Laurea triennale in Informatica

modulo (CFU 6) *di*

Programmazione II e Lab.

prof. Mariarosaria Rizzardi

Centro Direzionale di Napoli – Isola C4
stanza: n. 423 – IV piano Lato Nord
tel.: 081 547 6545
email: mariarosaria.rizzardi@uniparthenope.it

Programmazione in C++:

➢ overload di operatori

Esempio 1a

Si vuole creare una classe **Vettore2D** per implementare le operazioni sui vettori del piano:

$$u = \begin{pmatrix} u_x \\ u_y \end{pmatrix}, \quad v = \begin{pmatrix} v_x \\ v_y \end{pmatrix}, \quad w = u + v = \begin{pmatrix} u_x + v_x \\ u_y + v_y \end{pmatrix}, \quad \alpha = u \cdot v = u_x v_x + u_y v_y$$

somma vettoriale

prodotto scalare

Vettore2D.hpp (1ª versione)

```
class Vettore2D {
    float Vx, Vy;
  public :
    void setComponent(float v1, float v2);
    float* getComponent();
};
```

Vettore2D.cpp (1ª versione)

```
#include "Vettore2D.hpp"
void Vettore2D::setComponent(float v1, float v2)
{   this->Vx = v1;
    this->Vy = v2;
}
float* Vettore2D::getComponent()
{   float* pt = new float[2];
    pt[0] = this->Vx;
    pt[1] = this->Vy;
    return pt;
}
```

Esempio 1a (cont.)

esempio1a.cpp

```
#include <iostream>
#include "Vettore2D.hpp"
using namespace std;

int main()
{
    Vettore2D U, V;
    U.setComponent(2.5f, 1.0f);
    cout << "ascissa  di U: Ux = " << U.getComponent()[0] << endl;
    cout << "ordinata di U: Uy = " << U.getComponent()[1] << endl << endl;

    V.setComponent(0.5f, -1.0f);
    cout << "ascissa  di V: Vx = " << V.getComponent()[0] << endl;
    cout << "ordinata di V: Vy = " << V.getComponent()[1] << endl << endl;

   return 0;
}
```

```
ascissa  di U: Ux = 2.5
ordinata di U: Uy = 1

ascissa  di V: Vx = 0.5
ordinata di V: Vy = -1
```

Overload di operatori

In una classe si possono definire degli operatori (+, *, >,...) che sostituiscano quelli default del C++. Una funzione operatore, membro di una classe, è definito come:

```
return_type class_name::operator#(arg_list)
```

Il simbolo **#** indica l'operatore che si vuole ridefinire.

Esempio 1b

Vettore2D.hpp (2ª versione)

```
class Vettore2D {
    float Vx, Vy;
  public :
    void setComponent(float v1, float v2);
    float* getComponent();
    Vettore2D operator+(Vettore2D V2);
    float operator*(Vettore2D V2);
};
```

ridefinisce gli operatori +, * per gli oggetti di tipo Vettore2D

Vettore2D.cpp (2ª versione)

```
...
Vettore2D Vettore2D::operator+(Vettore2D V2)
{   Vettore2D tmp;
    tmp.Vx = this->Vx + V2.Vx;
    tmp.Vy = this->Vy + V2.Vy;
    return tmp;
}

float Vettore2D::operator*(Vettore2D V2)
{   float dot;
    dot = this->Vx*V2.Vx + this->Vy*V2.Vy;
    return dot;
}
```

Esempio 1b (cont.)

esempio1b.cpp

```
#include <iostream>
#include "Vettore2D.hpp"
using namespace std;

int main()
{
    Vettore2D U, V, W;
    U.setComponent(2.5f, 1.0f);
    cout << "ascissa  di U: Ux = " << U.getComponent()[0] << endl;
    cout << "ordinata di U: Uy = " << U.getComponent()[1] << endl << endl;

    V.setComponent(0.5f, -1.0f);
    cout << "ascissa  di V: Vx = " << V.getComponent()[0] << endl;
    cout << "ordinata di V: Vy = " << V.getComponent()[1] << endl << endl;

    W = U + V;
    cout << "vettore somma di due vettori:" << endl;
    cout << "ascissa  di W: Wx = " << W.getComponent()[0] << endl;
    cout << "ordinata di W: Wy = " << W.getComponent()[1] << endl << endl;

    float uv = U * V;
    cout << "prodotto scalare di due vettori = " << uv << endl;

   return 0;
}
```

```
...
somma di due vettori:
ascissa  di W: Wx = 3
ordinata di W: Wy = 0

prodotto scalare di due vettori = 0.25
```