1

Capitolo J18
c++, elenchi di riferimento

Contenuti delle sezioni

a. glossario p. 2
b. tipi di dati, modificatori ed estensioni p. 7
c. escape sequences p. 8
d. formati di emissione dei valori dei vari tipi p. 9
e. functions matematiche p. 10
f. algoritmi generici p. 12
g. header files p. 13

14 pagine

J18 0.01 Questo capitolo è dedicato principalmente ad un elenco di termini che riguardano il linguaggio C++ e più in generale la pratica della programmazione e della elaborazione dei dati, gli strumenti dei quali essa si può avvalere e gli ambienti sociali nei quali è utile inquadrarla.

Successivamente sono presentati elenchi di elementi specifici che può essere opportuno visionare nel loro insieme.

In particolare vengono elencate in ordine alfabetico le parole riservate del linguaggio, ciascuna accompagnata da una breve segnalazione del suo ruolo.
Vengono presentati vari termini in inglese, molti dei quali sono di uso gergale; di questi si danno versioni in italiano, talora preferendo una locuzione che esprime il suo significato a una al presente più diffusa.

J18 a. glossario

Elenco alfabetico di termini e stringhe appartenenti al lessico di C++; queste sono nella font `teletype`.

Ogni item dell'elenco ha la forma

`\n`⟨ termine ⟩ := ⟨ breve spiegazione ⟩ :: ⟨ riferimenti ⟩

Tra i termini si privilegiano quelli in inglese, mentre quelli in italiano sono usati preferibilmente nelle brevi spiegazioni.

adaptabilit, adapter = adattabilità, adattatore := 363
address = indirizzo :=
allocation, allocator = allocazione , allocatore := ::
ANSI := ::
append / to = concatenare una stringa a un'altra o un file a un altro
`argc` := numero degli argomenti della intestazione main ::
`argv` := argomento della intestazione main ::
array := ::
ASCII := ::
assign / to := assegnare il valore di un dato-el a un altro
`auto` := ::
`based` := ::
basic data types := ::
binary file := file binario ::
binary operator := operatore binario, a due operandi.
`break` := enunciato per uscire da un blocco.
caret notation := notazione per caratteri di controllo ASCII.
`case` := separatore in un blocco selettivo.
cella-mm := cella di memoria.
cella-kb := cella-mm che richiede k bites, con k = 8, 16, 32, 64, 128, ... ::
cella-kB := cella-mm che richiede k bytes, con k = 1, 2, 4, 8, 16, ... ::
`char` := dichiarazione per variabili simboliche, in un byte ::
char-literal := scrittuta di costante carattere ::
`cin` := console input ::
class := entità costituita da dati-el, metodi e proprietà che diventano operativi per tutti gli oggetti che sono dichiarati sue istanze
`class` := intestazione di un classe di oggetti.
clausola := ::
compile time := durata nella quale viene effettuata la compilazione di un modulo di programma.
compiler := programma che effettua la compilazione dei moduli di programma.
codice sorgente := Sin. testo sorgente
comma separated list := comune elenco di stringhe separate da ",".
`const` := stabilisce che un identificatore introdotto riguarda un valore costante, non modificabile.
constant := costante ::
contenitore ordinato associativo := map, multimap, set, multiset ::
contenitore sequenziale := array, vector, deque, list

