

Tarea 4

Laboratorio Avanzado

Diego Sarceño

201900109

28 de febrero de 2023

Tomando como base la clase proporcionada por el profesor `VecR2` y se modificaron tanto los atributos, como los métodos a conveniencia para poder manejar vectores en 3 dimensiones. Se crean los 3 archivos a utilizar *t1l13.cpp*, *VecR3.cpp* y *VecR3.hpp* (adjuntos a este documento). Dicho esto, se agregan los operadores $+$, $-$, $*$ (escalar y producto punto), $=$, $==$, $/$ y $\%$ (producto cruz). Cuyos métodos se pueden ver a continuación:

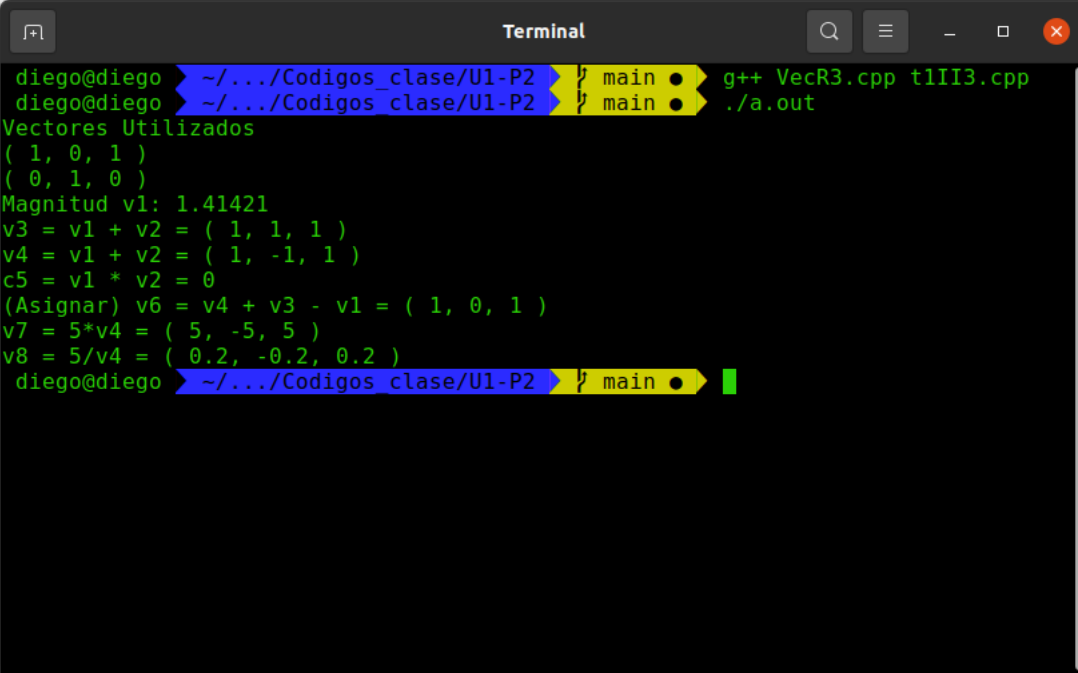
```
1  /* Magnitud */
2  float VecR3::Magnitud() const
3  {
4      return std::sqrt( Xcor*Xcor + Ycor*Ycor + Zcor*Zcor );
5  }
6
7
8  /* Operador suma */
9  VecR3 VecR3::operator+( const VecR3 &avec) const
10 {
11
12     VecR3 tmp;
13
14     tmp.Xcor = this->Xcor + avec.Xcor;
15     tmp.Ycor = this->Ycor + avec.Ycor;
16     tmp.Zcor = this->Zcor + avec.Zcor;
17
18     return tmp;
19 }
20
21
22 /* Operador Resta */
23 VecR3 VecR3::operator-( const VecR3 &avec) const
24 {
25     VecR3 tmp;
26
27     tmp.Xcor = this->Xcor - avec.Xcor;
28     tmp.Ycor = this->Ycor - avec.Ycor;
29     tmp.Zcor = this->Zcor - avec.Zcor;
30
31     return tmp;
32 }
33
34
```

```
35 /* Operador Producto punto */
36 float VecR3::operator*( const VecR3 &avec ) const
37 {
38     /* Ver los comentarios de operator+ */
39     float tmp;
40
41     tmp = this->Xcor * avec.Xcor + this->Ycor * avec.Ycor + this->Zcor *
        avec.Zcor;
42
43     return tmp;
44 }
45
46
47 /* Calcula el producto cruz de dos vectores */
48 VecR3 operator%( const VecR3 &avec )
49 {
50     VecR3 tmp;
51
52     tmp.Xcor = this->Ycor * avec.Zcor - this->Zcor * avec.Ycor;
53     tmp.Ycor = this->Xcor * avec.Zcor - this->Zcor * avec.Xcor;
54     tmp.Zcor = this->Xcor * avec.Ycor - this->Ycor * avec.Xcor;
55
56
57     return tmp;
58 }
59
60
61 /* Operador de asignacion */
62 VecR3 VecR3::operator=( const VecR3 &avec)
63 {
64     /* El vector que llama el operador es el que
65      * esta al lado izquierdo de este, y el que
