20-09-2005

```
#include <string>

class D;

class C {
private:
  // ...
public:
  C() {}
  C(const D&) {}
  // ...
};

class D {
private:
  // ...
public:
  D() {}
  D(const C&) {}
  // ...
};


void f(double d);                       // funzione #1
void f(int i, C c = C());               // funzione #2
void f(double d, C c = C());            // funzione #3

void g(C c, D d);                       // funzione #4
void g(D d, C c = C());                 // funzione #5

void h(const char* s);                  // funzione #6
void h(const std::string& s);           // funzione #7


int main() {
  C c;
  D d;

  f('a');          // chiamata #1
  f('a', c);       // chiamata #2
  f(3.2);          // chiamata #3

  g(c, d);         // chiamata #4
  g(d, c);         // chiamata #5
  g(c);            // chiamata #6
  g(d);            // chiamata #7

  h("abra");       // chiamata #8
  h('a');          // chiamata #9
}
```

Chiamata	Candidate	Utilizzabili	Migliore
1	1 2 3	2	2
2	2 3	2	2
3	1 2 3	1 3	1
4	4 5	4	4
5	4 5	5	5
6	X	X	X
7	5	5	5
8	6 7	6 7	6
9	6 7	X	X

06-02-2006

```
#include <iostream>

class Base {
public:
  void f(int, double);               // funzione #1
  void f(double, int) const;         // funzione #2
  void g(double);                    // funzione #3
  void print(std::ostream&) const;   // funzione #4
};

class Derived : public Base {
public:
  using Base::f;
  void f(double, double);            // funzione #5
  void g(double) const;              // funzione #6
private:
  void print(std::ostream&);         // funzione #7
};

int main() {
  Base b;
  Derived d;
  Base* pb = &d;
  const Derived* pd = &d;

  b.print(std::cout);       // chiamata (a)
  d.print(std::cerr);       // chiamata (b)
  pb->print(std::cerr);     // chiamata (c)
  pd->print(std::cout);     // chiamata (d)

  b.f('a', 0.7);            // chiamata (e)
  d.f(12.5, 1.4);           // chiamata (f)
  pb->f(2, 0);              // chiamata (g)
  pd->f(7.2, 7);            // chiamata (h)
  pd->f(7, 7.2);            // chiamata (i)

  const Base* pb2 = static_cast<const Base*>(pd);
  pb2->g(0.0);              // chiamata (j)
  pd->g(0.0);               // chiamata (k)
}
```

Chiamata	Candidate	Utilizzabili	Migliore
a	4	4	4
b	4 7	X	X
c	4	4	4
d	4 7	4	4
e	1 2	1	1
f	1 2 5	1 2 5	5
g	1 2	1 2	ambiguo
h	1 2 5	2	2
i	1 2 5	2	2
j	3	3	3
k	3 6	6	6

1-02-2005

```
namespace NB {
  class D {};
} // namespace NB

namespace NA {
  class C {};
  void f(int i);                          // funzione #1
  void f(double d, C c = C());            // funzione #2
  void g(C c = C(), NB::D d = NB::D());   // funzione #3
  void h(C c);                            // funzione #4

  void test1() {
    f(2.0);            // chiamata #1
  }
} // namespace NA

namespace NB {
  void f(double d);                       // funzione #5
  void g(NA::C c = NA::C(), D d = D());   // funzione #6
  void h(NA::C c, D d);                   // funzione #7

  void test2(double d, NA::C c) {
    f(d);              // chiamata #2
    g(c);              // chiamata #3
    h(c);              // chiamata #4
  }
} // namespace NB

void f(NA::C c, NB::D d);                 // funzione #8

void test3(NA::C c, NB::D d) {
  f(1.0);            // chiamata #5
  g();               // chiamata #6
  g(c);              // chiamata #7
  g(c, d);           // chiamata #8
}
```

Chiamata	Candidate	Utilizzabili	Migliore
1	1 2	1 2	2
2	5	5	5
3	3 6	3 6	Ambiguo
4	4 7	4	4
5	8	X	X
6	X	X	X
7	3	3	3
8	6 3	6 3	6 3

16-06-2008

```
#include <string>

namespace N {

  class C {
  private:
    std::string& first();                    // funzione #1

  public:
    const std::string& first() const; // funzione #2

    std::string& last();                     // funzione #3
    const std::string& last() const;  // funzione #4

    C(const char*);                          // funzione #5
  }; // class C

  void print(const C&);                      // funzione #6
  std::string& f(int);                       // funzione #7

} // namespace N

class A {
public:
  explicit A(std::string&);                  // funzione #8
};

void print(const A&);                        // funzione #9

void f(N::C& c)                              // funzione #10
{
  const std::string& f1 = c.first();  // chiamata #1
  std::string& f2 = c.first();        // chiamata #2
  const std::string& l1 = c.last();   // chiamata #3
  std::string& l2 = c.last();         // chiamata #4
}

void f(const N::C& c)                        // funzione #11
{
  const std::string& f1 = c.first();  // chiamata #5
  std::string& f2 = c.first();        // chiamata #6
  const std::string& l1 = c.last();   // chiamata #7
  std::string& l2 = c.last();         // chiamata #8
}

int main() {
  N::C c("begin");    // chiamata #9
  f(c);               // chiamata #10
  f("middle");        // chiamata #11
  print("end");       // chiamata #12
}
```