`continue` := elemento di blocco iterativo ::
costruttore := ::
`cout` := console output ::
CPU := central processing unit = unità centrale := parte del computer comprendente i circuiti operativi centrali e dispositivi di memoria.
CSI := control sequence introducer ::
cursor = cursore ::
data type := tipo di dati (dati-el) ::
data type modifier = modificatore di tipo di dati := parole riservate da porre prima delle parole chiave di tipi di dati basici al fine di richiedere varianti di tali tipi. Specificamente sono `short, long, signed, unsigned .`
dato-el := dato elaborabile, termine qui usato per denotare una entità singola o multipla, variabile o costante, identificabile e associata a celle-mm, che può essere elaborata nel corso di una esecuzione di un programma.
`default` := ::
`delete` := operatore per la liberazione di memoria dinamica.
deque := coda a doppia entrata: inserimenti e cancellazioni alla fine o all'inizio.
dereference operator := `*` prefisso di indentificatore
derived data types := ::
`do` := ::
`double` := tipo di dati floating in doppia precisione.
driver program := programma con il solo compito di provare altre functions.
`else` := precede un blocco alternativo.
`enum` := dichiarazione di enumerazione.
`envp` := ::
erase / to = cancellare, rimuovere caratteri successivi da una stringa
inheritance = eredità := manovra che consiste nel trasferimento delle proprietà di una class ad una seconda meno generale, evitando al programmatore di richiederlo esplicitamnte e dettagliatamente.
`extern` := ::
`false` := falso, implementato da b00000000 = 0.
file header := ::
`find` := metodo per trovare la prima sottostringa di una stringa procedendo da sinistra o la prima occorrenza in un array , una lista o una deque.
`findr` := metodo per trovare la prima sottostringa di una stringa arretrando da destra.
`float` := :: tipo di dati
floating point number := scrittura di numero in virgola mobile.
`for` := precede iterazione ::
forward declaration := dichiarazione anticipata := ogni function deve essere dichiarata prima di ciscuna delle frasi che la invocano
forward slash := "/".
frammento := porzione di programma con un ruolo tendenzialmente definito.
free store := syn. di hrap
`friend` := ::
function := unità operativa del linguaggio dotata di una certa autonomia, con un identificatore, un tipo e un suo corpo costituente un blocco.
function definition := consiste nella function declaration e nel suo corpo

function arguments = function parameters
function prototype = function declaration := informa il compilatore della function signsture, ma mon del suo comportamento, ossia delle azioni che può effettuare.
function signature := comprende nome, tipo restituito e sequenza dei parametri
funzionalità in un file := possono essere dati-el, functions e classes.
generic algorithm := algoritmo applicabile a strutture diverse
`goto` := comando di salto per il controllo.
GUI := graphical user interface, .. ::
heap := area di memoria gestita dinamicamente dal programmatore, nella quale può richiedere la creazione di nuovi dati-el.
header / file :=
IDE := integrated development environment ::
identifier = identificatore := nome di un dato-el trattato da un programma.
`if` := precede una clausola.
in place = in situ := caratterizzazione di una trasformazione effettuata su una variabilr o su una struttura di dati che si serve solo della memoria precedentemente occupata.
`#include` := direttiva di compilazione che comporta l'inclusione nel modulo attuale delle funzionalità del file indicato alla sua destra.
information hiding := tenere nascoste alcune informazioni in una function per evitare che che siano erroneamente modificate da chi può modificare il sorgente in causa.
`inline` := dichiarazione di una fuction semplice per evitare sua organizzazione autonoma.
insert / to = inserire caratteri successivi in una stringa
`int` := tipo di dati.
integer-literal := scrittura di costante intera.
invocazione := di una function da parte di una function chiamante
iterator := dato-rl che viene usato per scorrere i componenti di un contenitore
juxtaposition = giustapposizione, concatenazione := operazione elementare riguardante stringhe o sequenze di ogni genere di oggetti.
keyword = parola chiave := Syn. parola riservata.
lessema := unità lessicale elementare :: sin. token
lexical analyzer := analizzatore lessicale
LIFO := last in first out; meccanismo di gestione di uno stack.
linker := programma che collega i moduli compilati per produrre un programma eseguibile.
list := ::
literal := scrittura di costante ::
`long` := modificatore di tipo di dati.
`main` := nome della function principale di un programma.
map := contenitore i cui componenti sono associati a chiavi peculiari.
multimap := contenitore i cui componenti sono associati a chiavi che possono presentare ripetizioni.
method := metodo, procedimento costituente di una class in grado di agire sopra i suoi oggetti.
modifier / type := modificatore dei tipi di dati.
module := modulo di programma.
multiset := contenitore con componenti che possono presentare ripetizioni.
`naked` := ::
`new` := operatore per la richiesta di memoria dinamica, ossia la creazione di un dato-el nel run time.
null terminated string := stringa che ha NULL come ultimo carattere.