66      * esta al lado derecho se pasa como argumento
67      * por lo que a "this" se le debe asignar el
68      * valor del argumento */
69     this->Xcor = avec.Xcor;
70     this->Ycor = avec.Ycor;
71     this->Zcor = avec.Zcor;
72
73     return (*this);
74 }
75
76
77 /* Operador producto por escalar */
78 VecR3 operator*( const float &aesc, const VecR3 &avec )
79 {
80     VecR3 tmp;
81     tmp.Xcor = aesc*avec.Xcor;
```



```
82     tmp.Ycor = aesc*avec.Ycor;
83     tmp.Zcor = aesc*avec.Zcor;
84
85     return tmp;
86 }
87
88
89 /* Operador division por escalar */
90 VecR3 operator/( const float &aesc, const VecR3 &avec )
91 {
92     VecR3 tmp;
93     tmp.Xcor = (1/aesc)*avec.Xcor;
94     tmp.Ycor = (1/aesc)*avec.Ycor;
95     tmp.Zcor = (1/aesc)*avec.Zcor;
96
97     return tmp;
98 }
99
100
101
102 /* Despliega un vector con cout */
103 std::ostream &operator<<( std::ostream &salida, const VecR3 &avec )
104 {
105     /* Se decide el tipo de salida en funcion del valor del atributo
106      * de clase Polar. */
107     if( VecR3::Polar )
108     {
109         /* Se calcula el angulo polar del vector. La magnitud
110          * se obtiene del metodo ya implementado */
111         float ang = std::atan2( avec.Ycor , avec.Xcor );
112         salida << "(" << avec.Magnitud() << "<< " << ang << ")";
113     }
114     else
115         salida << "(" << avec.Xcor << ", " << avec.Ycor << ", " << avec
116             .Zcor << ")";
117
118     return salida;
119 }
```

Con estos operadores definidos, la salida probandolos es la siguiente:



```
diego@diego ~/.../Codigos_clase/U1-P2 > g++ VecR3.cpp t1II3.cpp
diego@diego ~/.../Codigos_clase/U1-P2 > ./a.out
Vectores Utilizados
( 1, 0, 1 )
( 0, 1, 0 )
Magnitud v1: 1.41421
v3 = v1 + v2 = ( 1, 1, 1 )
v4 = v1 + v2 = ( 1, -1, 1 )
c5 = v1 * v2 = 0
(Asignar) v6 = v4 + v3 - v1 = ( 1, 0, 1 )
v7 = 5*v4 = ( 5, -5, 5 )
v8 = 5/v4 = ( 0.2, -0.2, 0.2 )
diego@diego ~/.../Codigos_clase/U1-P2 >
```

Figura 1: Salida del programa representando los resultados al utilizar los distintos operadores.