Chiamata	Candidate	Utilizzabili	Migliore
1	1 2	2	2
2	1	X	X
3	3 4	3	3Qui viene chiamata la funzione 3 perché anche se l1 è un riferimento costante ad una stringa, c (argomento della funzione) non lo è, quindi viene invocata la funzione 3
4	3 4	3	3
5	1 2	2	2 Qui c (argomento della funzione) è un riferimento costante, quindi è possibile invocare la funzione 2 marcata const
6	1 2	X	X Il ragionamento è lo stesso della chiamata 5 però viene restituita un riferimento costante a stringa che non può essere convertito in riferimento non costante
7	3 4	4	4
8	3 4	X	X Stesso ragionamento della chiamata 6
9	9	5	5
10	7 10 11	10 11	10 Qui viene chiamata la 10 perché la c creata alla chiamata 9 non è costante, quindi viene preferita la funzione senza qualificazione
11	10 11	X	X Non è presente nessuna funzione utilizzabile perché le candidate aspettano come argomento un oggetto di tipo C
12	9	X	X

14-06-2010

```
namespace N {
  struct C {
    C(int);       // funzione #1
    C(const C&);  // funzione #2

    void m();     // funzione #3
    void m(int);  // funzione #4
  };

  void f(double d);        // funzione #5
  void f(const C& c);      // funzione #6
  void g(int i, double d); // funzione #7
  void g(int i, int j);    // funzione #8
  void h(C* pc);           // funzione #9

} // namespace N


void f(char); // funzione 10

int h(const char* s = 0); // funzione 11
int h(const N::C* pc);    // funzione 12


int main() {
  N::C c(5);    // chiamata A

  f(5);         // chiamata B
  f(c);         // chiamata C
  N::f('a');    // chiamata D

  g(5, 3.7);    // chiamata E
  N::g(2.3, 5); // chiamata F
  N::g(5, 2.3); // chiamata G

  h(&c);        // chiamata H
  h();          // chiamata I

  m(&c);        // chiamata J
}
```

Chiamata	Candidate	Utilizzabili	Migliore
A	1 2	1 2	1
B	10	10	10
C	5 6 10	5 6	6
D	5 6	5	5
E	X	X	X
F	7 8	7 8	8
G	7 8	7 8	7
H	9 11 12	9 12	9
I	11 12	11	11
J	3 4	X	X

22-02-2005

```
void f(const char* s);    // funzione #1
template <typename T>
void f(T t);              // funzione #2

template <typename T>
void f(T t1, T t2);       // funzione #3
template <typename T, typename U>
void f(T t, U u);         // funzione #4
template <typename T>
void f(T* pt, T t);       // funzione #5
template <typename T, typename U>
void f(T* pt, U u);       // funzione #6

template <typename T>
void g(T t, double d);    // funzione #7
template <typename T>
void g(T t1, T t2);       // funzione #8

int test() {
  f('a');          // chiamata #1
  f("aaa");        // chiamata #2
  int i;
  f(i);            // chiamata #3
  f(i, i);         // chiamata #4
  f(i, &i);        // chiamata #5
  f(&i, i);        // chiamata #6
  double d;
  f(i, d);         // chiamata #7
  f(&d, i);        // chiamata #8
  long l;
  g(l, i);         // chiamata #9
  g(l, l) ;        // chiamata #10
  g(l, d) ;        // chiamata #11
  g(d, d) ;        // chiamata #12
}
```

Chiamata	Funzione Invocata
1	2
2	1
3	2
4	3
5	4
6	5
7	4
8	6
9	7(promozione)
10	8
11	7
12	Ambiguo

27-02-2006

```
void foo(int*);                // Funzione #1
template <typename T>
void foo(const T&);            // Funzione #2

template <typename T>
void foo(T, T);                // Funzione #3
template <typename T, typename U>
void foo(T, U)                 // Funzione #4
template <typename T>
void foo(const T*, T);         // Funzione #5
template <typename T, typename U>
void foo(T*, U);               // Funzione #6

namespace A {

struct Delta {
  Delta(double = 0) {};
};

template <typename T>
void bar(T, double);          // Funzione #7
template <typename T, typename U>
void bar(T, U);               // Funzione #8

} // namespace A

template <typename T>
void bar(T, const A::Delta&); // Funzione #9

int main() {
  int alfa;
  const int* p_alfa = &alfa;
  double beta;
  long gamma;
  A::Delta delta;

  foo(&alfa, beta);             // chiamata A
  foo(&alfa, alfa);             // chiamata B
  foo(p_alfa, alfa);            // chiamata C
  foo(alfa, p_alfa);            // chiamata D

  foo(alfa);                    // chiamata E
  foo(&alfa);                   // chiamata F
  foo(p_alfa);                  // chiamata G
  foo(const_cast<int*>(p_alfa)); // chiamata H

  bar(gamma, gamma);            // chiamata I
  bar(delta, alfa);             // chiamata L
  bar(delta, beta);             // chiamata M
  A::bar(beta, gamma);          // chiamata N
}
```

Chiamata	Candidate	Utilizzabili	Migliore
A	4 5 6	4 6	6
B	4 5 6	4 6	6
C	4 5	4 5	5
D	4	4	4
E	1 2	2	2
F	1 2	1	1
G	1 2	2	2
H(toglie il cast)	1	1	1
I(long->double)	9	9	9
L	7 8 9	7 8 9	8
M	7 8 9	7 8 9	7
N	7 8	8	8