



# Laurea triennale in Informatica

*modulo (CFU 6) di*

## Programmazione II e Lab.

**prof. Mariarosaria Rizzardi**

Centro Direzionale di Napoli – Isola C4

stanza: n. 423 – IV piano Lato Nord

tel.: 081 547 6545

email: [mariarosaria.rizzardi@uniparthenope.it](mailto:mariarosaria.rizzardi@uniparthenope.it)

- 
- The background features a large, faint watermark of the University of Naples Federico II seal. The seal is circular with a blue border. Inside the border, the text "1920 - 2020" is at the top, "DEGLI STUDI" is on the left, "NAPOLI" is on the right, and "100° ANNIVERSARIO" is at the bottom. In the center of the seal is a figure of a person, likely a saint or scholar, holding a book and a staff.
- **C e C++: compilare da riga di comando**
  - **Passaggio dei parametri: per reference in C++**

# Compilare da riga di comando (Windows)

## 1. Installare MinGW

**Scaricare** mingw-w64-install.exe **da:**

<https://sourceforge.net/projects/mingw-w64/files/latest/download?source=files>

ultima versione della libreria **MinGW** (Minimalist GNU for Windows),  
porting in ambiente Windows del famoso compilatore GCC per Linux.

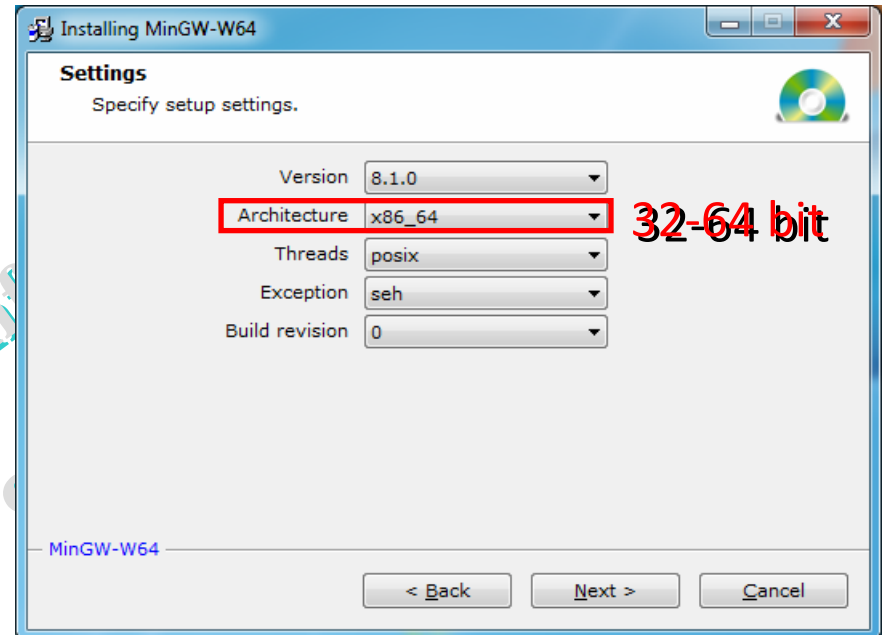
**Eseguire** mingw-w64-install.exe **con accesso a Internet:**



# 1. Installare MinGW

Scegliere l'ultima versione:

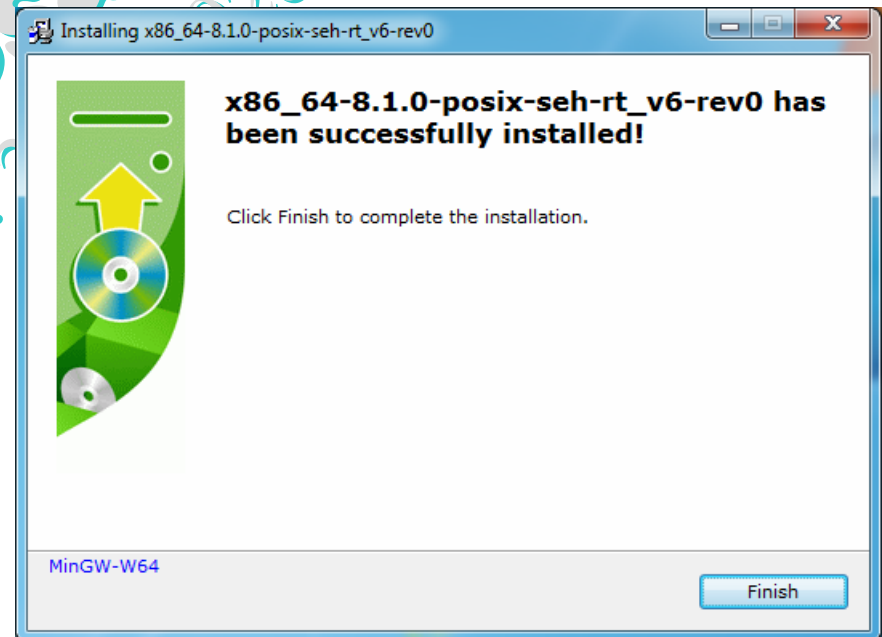
Scegliere 32 o 64 bit:



32-64 bit



dove installare la libreria del compilatore



# 1. Installare MinGW

Per avere un “launcher” sul Desktop, copiare (dalla directory di installazione) mingw-w64.bat sul Desktop ed eventualmente rinominarlo:

```
echo off
```

mingw-w64.bat

```
set PATH=Directory di installazione\mingw64\bin;%PATH%
```

```
rem echo %PATH%
```

```
rem cd "Directory di installazione\mingw64\bin"
```

```
cd "C:\"
```

```
"C:\Windows\system32\cmd.exe"
```

apre un terminale

aggiunge temporaneamente la cartella

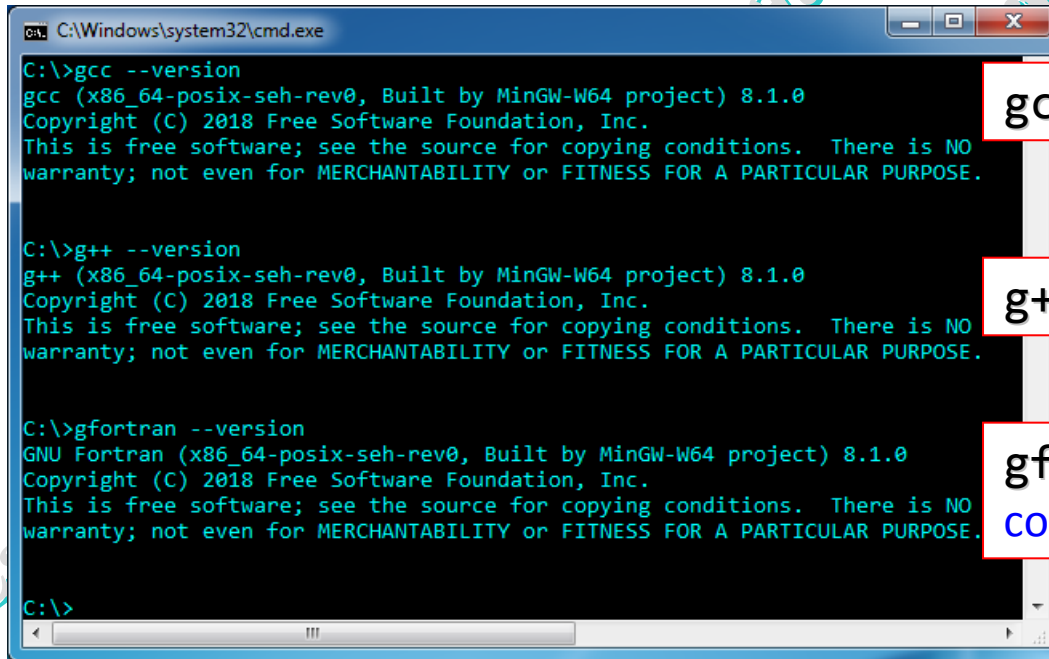
Directory di installazione\mingw64\bin

alla variabile di sistema PATH

In tal modo non viene aggiunto permanentemente il path della libreria del compilatore alla variabile di ambiente PATH e quando si chiude la finestra DOS il sistema ritorna allo stato precedente.