object := oggetto, complesso di dati-el costituente l'istanza di una class.
OOP := object oriented programming = programmazione orientata agli oggetti.
`operator` := operatore ::
parola riservata := stringa formate da lettere minuscole che nel linguaggio ha un ruolo ben definito. Sin. keyword.
pointer := puntatore :: J12d01
`private` := qualifica dei dati-el (variabili, ...) di una class; consente anche di dichiarare una class derivata di un'altra e quindi con la possibilità di ereditare da questa.
programma eseguibile := programma leggibile dalla macchina atto a governare le sue esecuzioni, ciascuna delle quali con il compito di risolvere una istanza del problema al quale è dedicata.
`protected` := qualifica di ... .
punctuator := segno di interpunzione.
`public` := qualifica dei dati-el (variabili, ...) di una class.
punctuator := segno di interpunzione :: serve a demarcare il codice sorgente.
`register` := registro interno; cella-mm a stretto contatto con i circuiti operativi della CPU.
recursion := ricorsione, esecuzione di una function che può invocare se stessa.
replace / to = sostituire caratteri successivi in una stringa
`return` := comando di uscita da una function.
scrittura di costante := scrittura che esprime un dato-el il cui valore non può cambiare nel corso di una esecuzione: può riguardare un numero intero, un numero floating, un carattere o una stringa.
run time := intervallo di tempo nel quale si svolge una esecuzione di un programma
set := contenitore con componenti senza ripetizioni disposti ordinatamente.
`short` := modificatore di tipo di dati.
`signed` := modificatore di tipo di dati.
size := ampiezza di una sequenza (array monodimensionale, stringa vector)
`sizeof` := ::
sort / to := ordinare.
stack := area di memoria che fa da deposito a pila per i records di attivazione relativi alle invocazioni di una function, ossia per le celle-mm dedicate ai dati-el all'interno della function.
`static` := keyword per dichiarare che una variabile all'interno di una fuction alla fina di ogni sua attivazione non va distrutta, ma conserva il suo valore.
string := stringa.
string-literal := scrittura di sequenza di caratteri.
`struct` := ::
`substr` := metodo per ottenere una sottostringa d una stringa data
swap / to := scambiare i valori di due variabili o di due contenitori di altro genere.
`switch` := ...
testo sorgente :: testo scritto in un linguaggio di programmazione.
text file := file con un testo
`this` := in una fuction è il puntatore all'oggetto della invocazione attuale della function e fa parte del record della attivazione attuale::
`thread` := ::
token := unità lessicale elementare in un linguaggio artificiale.
`true` := vero, implementato da b00000001=1.
type modifier := modificatore dei tipi di dati.
`typedef` := ::

`union` := ::
`unsigned` := modificatore di tipo di dati.
user defined := a disposizione del programmatore.
user defined data types := ::
UTF-8 := Unicode Transformation Format - 8-bit. [e[UTF-8]e]
vector := sequenza a lunghezza variabile.
`virtual` := dichiarazione all'interno di una class dotata di classes derivate::
`void` := tipo di dati neutro.
`volatile` := ::
`while` := precede clausola ::

J18 b. tipi di dati, modificatori ed estensioni

data type ampiezza in bytes intervallo dei valori
signed char – 1 – -128 .. 127
[unsigned] char – 1 – 0 .. 255
short [signed] int – 2 – -32,768 .. 32,767
short unsigned int – 2 – 0 .. 65,535
[signed] int – 4 – −2 147 483 648 .. 2 147 483 647
unsigned int – 4 – 0 .. 4,294,967,295
long [signed] int – 8 – -9,223,372,036,854,775,808 .. 9,223,372,036,854,775,807
long unsigned int – 8 – 0 .. 18,446,744,073,709,551,615
long long [signed] int – 8 – -9,223,372,036,854,775,808 .. 9,223,372,036,854,775,807
long long unsigned int – 8 – 0 to 18,446,744,073,709,551,615
double – 8 – circa 1.7E-308 .. 1.7E+308
long double – 16 – alta precisione con range dipendente dalla implementazione

