carattere (**(char *)** &atleta) e del numero di caratteri da scrivere (**sizeof**(Atleti)) e trasferisce sul file binary, un byte dopo l'altro, il contenuto della struttura passata (**atleta**).

Utilizzando **write** sulla struttura ***C++* atleta**, l'intero contenuto della struttura viene trasferito: non solo i caratteri che costituiscono il **nome** e il punteggio (**punti**), ma **anche gli 8 byte iniziali della stringa NOME**: questo uso improprio di **write** ci ha però fornito informazioni sulla **tecnica di memorizzazione di un oggetto *string* del *C++* in Memoria Centrale**.

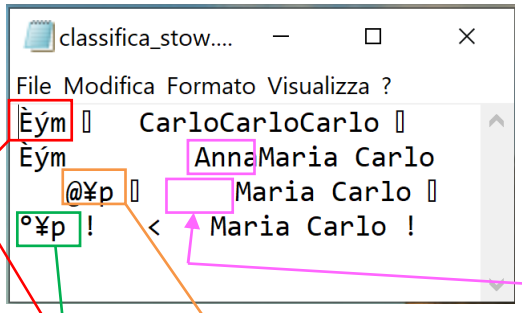
Ma cosa succede per stringhe più lunghe di 15 byte? (il valore massimo per string, teorico, è circa 2^{31} byte)

Eseguendo il programma con i dati indicati a destra →

CarloCarloCarlo 15
AnnaMaria 9
VittoriaVittoria 16
GiangiacomoGiangiacomoGiangiacomo 33

NOTA**: il primo NOME è di 15 caratteri, il secondo di 9, il terzo di 16, il quarto di 33: dovendo considerare anche il NULL, **il quarto nome NON può essere memorizzato nei 16 byte di nome !

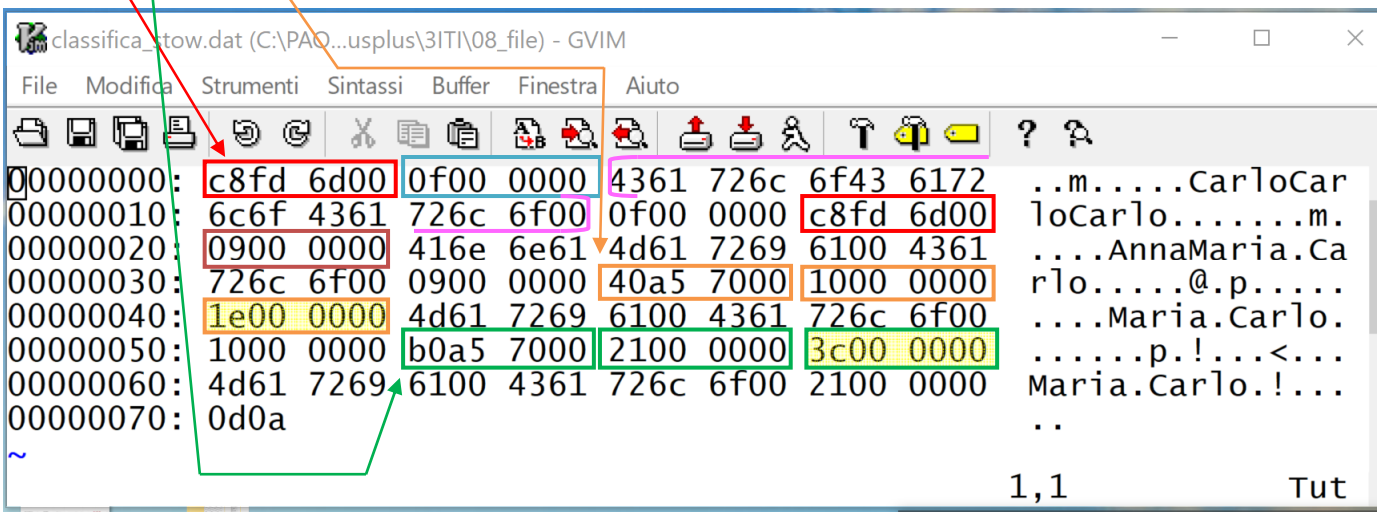
si ottiene il seguente **OUTPUT** sul file binary



i primi 2 record memorizzano il **nome** nei **16** byte della struttura **atleta** assegnati a NOME (utilizzando **0F=15** byte il primo, **09** byte il secondo), a partire dall'indirizzo **D6FDC8**

NOTA: il secondo nome è più corto e non ricopre tutto il contenuto del primo

gli ultimi 2 record non modificano questi **16** byte del **nome** (tranne i primi 4 byte di *Anna*) !



Per il terzo e quarto nome più lunghi di 15 caratteri (compreso il NULL), i 16 byte iniziali **non vengono sovrascritti**; invece:

- viene **assegnata un'altra zona di memoria** sufficientemente grande per contenere **nome**
- nei primi **4** byte (dei 24 iniziali assegnati alla *string* NOME) viene salvato l'**indirizzo** di questa **NUOVA zona di memoria** (indirizzo **70A540** per il terzo nome, **70A5B0** per il quarto nome)
- nei successivi **4** byte viene salvata (come sempre) la **lunghezza in caratteri** di **nome** (10_H=16 per il terzo nome, 21_H=33 per il quarto nome)
- nei successivi **4** byte viene registrata la **lunghezza in caratteri massima con cui può crescere la stringa** prima che l'oggetto stringa debba ridimensionare il suo buffer di dati interno (**1E=30** per il terzo nome, **3C=60** per il quarto nome che occupa più di **30** byte e, pertanto, si è reso necessario un secondo ridimensionamento del buffer interno per scrivere in M.C. il quarto nome)
- **nome** viene scritto a partire dall'indirizzo **70A540** per il terzo nome, **70A5B0** per il quarto nome.