## 2. Compilare da riga di comando (Windows)

Due click su `mingw-w64.bat` apre una finestra dei comandi DOS:



```
C:\Windows\system32\cmd.exe
C:\>gcc --version
gcc (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

C:\>g++ --version
g++ (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

C:\>gfortran --version
GNU Fortran (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0
Copyright (C) 2018 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

C:\>
```

gcc: compilatore C

g++: compilatore C++

gfortran:  
compilatore

\* se si è scaricato MinGW con  
il compilatore FORTRAN\*

### Principali comandi DOS (Windows):

`cls`: pulisce la finestra

`dir`: visualizza il contenuto della cartella corrente

`G:`: cambia unità corrente in G:

`cd "nuova cartella"`: cambia la cartella corrente

`del "nome file"`: cancella il file (o i file)

### ... Linux:

`man cd`: manuale di `cd`

`clear`: pulisce la finestra

`ls -la`: lista directory

`cd "G:nuova cartella"`:

`rm "nome file"`: cancella



# Confronto /MinGW/bin con MinGW di Code::Blocks v.17.12

## Code::Blocks

```
C:\Windows\system32\cmd.exe
G:\My_Programs\My_ProgramFiles_x86\CodeBlocks\MinGW\bin>dir *.dll
Il volume nell'unità G è Volume
Numero di serie del volume: 4A03-6127

Directory di G:\My_Programs\My_ProgramFiles_x86\CodeBlocks\MinGW\bin
08/11/2001  02:27          237.568 glut32.dll
27/06/2015  07:11          61.454 libatomic-1.dll
01/09/2013  03:38          149.207 libcharset-1.dll
27/06/2015  23:50          145.934 libgcc_s_dw2-1.dll
27/06/2015  07:11          124.430 libgcc_s_sjlj-1.dll
27/06/2015  07:11         1.038.350 libgfortran-3.dll
27/06/2015  07:11          921.614 libiconv-2.dll
27/04/2014  00:46          484.613 libintl-8.dll
27/06/2015  07:11          513.038 libquadmath-0.dll
27/06/2015  07:11           35.854 libssp-0.dll
27/06/2015  07:11         1.488.910 libstdc++-6.dll
27/06/2015  07:11           9.230 libvtv-0.dll
27/06/2015  07:11           9.230 libvtv_stubs-0.dll
27/06/2015  07:11           58.880 libwinpthread-1.dll
29/06/2012  23:52           22.086 mingwm10.dll
             15 File          5.300.398 byte
             0 Directory 1.425.604.038.656 byte disponibili
```

`dir *.dll`

glut32.dll: interfaccia per le finestre grafiche tra OpenGL e il window manager di MS Windows

libwinpthread-1.dll: libreria per il calcolo parallelo con pthread

`dir *.dll`

libgomp-1.dll: libreria per il calcolo parallelo con OpenMP

libwinpthread-1.dll: libreria per il calcolo parallelo con pthread

## mingw-w64

```
C:\Windows\system32\cmd.exe
>dir *.dll
Il volume nell'unità G è Volume
Numero di serie del volume: 4A03-6127

Directory di G:\My_Downloads\Software\MinGW-64\SourceForge_net\gcc_8-1
12/05/2018  08:11          32.768 libatomic-1.dll
12/05/2018  08:11          76.288 libgcc_s_seh-1.dll
12/05/2018  08:11          156.672 libgomp-1.dll
12/05/2018  08:11          334.848 libquadmath-0.dll
12/05/2018  08:11           17.408 libssp-0.dll
12/05/2018  08:11         1.417.216 libstdc++-6.dll
12/05/2018  08:11          52.224 libwinpthread-1.dll
             7 File          2.087.424 byte
             0 Directory 1.425.604.038.656 byte disponibili

G:\My_Downloads\Software\MinGW-64\SourceForge_net\gcc_8-1-0_2019_08_06\
```

