## esercizi su funzioni:

- passaggio dei parametri
- restituzione dei risultati
- invocazione

# esempio 0:

```
int * f(int * x){*x=5; return x;}
main()
{ int y=1;
*f(&y)=25;
cout<<y<<endl;
}
```

è corretto e stampa 25

*f* riceve e ritorna un puntatore a *y*

*y* riceve 5 in *f* e poi 25

# esempio 1 :

```
int * f(int  x){x=5; return &x;}
main()
{ int y=1;
*f(y)=7*y;
cout<<y<<endl;}
```

ERRORE LOGICO, ma no warning, *f* ritorna un puntatore a x (var. locale che viene deallocata) quindi *f(y)=7*y ha effetto su x (ORRORE!) e non su y che rimane 1.

# esempio 2:

```
int & f(int & x){x=5; return x;}

main()
{ int y=1;
*f(y)=25;
cout<<y<<endl;}
```

Errore:

In function ‘int  main’ : invalid type argument of ‘unary *’

# esempio 3:

```
int * f(int & x){x=5; return &x;}
main() { int y=1;
*f(y)=25;
cout<<y<<endl;
}
```

è corretto e stampa 25

*x* è un alias di *y* ed *f* ritorna un puntatore a *y*

quindi *y* diventa 5 in *f* e poi 25

# esempio 4:

```
int & f(int & x){x=5; return &x;}

main() { int y=1;
*f(y)=25;

cout<<y<<endl;
}
```

ERROR: In function ‘int main()’: invalid type argument of ‘unary *’

# esempio 5:

```
int * *f(int * x){*x=5; return &x;}
main() { int y=1;
**f(&y)=25;
cout<<y<<endl;
}
```

ERRORE LOGICO, ma no warning,

# esempio 6:

```
int * f(int * & x){*x=5; return x;}
main()
{ int y=1, x;
x=*f(&y);
cout<<y<<“ “<<x<<endl;}
```

In function ‘int main()’: inizialization of non-const reference ‘int *&’ from r-value ‘int *’ in passing argument 1 of ‘f(int * &)’ .

# esempio 7:

```
int & f(int * & x){*x=5; return *x;}
main() { int y=1, x; int *z=&y;
x= f(z);
cout<<y<<“ “<<x<<endl;    }
```

CORRETTA: stampa 5  5

*f* ritorna un alias di *y* ed *x* è un alias di *z*

# esempio 8:

```
int & f(int * & x){*x=5; return *x;}
main() { int y=1, *z=&y;
f(z)=25;
cout<<y<<endl;
}
```

CORRETTA: stampa 25

come nell'esempio precedente, *f* ritorna un alias di *y*, quindi *y* riceve 25

....f(int &x)

int a=1, *p=&a;
....f(*p)..... // ok

1) Si consideri il seguente programma e si dica se è corretto oppure no spiegando in modo preciso le ragioni della risposta. Si consiglia di usare un grafico che mostri le relazioni tra le diverse variabili.

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

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

stesso esercizio:

```
int * & F(int** & p){int*x=(*p)+2;
*p=x+1; return *p;}

main(){ int X[5]={0,1,2,3,4}, *q=X+1,
**p=&q; F(p)=q-2;
cout<<*q<<**p<<endl;}
```

ancora

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

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

e ancora

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

main()
 {int y=5, b=2,*q=&b;
*f(&q)=y*2; cout<<y<<b<<*q;
 }
```

per finire

```
int *f(int **p){int b=3,*x=&b; **p=b; x=*p; return x;
}
main() {int y=5, b=2,*q=&b; *f(&q)=y*2;}
```

# esempio !

```
int * f(int * x){*x=5; return x;}
main() { int y=1;
*f(&y)=25*y;
cout<<y<<endl;
}
```

è corretto, ma ha un problema dovuto al side-effect di f

esercizio:

```
int k=5, *z=&k;

*f(&z)=k+5;

cout<<*z <<endl;
```

vogliamo una f() t. c. stampi 10

fatelo!

# Domande a risposta multipla 1

```
int F(int *x){*x = 10; return *x;}
main(){
        int x =1, y=1, *p = &x, &q = y;
        y = F(&q) + 1;
        cout << x <<y<<endl;}
```

1. La compilazione da un errore di tipo
2. Il programma è corretto e stampa 1 11
3. Il programma è corretto e stampa 10 11

# Domande a risposta multipla 2

```
int F(int *x){*x = 10; return *x;}
main(){
        int x =1, y=1, *p = &x, * &q = &y;
        y = F(p);
        cout << x <<y<<endl;
}
```

1. La compilazione da un errore di tipo
2. Il programma è corretto e stampa 10 10

# Domande a risposta multipla 3

```
main(){
        int y=1, *ptr = &y, *&q = ptr;
        *ptr = 2; cout << y;
        *q = 3; cout << y <<endl;
}
```

1. Il programma è corretto e stampa 1 1
2. La compilazione da un errore di tipo
3. Il programma è corretto e stampa 2 3

# Domande a risposta multipla 4

```
char * C(char x, char &y) {x='b'; y=x; return &x;}

main(){
        char A[] = {'a','b','c'}, *p;
        p=C(A[2],A[0]);
        cout << A[0] <<endl; }
```

1. Il programma è sbagliato
2. Il programma è corretto e stampa b
3. Il programma è corretto e stampa a

# Domande a risposta multipla 5

```
char & C(char &x, char &y) {y=x; return x;}

main(){
        char A[] = {‘a’,’b’,’c’};
        C(A[0],A[1]) = A[2];
        cout << A[0] << A[1] << A[2] <<endl; }
```

1. Il programma è sbagliato
2. Il programma è corretto e stampa c a c
3. Il programma è corretto e stampa b b c

# Domande a risposta multipla 6

```
char  C(char x, char &y) {x=y; return x;}

main(){

        char A[] = {‘a’,’b’,’c’};

        A[2] = C(A[0],A[1]) ;

        cout << A[0] << A[1] << A[2] <<endl; }
```

1. Il programma è sbagliato
2. Il programma è corretto e stampa a b b
3. Il programma è corretto e stampa b b b