J18 c. escape sequences

`\a` – alarm o beep – produce un breve tintinnio
`\b` – backspace – arretra il cursore di una posizione
`\f` – form feed – fa avanzare il cursore all'inizio della successiva pagina logica
`\n` – new line – fa avanzare il cursore all'inizio di una nuova linea
`\r` – carriage return – fa tornare il cursore all'inizio della linea corrente
`\t` – horizontal tab – inserisce blanks e sposta di conseguenza il cursore verso destra
`\v` – vertical tab – inserisce spazi in verticale
`\\` – backslash – emette il segno `\`
`\'` – single quote – emette un segno apice
`\"` – double quote – emette un segno doppio apice
`\?` – question mark emette un segno `?`
`\`*ooo* – octal number – scrittura di un numero ottale
`\x`*hh* – hexadecimal number – scrittura di un carattere esadecimale
`\0` – NULL – carattere NULL

J18 d. formati di emissione dei valori dei vari tipi

Sono espressi da digrammi o trigrammi che sono componenti delle stringhe argomenti di functions di output come `printf`

`%c` valori `char`
`%d` : scritture in notazione decimale
`%i` : scritture in notazione decimale
`%x` e `%X` : scritture in notazione esadecimale
`%o` : scritture in notazione ottale
`%l` e `%ld` : valori `long int`
`%u` : valori `unsigned`
`%f` : valori `float`
`%l` ftt : valori `double`
`%p` : indirizzo decimale di un variabile
`%s` : valori `string`
`%n` : numero dei bytes scritti sullo stack dalla function `printf`

J18 e. functions matematiche

Si trovano nella libreria `<cmath>` resa disponibile dalla direttiva
`#include <cmath>`
`abs(x)` : calcola il valore assoluto di x
`acos(x)` : calcola l'arcocoseno di x espresso in radianti
`acosh(x)` : calcola l'arcoseno iperbolico di x
`atan(x)` : restituisce l'arcotangente di x come valore compreso tra $-\pi/2$ e $\pi/2$ radianti
`atan2(y,x)` : restituisce l'angolo θ dalle coordinate polari corrispondenti alle cartesiane $\langle x, y\rangle$
`atanh(x)` : restituisce l'arcotangente iperbolico di x
`cbrt(x)` : restituisce la radice cubica di x
`ceil(x)` : restituisce il valore di x arrotondato al suo intero più vicino
`copysign(x,y)` : restituisce il valore in virgola mobile x con il segno del reale in virgola mobile y
`cos(x)` : restituisce il coseno di x espresso in radianti
`cosh(x)` : restituisce il coseno iperbolico di x
`exp(x)` : restituisce il valore di e^x
`exp2(x)` : restituisce il valore di $2x$
`expm1(x)` : restituisce $e^x - 1$
`erf(x)` : restituisce il valore della funzione degli errori in x
`erfc(x)` : restituisce il valore della funzione degli errori complementata in x
`fabs(x)` : restituisce il valore assoluto del numero in virgola mobile x
`fdim(x)` : restituisce la differenza positiva tra x e y
`floor(x)` : restituisce il valore di x approssimato per difetto al suo intero più vicino
`fma(x,y,z)` : restituisce `x*y+z` senza perdere precisione
`fmax(x,y)` : restituisce il massimo tra due argomenti in virgola mobile
`fmin(x,y)` : restituisce il minimo tra due argomenti in virgola mobile
`fmod(x,y)` : restituisce il resto in virgola mobile di `x/y`
`frexp(x,y)` : se x si esprime come`m*2n`, restituisce il valore di m (un valore compreso tra 0,5 e 1,0) e scrive il valore di n nella memoria del puntatore y
`hypot(x,y)` : restituisce $sqrt(x^2 + y^2)$ senza overflow o underflow intermedio
`ilogb(x)` : restituisce la parte intera del logaritmo in base b di xin virgola mobile
`ldexp(x,y)` : restituisce `x*2y`
`lgamma(x)` : restituisce il logaritmo del valore assoluto che la funzione gammaassume in x
`llrint(x)` : arrotonda x all'intero più vicino e restituisce il risultato come long long integer
`llround(x)` : arrotonda x all'intero più vicino e restituisce il risultato come long long integer
`log(x)` : restituisce il logaritmo naturale di x
`log10(x)` : restituisce il logaritmo in base 10 di x
`log1p(x)` : restituisce il logaritmo naturale di x+1
`log2(x)` : restituisce il logaritmo in base e del valore assoluto di x
`logb(x)` : restituisce il logaritmo in base e del valore assoluto di x
`lrint(x)` : arrotonda x a un intero vicino e restituisce il risultato come intero lungo
`lround(x)` : arrotonda x all'intero più vicino e restituisce il risultato come long integer
`modf(x, y)` : restituisce la parte decimale di x e scrive la parte intera nella memoria del puntatore y
`nan(s)` : restituisce un valore NaN (non un numero)
`nearbyint(x)` : restituisce x arrotondato all'intero più vicino