# Confronto /MinGW/bin con MinGW di Code::Blocks v.17.12

## Code::Blocks

```
Directory di installazione\CodeBlocks\MinGW\bin>gcc --version
```

```
gcc (tdm-1) 5.1.0
```

```
Copyright (C) 2015 Free Software Foundation, Inc.
```

```
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

più vecchio

più nuovo

mingw-w64

```
C:\>gcc --version
```

```
gcc (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0
```

```
Copyright (C) 2018 Free Software Foundation, Inc.
```

```
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

Come individuare se si usa la versione a 32 o 64 bit della libreria?

```
C:\>gcc -dumpmachine
```

```
x86_64-w64-mingw32
```

mingw-w64



# Come individuare se la libreria usa indirizzi di memoria a 32 o 64 bit?

```
#include <stdio.h>
int main()
{ printf("\nsizeof(char)           = %d\n", sizeof(char));
  printf("sizeof(short)           = %d\n", sizeof(short));
  printf("sizeof(int)              = %d\n", sizeof(int));
  printf("sizeof(long int)          = %d\n", sizeof(long));
  printf("sizeof(long long int)     = %d\n", sizeof(long long));
  printf("sizeof(char*)            = %d\n", sizeof(char*));
  return 0;
}
```

main.c

&gt;main.exe

Code::Blocks

```
sizeof(char)           = 1
sizeof(short)          = 2
sizeof(int)            = 4
sizeof(long int)       = 4
sizeof(long long int)  = 8
sizeof(char*)          = 4
```

indirizzi di memoria a 32 bit

&gt;main.exe

mingw-w64

```
sizeof(char)           = 1
sizeof(short)          = 2
sizeof(int)            = 4
sizeof(long int)       = 8
sizeof(long long int)  = 8
sizeof(char*)          = 8
```

indirizzi di memoria a 64 bit

## 2. Compilare da riga di comando (Windows)

Selezionare la directory del programma sorgente:

```
C:>G:      cambiare unità  
G:>cd "mydir\mysrc"
```

```
G:\mydir\mysrc>gcc -c main.c  
G:\mydir\mysrc>gcc -o main.exe *.o  
G:\mydir\mysrc>main  
...
```

produce gli object file

(.o)

Compile sources.  
Link object files.  
Run executable.

produce l'eseguibile

Mettendo insieme in un'unica riga (produce solo l'eseguibile):

```
G:\mydir\mysrc>gcc -o main.exe main.c
```

## 2. Compilare da riga di comando (Windows)

Creare i sorgenti C (per es. in G:\mydir\mysrc\):

```
/* main.c */
#include <stdio.h>
#include "sommaArray.h"

int main(int argc, char *argv[])
{
    int Nv=10;
    float somma, v[]={0.0f,1.0f,2.0f,3.0f,4.0f,
                      5.0f,6.0f,7.0f,8.0f,9.0f};

    somma = sum(v,Nv);
    printf("\nsomma di array = %g\n", somma);
    return 0;
}
```

```
/* sommaArray.c */
float sum(float a[], int N)
{
    float s=0;
    for (int k=0; k<N; k++)
        s += a[k];
    return s;
}
```

```
/* sommaArray.h */
float sum(float a[], int N);
```

Selezionare la directory dei sorgenti:

```
C:>G:
G:>cd "mydir\mysrc"
```

```
G:\mydir\mysrc>gcc -c main.c sommaArray.c
G:\mydir\mysrc>gcc -o main.exe *.o
G:\mydir\mysrc>main
somma di array = 45
```

Compile sources.  
Link object files.  
Run executable.

