Esercizi sui dangling pointer

Il seguente materiale è da consultare e studiare con una buona dose di spirito critico.
Simone Magagna

```
main()
{
   const int x=99, *p=&x;
   int* q=const_cast<int*>(p);
   (*q)++; cout<<x<<' '<<*q<<' '<<*p<<endl;
}
```

Stampa: 99 100 100.
Questo avviene poichè il compilatore sostituisce ogni costante col valore assegnatole quanto è stata creata, senza controllare quale valore ha realmente la variabile.

```
int ** f(int *&p){int **x=&p; p[0]--; p++; return x;}

main(){int b[]={2,3,4,5}, *q=b+1; **f(q)=*q; cout<<b[0]<<' '<<b[1]<<' '<<b[2]<<' '<<b[3];}
```

Non vi è dangling pointer e stampa: 2 2 4 5.

```
int* f(int *&p){int b=3, *x=&b; x=p+1; p++; return x-2;}

main(){int b[]={1,2,3,4}, *q=b+2; *f(q)=*q; cout<<b[0]<<b[1]<<b[2]<<b[3];}
```

Non vi è dangling pointer e stampa: 1 4 3 4.

```
int* f(int** p){int b=3, *x=&b; *p=x; x=*p; return x+1;}

main(){int b[]={2,3}, *q=b; *f(&q)=*q; cout<<b[0]<<b[1]<<*q;}
```

Vi è un dangling pointer poichè la funzione restituisce un puntatore ad una cella successiva ad una variabile
locale della funzione.

```
int** f(int * & p){int**x=&p; (*((*x)+1))++ ; p--; return x; }
main() {int b[]={2,3,4,5},*q=b+1; **f(q)=*q; cout<<b[0]<<b[1]<<b[2]<<b[3];}
```

Non viè dangling pointer e stampa: 2355

```
int*& f(int * & p){int*& x=p; ++x; ++p; return x; }
```

main() {int b[]={2,3,4,5},*q=b; f(q)=b; cout<<*q<<b[0]<<b[1]<<b[2]<<b[3];}

Non vi è dangling pointer e stampa 22354

int ** F(int** p){(*p)++; return p;}
main(){int a[]={0,1,2,3}, *q=&(*a); **F(&q)=a[3]+1; cout<<q[0]<<q[1]<<q[2]<<a[3];}

Non vi è dangling pointer e stampa 4233