Il programma C++ più semplice

```
int main() {
    return 0;
}
```

Commenti in C++

```
void f() {
  int a = 1, b = 2, c = 3;

  a = b + c; // This part is a comment.


/* This is a comment. */

}
```

C++ type	*Size (bytes)*	*FORTRAN analog*
char	1	CHARACTER*1
short int	2	INTEGER*2
int	2 or 4	INTEGER*2 or INTEGER*4
long int	4 or 8	INTEGER*4 or INTEGER*8
float	4	REAL*4
double	8	REAL*8
long double	16	REAL*16

Table 2.1 Numeric Types and Typical Sizes. The actual size of each object is implementation dependent. Usually, int is the natural machine word size and either short or long int will be the same size as int. The type long int can be written long, and short int can be written short.

Ingresso / uscita in C++

```
#include <iostream.h>

int main() {
    // Read and print three floating point numbers
    float a, b, c;
    cin >> a >> b >> c;
    cout << a << ", " << b << ", " << c << endl;

    return 0;
}
```

C++	*Purpose*	*FORTRAN*
`x ++`	Postincrement	
`++ x`	Preincrement	
`x --`	Postdecrement	
`--x`	Predecrement	
`+ x`	Unary plus	`+ X`
`- x`	Unary minus	`- X`
`x * y`	Multiply	`X * Y`
`x / y`	Divide	`X / Y`
`x % y`	Modulus	`MOD (X,Y)`
`x + y`	Add	`X + Y`
`x - y`	Subtract	`X - Y`
`pow(x,y)`	Exponentiation	`X ** Y`

Table 2.2 Arithmetic Operators

```
Un programma per calcolare le coordinate x e y dell'intersezione di una retta
#include <iostream.h>
int main() {
    // Legge i coefficienti dell'equazione della retta nella forma ax+by+c = 0,
    // e stampa le coordinate delle intersezione con gli assi x e y.

    // Legge i coefficienti.
    float a, b, c;
    cin >> a >> b >> c;

    // Stampa i coefficienti.
    cout << "Coefficienti: " << a << ", " << b << ", " << c << endl;

    // Calcola e stampa l'ascissa x.
    cout << "intersezione-x: ";
    if (a != 0) {
        cout << -c / a << ", ";
    }
    else {
        cout << "none, ";
    }

    // Calcola e stampa l'ordinata y.
    cout << "intersezione-y: ";
    if (b != 0) {
        cout << -c / b << endl;
    }
    else {
        cout << "none" << endl;
    }
    return 0;
}
```

C++	*Purpose*	*FORTRAN*
`x < y`	Less than	`X .LT. Y`
`x <= y`	Less than or equal	`X .LE. Y`
`x > y`	Greater than	`X .GT. Y`
`x >= y`	Greater than or equal	`X .GE. Y`
`x == y`	Equal	`X .EQ. Y`
`x != y`	Not equal	`X .NE. Y`

Table 2.3 Relational Operators

C++	*Purpose*	*FORTRAN*
`0`	False value	`.FALSE.`
nonzero	True value	`.TRUE.`
`! x`	Logical negation	`.NOT.X`
`x && y`	Logical and	`X .AND. Y`
`x || y`	Logical inclusive or	`X .OR. Y`

Table 2.4 Logical Values and Operators

C++ Escape Sequence	*Meaning*
`\n`	Newline
`\t`	Horizontal tab
`\v`	Vertical tab
`\b`	Backspace
`\r`	Carriage return
`\f`	Form feed
`\a`	Alert
`\\`	Backslash
`\?`	Question mark
`\'`	Single quote
`\"`	Double quote
`\`*ddd*	Octal number with 1, 2, or 3 digits, *d*
`\x`*ddd*	Hexadecimal number with any number of digits, *d*

Table 2.5 Character Escape Sequences

C++	*FORTRAN*
while (*expr*) {	10 IF (.NOT. *expr*) GOTO 20
...	...
	GOTO 10
}	20 CONTINUE
do {	10 CONTINUE
...	...
} while (*expr*);	IF (*expr*) GOTO 10
	init-stmt
	10 IF (.NOT.(*cont-expr*)) GOTO 20
for (*init-stmt; cont-expr; incr-expr*) {	...
...	*incr-expr*
}	GOTO 10
	20 CONTINUE
for (i=1; i <=j; i += k) {	DO 10 I=1, J, K
...	...
}	10 CONTINUE

Table 2.8 Summary of C++ and FORTRAN Loops. These are equivalent, but note that C++ arrays have index 0 as the first element and therefore most C++ loops begin with element 0.

```
IF

if (x < 0) x = -x;        // x = abs(x)

IF-ELSE

if (current_temp > maximum_safe_temp) {
    // Emergency cool down.
    cerr << "EMERGENCY: Too hot--flushing" << endl;
    flushWithWater();
}
else {
    // Normal control strategy.
    if (current_temp > operating_temp + temp_tolerance) {
        heaterOff();
        if (current_temp > operating_temp + 2*temp_tolerance) {
            coolingWaterOn();
        }
    }

    if (current_temp < operating_temp - temp_tolerance) {
        coolingWaterOff();
        if (current_temp < operating_temp - 2*temp_tolerance) {
            heaterOn();
        }
    }
}
```

Una tabella della radice quadrata

```
#include <iostream.h>
#include <iomanip.h>
#include <math.h>

int main() {

float x;
while (cin >> x) {
    cout << setw(25) << x << setw(25) << sqrt(x) << endl;
}

return 0;
}
```

Calcolo dello zero di una funzione col metodo di Newton

```
#include <iostream.h>
#include <math.h>
static double f(double x) { return x*x-2; }
static double fprime(double x) { return 2*x; }
int main() {
double x, dx;
const double initial_guess = 10;
const double desired_accuracy = 1e-5;

x = initial_guess;
do {

    dx = f(x) / fprime(x);
    x  -= dx;

} while (fabs(dx) > desired_accuracy);

cout << x << endl;
return 0;
}
```