Mettendo insieme in un'unica riga (produce solo l'eseguibile):

```
G:\mydir\mysrc>gcc -o main.exe main.c sommaArray.c
```

# Compilare con make (Windows)

Duplicare

*Directory di installazione* \mingw64\bin\mingw32-make.exe  
e rinominare la copia make.exe

Creare il file di nome Makefile

\t (tab)

```
# Makefile for GNU gcc compiler
build:
    gcc -c main.c sommaArray.c
    gcc -o main.exe *.o
```

target

rule

e

Opzioni di GNU gcc:

<https://gcc.gnu.org/onlinedocs/gcc/Option-Summary.html>

Creare l'eseguibile con make:

```
G:\mydir\mysrc>make
G:\mydir\mysrc>main
somma di array = 45
```

Opzioni di make:

<https://linux.die.net/man/1/make>

[https://www.gnu.org/software/make/manual/html\\_node/Options-Summary.html](https://www.gnu.org/software/make/manual/html_node/Options-Summary.html)

# Compilare con make (Windows)

Se i sorgenti si trovano in una cartella diversa (mydir/mysrc) da quella dove c'è il Makefile:

Creare il file Makefile in mydir:

target

\t (tab)

```
# simple Makefile for GNU gcc compiler
```

```
build:
```

```
gcc -c ./mysrc/main.c ./mysrc/sommaArray.c
```

```
gcc -o ./main.exe ./*.o
```

rule

O meglio, creare il file Makefile in mydir:

variabil

e

```
# simple Makefile for GNU gcc compiler
```

```
SRC = ./mysrc/ directory dei sorgenti
```

```
build:
```

```
gcc -c $(SRC)main.c $(SRC)sommaArray.c
```

```
gcc -o ./main.exe ./*.o
```

Creare l'eseguibile con make:

```
G:\mydir> make
```

```
G:\mydir> main
```

```
somma di array = 45
```

# Makefile: aggiungere la clean: rule

Creare il file Makefile in mydir:

build: rule

clean: rule

```
# Makefile for GNU gcc compiler
```

```
SRC = ./mysrc/
```

```
build:
```

```
gcc -c $(SRC)main.c $(SRC)sommaArray.c
```

```
gcc -o ./main.exe ./*.o
```

```
clean:
```

```
del *.exe
```

```
del *.o
```

rule

 **del: vale solo per Windows!**

```
G:\mydir>make
```

oppure make build

```
G:\mydir>main
```

```
somma di array = 45
```

```
G:\mydir>make clean
```

cancella i file .o e l'eseguibile



# stesso Makefile per Windows e Linux

## clean: rule

Creare il file Makefile:

```
# Makefile for GNU gcc compiler (Windows and Linux)
```

```
SRC = ./mysrc/
```

```
build:
```

```
gcc -c $(SRC)main.c $(SRC)sommaArray.c
```

```
gcc -o ./main.exe /*.o
```

```
# Detect operating system
```

```
ifndef OS
```

```
  # Windows
```

```
  RM = del
```

```
else
```

```
  ifeq ($(shell uname), Linux)
```

```
    # Linux
```

```
    RM = /bin/rm -f
```

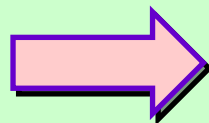
```
  endif
```

```
endif
```

```
clean:
```

```
$(RM) *.o
```

```
$(RM) *.exe
```



Windows

```
del *.o  
del *.exe
```

Linux

```
/bin/rm -f *.o  
/bin/rm -f *.exe
```

P2\_00\_01.16

# Makefile: uso delle dipendenze (Win)

A cosa servono le **dipendenze**?

Se si hanno più file e si modifica uno solo dei sorgenti, con lo stesso Makefile, il comando **make** ricompila solo i sorgenti che "**dipendono**" da quello modificato.

Con molti file sorgenti si ha un risparmio di tempo perché non vengono ricompilati tutti i file, ma solo quelli necessari.

La prima riga di ogni regola (**rule**) definisce un file **target** seguito da ":" e dai file da cui dipende.

La seconda riga (comincia con \t "tab") è il comando da eseguire se va rifatto il build del **target** a causa di una modifica avvenuta su una o più delle sue dipendenze.