`nextafter(x,y)` : restituisce il numero in virgola mobile più vicino a x nella direzione di y
`nexttoward(x,y)` : restituisce il numero in virgola mobile più vicino a x nella direzione di y
`pow(x,y)` : restituisce il valore x elevato a y
`remainder(x,y)` : restituisce il resto di x/y arrotondato all'intero più vicino
`remquo(x,y,z)` : restituisce x/y arrotondato all'intero più vicino, scrive il risultato nella memoria del puntatore z e restituisce il resto.
`rint(x)` : restituisce x arrotondato a un intero vicino
`round(x)` : restituisce x arrotondato all'intero più vicino
`sin(x)` : restituisce il seno di x (x è in radianti)
`sinh(x)` : restituisce il seno iperbolico di x
`sqrt(x)` : restituisce la radice quadrata di x
`tan(x)` : restituisce la tangente di x (x è in radianti)
`tanh(x)` : restituisce la tangente iperbolica di x
`tgamma(x)` : restituisce il valore della funzione gamma in x
`trunc(x)` : restituisce la parte intera di X

J18 f. algoritmi generici

In gran parte sono resi disponibili dallo header `algorithm.h` :

adjacent_find
binary_search
bound
copy
copy
copy_backward
count
count_if
element
equal
equal_range
fill
fill_n
find
find_first_of
find_if
for_each
generate
generate_n
includes
inplace_merge
iter_swap
lexicographical_compare
lower_bound
make_heap
max
max_element
merge
min
min_
mismatch
next_
nth_
partial_sort
partial_sort_
partition
permutation
pop_heap
prev_permutation
push_heap
random_shuffle
ranges

remove
remove_copy
remove_copy_if
remove_if
replace
replace_copy
replace_copy_if
replace_if
reverse
reverse_copy
rotate
rotate_copy
search
set_difference
set_intersection
set_symmetric_difference
set_union
sort
sort_heap
stable_
stable_partition
swap
swap_
transform
unique
unique_
upper_

Inoltre sono resi disponibili dallo header in `numeric.h`:

accumulate adjacent_difference
inner_product

J18 g. header files

Gli header files standard per il linguaggoio C sono stati inseriti nella libreria standard di C++; in tal modo sono state rese disponibile per il linguaggio C++ tutte le funzionalità delle librerie standard per il linguaggio C.

La C++ standard library consiste dei seguenti headers:

algorithm	ios	map	stack
bitset	iosfwd	memory	stdexcept
complex	iostream	new	streambuf
dequeue	istream	numeric	string
exception	iterator	ostream	typeinfo
fstream	limits	queue	utility
functional	list	set	valarray
iomanip	locale	sstream	vector

La C standard library è invece costituita da:

assert.h	limits.h	stdarg.h	time.h
ctype.h	locale.h	stddef.h	wchar.h
errno.h	math.h	stdio.h	wctype.h
float.h	setjmp.h	stdlib.h	
iso646.h	signal.h	string.h	

Testo fruibile in https://www.mi.imati.cnr.it/alberto/ e https://arm.mi.imati.cnr.it/Matexp/matexp_main.php