Università degli Studi di Cassino e del Lazio Meridionale

Corso di Fondamenti di Informatica

Il sistema dei tipi in C++

Anno Accademico 2016/2017

Francesco Tortorella

Struttura di un programma C++

```
// Programma semplice in C++
#include <iostream>
using namespace std;


int main()
{
    cout << "Salve, mondo !\n";
    return (0);
}
```

I tipi di dato in C++

In C++ sono disponibili vari tipi di dato.

- Tipi numerici:
  - `int`
  - `float`
  - `double`
- Tipi non numerici:
  - `char`
  - `bool`
  - `void`

Il tipo `int`

- È costituito da un sottoinsieme limitato dei numeri interi
- Caratteristiche:
  - Dimensione: 4 bytes
  - Valore minimo: -2147483648
  - Valore massimo: +2147483647
- Operazioni ammesse:
  - Assegnazione =
  - Somma +
  - Sottrazione -
  - Moltiplicazione *
  - Divisione /
  - Confronto > < >= <= == !=

Il tipo `float`

- È costituito da un sottoinsieme limitato e discreto dei numeri reali
- Caratteristiche:
  - Dimensione: 4 bytes
  - Valore minimo (abs): 3.4E- 38
  - Valore massimo (abs): 3.4E+38
- Operazioni ammesse:
  - Assegnazione =
  - Somma +
  - Sottrazione -
  - Moltiplicazione *
  - Divisione/
  - Confronto > < >= <= == !=

Il tipo `double`

- È costituito da un sottoinsieme limitato e discreto dei numeri reali, ma con range e precisione maggiore rispetto a `float` (doppia precisione)
- Caratteristiche:
  - Dimensione: 8 bytes
  - Valore minimo (abs): 1.7E- 308
  - Valore massimo (abs): 1.7E+308
- Operazioni ammesse:
  - Assegnazione =
  - Somma +
  - Sottrazione -
  - Moltiplicazione *
  - Divisione /
  - Confronto > < >= <= == !=

Il tipo `char`

- Consiste in un insieme di caratteri, alcuni stampabili (caratteri alfabetici, cifre, caratteri di punteggiatura, ecc.) ed altri non stampabili tramite i quali si gestisce il formato dell'input/output (caratteri di controllo).
- I sottoinsiemi delle lettere e delle cifre sono ordinati e coerenti.
- Per la rappresentazione interna, viene tipicamente usato il codice ASCII, che mette in corrispondenza ogni carattere con un numero intero compreso tra 0 e 255.

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Il tipo `char`

- Di fatto anche `char` è un tipo numerico
- Caratteristiche:
  - Dimensione: 1 byte
  - Valore minimo: -128
  - Valore massimo: +127
- Sono permesse le operazioni aritmetiche tipiche degli interi

Il tipo `bool`

- È un tipo costituito da due soli valori **`false`** e **`true`**, corrispondenti a *falso* e *vero* e rappresentati da 0 e 1. Il tipo rappresenta le informazioni di tipo logico (es. il risultato di un confronto, il verificarsi di una situazione).
- Caratteristiche:
  - Dimensione: 1 byte
  - Valore minimo: `false`
  - Valore massimo: `true`
- Operazioni ammesse
  - assegnazione =
  - disgiunzione ||
  - congiunzione &&
  - negazione !

Operazioni sul tipo `bool`

x	y	AND x && y	OR x || y
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

Operazioni sul tipo `bool`

`x`	NOT `!x`
`false`	`true`
`true`	`false`

Modificatori di tipo

- Sono usati per creare nuovi tipi modificando i tipi base.
- **`signed`** e **`unsigned:`** senza ulteriori specificazioni, i tipi sono signed. Con il modificatore unsigned, il tipo è in grado di contenere soltanto valori non negativi
- **`unsigned char`**: 0…255
- **`unsigned int:`** 0…4294967295

Modificatori di tipo

- **`short`** e **`long:`** modificano l'estensione del tipo
  - **`short int`**: 2 byte
  - **`long int`**: 4 byte
  - **`long double`**: 12 byte
- I modificatori possono combinarsi:
  - **`unsigned short int`**
  - **`unsigned long int`**

Definizione di variabili

- Per usare una variabile, questa deve essere dapprima *definita*.
- La definizione rende disponibile la variabile che mantiene il tipo assegnato nella definizione fino al termine del programma.
- La sintassi è `<tipo> nome_variabile;`
- Esempi:
  - `int a;`
  - `int a,b,c;`
  - `float x,y,z;`