# Redirezione dell'input standard (<)

## Creare i sorgenti C:

```
/* main.c */
#include <stdio.h>
#include <stdlib.h>
#include "sommaArray.h" /* stesso di prima */
#include "leggiArray.h"

int main()
{
    int Nv; float somma, *v;
    leggiArray(&Nv, &v);
    somma = sum(v, Nv);
    printf("\nsomma di array = %g\n", somma);
    return 0;
}
```

```
/* leggiArray.c */
#include <stdio.h>
#include <stdlib.h>

void leggiArray(int *N, float **v)
{
    fflush(stdin); scanf("%d", N);
    *v=(float*)malloc(*N*sizeof(float));

    for (int k=0; k<*N; k++)
        scanf("%f,", (*v+k));
}
```

Allocazione dinamica di un array:  
più avanti nel corso!

```
G:\mydir\mysrc>gcc -o main.exe main.c leggiArray.c sommaArray.c
G:\mydir\mysrc>main < dati.txt
somma di array = 45
```

10  
0.0  
1.0  
2.0  
3.0  
4.0  
5.0  
6.0  
7.0  
8.0  
9.0

dati.txt

L'operatore di redirezione dell'input "<" comporta che tutte le `scanf()` prendano l'input non dalla tastiera (`stdin` - standard input) ma dal file specificato.

# Il primo programma con parametri alla funzione main

## Il primo programma in C

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    printf("\nCiao, %s!\n", argv[1]);
    return 0;
}
```

main.c

```
> gcc -o main.exe main.c
```

```
> main Mariarosaria
Ciao, Mariarosaria!
```

## Il primo programma in C++

```
#include <iostream>

int main(int argc, char *argv[])
{
    std::cout << "Ciao, " << argv[1] << "!" << std::endl;
    return 0;
}
```

main.cpp

```
> g++ -o main.exe main.cpp
```

o più semplicemente ...

```
#include <iostream>
using namespace std;

int main(int argc, char *argv[])
{
    cout << "Ciao, " << argv[1] << "!" << endl;
    return 0;
}
```

**std** è il **namespace** della libreria del C++. Il **namespace** definisce un ambito di visibilità: serve per evitare conflitti nei nomi degli identificatori.

# Passaggio dei parametri

In C e C++ il passaggio dei parametri default è "per valore"; si può usare il passaggio "per riferimento" ricorrendo ai puntatori.

**Es.:** programma in C/C++: scambio di due variabili

```
void swap(int*, int*);
int main()
{
    int a=2, b=3;
    swap(&a,&b);
    return 0;
}
```

```
void swap(int* p, int* q)
{
    int tmp=*p;
    *p=*q;
    *q=tmp;
}
```

passaggio dei parametri  
"per riferimento"  
mediante puntatori

**Es.:** programma in C++

```
#include <iostream>
void swap(int&, int&);
using namespace std;
int main()
{
    int a=2, b=3;
    cout << "prima dello scambio: a=" << a << ", b=" << b << endl;
    swap(a,b);
    cout << "dopo lo scambio:      a=" << a << ", b=" << b << endl;
    return 0;
}
```

```
void swap(int& p, int& q)
{
    int tmp=p;
    p=q;
    q=tmp;
}
```

**più semplice!**

In C++  
passaggio dei  
parametri  
"per reference"



# Reference

Il C++, come il C, prevede la dichiarazione di variabili:

- mediante nome: `int v;`
- mediante puntatore: `int* pt; pt=&v;`

In aggiunta il C++ prevede anche la dichiarazione di variabili **reference**: `int& r=v;` da questo momento in poi `r` e `v` rappresentano lo stesso valore. `r++` equivale a `v++`, mentre `pt++` incrementa di 4 byte l'indirizzo di memoria.

Un **reference**, quando dichiarato, deve puntare ad una variabile già dichiarata; quindi la dichiarazione ne prevede anche l'inizializzazione. L'indirizzo cui punta una variabile **reference** non può essere cambiato.

Il tipo **reference** è usato principalmente nel passaggio dei parametri (per riferimento) ad una funzione. I **reference** non sono puntatori.

## Restrizioni:

- Non può esserci un **reference** a una variabile **reference**.
- Non si può creare un array di **reference**.
- Non si può creare un puntatore a un **reference**, cioè l'operatore `&` (indirizzo) non è applicabile ad un **reference**.
- I **reference** non sono consentiti per i campi di bit.

# Documentazione online sul C++ (ISO 11):

<http://www.cplusplus.com/reference/>