*NOTA: l'indirizzo **6DFDD4** salvato su file nella prima esecuzione del programma e **D6FDC8** nella seconda esecuzione sono relativi ad una area dati di M.C. che è stata rilasciata al termine del programma, quindi non è più utilizzabile successivamente, perchè non è più valido!*

INUTILE quindi SALVARE su FILE i primi 8 byte! (e perdere il nome che ci interessa)

Per scrivere su file binary oggetti C++ (o comunque strutture contenenti puntatori) occorre un metodo più sofisticato: la **serializzazione**. (https://isocpp.org/wiki/faq/serialization)

Nel caso di oggetti *string* è sufficiente **copiare l'oggetto string in un array di char di lunghezza opportuna** e ricondurre la scrittura su file binary a quella usata nel programma *classifica_txt_bin.cpp*.

```
#include <iostream>
using namespace std;
#include <string.h>
#include <fstream>

#define DIM 28

struct Atleti {
 char NOME[DIM];
 int  punti;
};

int main(){

Ofstream fout_b("classifica_stdC.dat",ios::binary);

   Atleti atleta;
   string NOME;
   int  punti;

   for (int i=0; i<4; i++){
        cout << "\natleta? ";

        getline (cin, NOME);
        cin >> punti;

        cin.ignore (80, '\n');

        strncpy ( atleta.NOME, NOME.c_str(),
DIM);

        atleta.punti  = punti;
        if (NOME.length() >= DIM)
             atleta.NOME[DIM-1] = '\0';
      fout_b.write( (char *) &atleta, sizeof(Atleti) );
   }
   fout_b.close();
   return 0;
}
```

classifica_string_c_str.cpp procede alla **serializzazione** della stringa NOME utilizzando il **metodo c_str()** che restituisce nell'**array atleta.NOME** i caratteri contenuti nell'**oggetto string NOME** accodando anche il terminatore di stringa ('\0').

Per acquisire nomi composti (cioè un testo formato da parole separate da spazi) dobbiamo utilizzare la funzione **getline** a cui vanno passati lo *stream* di input (**cin**) e l'oggetto *string* (**NOME**).

http://www.cplusplus.com/reference/string/string/getline/

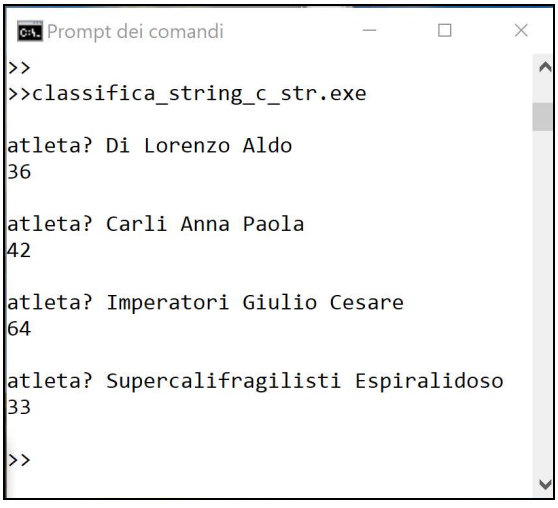
Dopo l'acquisizione di **punti** va eliminato il NULL con il metodo **ignore** di **cin** (PRO.TECH A - agg 311-312)

Mediante la funzione **strncpy** (libreria **string.h**) viene copiata nell'array di char **NOME** la serializzazione dell'oggetto *string* **NOME** ottenuta con il metodo **c_str()**

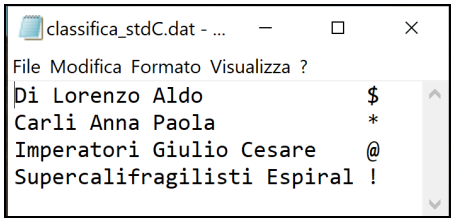
http://www.cplusplus.com/reference/cstring/strncpy/

Per **NOME** di lunghezza maggiore di 28 va inserito il terminatore di stringa (**'\0'**).

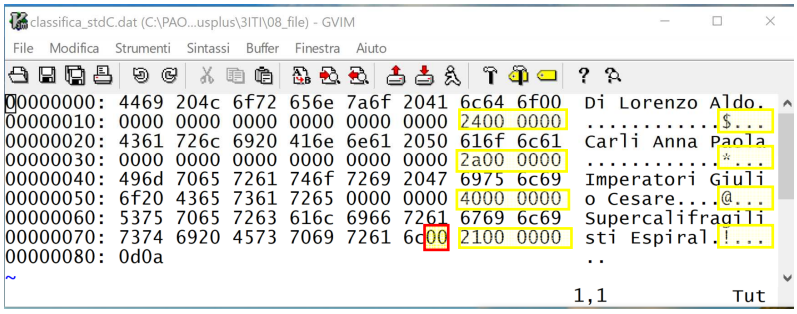
Per i dati di input indicati qui in basso



questo il file di output →



qui in basso il file di output aperto con *Vim*



Evidenziate in giallo le zone che contengono **punti** (24_H=36= '**$**' del primo record, $2A_H$=42='*****' del secondo, etc). Solo il quarto nome è stato troncato con aggiunta di **'\0'** perchè più lungo dei 28 byte disponibili.