Definizione di variabili

- Il nome della variabile non può coincidere con una delle parole chiave riservate del C++:

```
asm, auto, bool, break, case, catch, char, class,
const, const_cast, continue, default, delete, do,
double, dynamic_cast, else, enum, explicit,
export, extern, false, float, for, friend, goto,
if, inline, int, long, mutable, namespace, new,
operator, private, protected, public, register,
reinterpret_cast, return, short, signed, sizeof,
static, static_cast, struct, switch, template,
this, throw, true, try, typedef, typeid, typename,
union, unsigned, using, virtual, void, volatile,
wchar_t, while
```

- I caratteri ammessi sono lettere, cifre e carattere di sottolineatura (underscore _), messi in qualunque ordine, purché il primo carattere del nome sia una lettera o l'underscore (sconsigliato).
- C'è differenza tra caratteri minuscoli e maiuscoli, per cui **`a`** e **`A`** sono due variabili diverse.
- Nello scegliere il nome per le variabili, è consigliabile orientarsi verso nomi significativi del ruolo della variabile nel programma.

Costanti

- Il C++ prevede tre modalità per definire delle costanti:
  - **Costanti letterali** (literals)
  - **Costanti definite** (`#define`)
  - **Costanti dichiarate** (`const`)

Costanti letterali

- **Il valore della costante è rappresentato direttamente.**
- **Costanti di tipo intero**

  Sono definite come sequenze di cifre decimali, eventualmente precedute da un segno (+ o -):
  0 -1 3256 +34 12L 33U 5321UL 0713 0X12FF 0XFUL
- **Costanti di tipo reale**

  Sono definite come sequenze di cifre decimali, eventualmente precedute da un segno (+ o -), strutturate in virgola fissa o in virgola mobile (floating point):
  0.1 -3.7 0.0001 1.0E-4 -7.6E12 4.
- **Costanti di tipo carattere**

  sono definite come caratteri racchiusi tra singoli apici (‘):
  ‘x’ ‘A’ ‘\n’ ‘\t’ ‘2’
- **Costanti di tipo stringa di caratteri**

  sono definite come sequenze di caratteri racchiusi tra doppi apici (“):
  “Pippo” “Valore di x: ” “x”
- **Costanti di tipo logico**

  sono solo due: `false` e `true`

Costanti definite

- Viene utilizzata la direttiva `#define` che permette di associare **testualmente** un valore ad un identificatore

```
#define MAX  12
#define PI 3.14
```

- All'atto della compilazione il *preprocessore* sostituisce ogni occorrenza dell'identificatore con il valore corrispondente

Costanti dichiarate

- Il valore viene associato ad un identificatore e ne viene specificato anche il tipo:

```
const int MIN=0;
const float PI=3.14;
```

- In questo caso l'identificatore è a tutti gli effetti una variabile non modificabile.

Operatori

- Un *operatore* specifica un'operazione da eseguire su uno o due *operandi* definendo un'*espressione*.
- L'ordine con cui sono valutati definisce la *precedenza* degli operatori. Può essere alterata con l'uso delle parentesi ().

Alcuni operatori	Assoc.
`!` `-` (*unario*) `+` (*unario*)	destra
`*` (*moltiplicazione*) `/` `%`	sinistra
`+` (*binario*) `-` (*binario*)	sinistra
`<` `<=` `>` `>=`	sinistra
`==` `!=`	sinistra
`&&`	sinistra
`||`	sinistra
`=`	destra

Espressioni

- Un’espressione consiste in un operando o in una combinazione di operandi e operatori.
- La valutazione di un’espressione porta al calcolo di un valore di un certo tipo.
- Esempi:

```
2+3   2.1*3.5+pigreco   a>=2
(7+2)/3   (a>2)&&(b==0)   !trovato
```

Espressioni

- Nel caso ci siano operandi di tipo diverso, l'espressione assume il tipo "più ampio" tra quelli presenti.
- Esempio:
  `7+5*2.4` → `19.0`
- **Achtung !** Qual è il valore dell'espressione?
  `7.5+1/2`
- Uso del casting: `7.5+(float)1/2`

Operatori

Operatore	Associatività
`()` `[]` `.` `->` `++` (*postfisso*) `--` (*postfisso*)	sinistra
`++` (*prefisso*) `--` (*prefisso*) `~` `!` `-` (*unario*) `+` (*unario*) `&` (*indirezione*) `*` (*indirizzo*) **`sizeof new delete`**	destra
(*tipo*) (*casting convenzionale*)	destra
`*` (*moltiplicazione*) `/` `%`	sinistra
`+` (*binario*) `-` (*binario*)	sinistra
`<` `<=` `>` `>=`	sinistra
`==` `!=`	sinistra
`&&`	sinistra
`||`	sinistra
`?` `:`	destra
`=` `*=` `/=` `%=` `+=` `-=` `&&=` `||=`	destra
`,` (*operatore virgola*)	sinistra