Corso Programmazione 2010-2011

Fabio Aiolli
E-mail: aiolli@math.unipd.it
Web: www.math.unipd.it/~aiolli

Dipartimento di Matematica Pura ed Applicata
Torre Archimede, Via Trieste 63

Variabili, Assegnamenti e Scope

Variabili

- Le variabili sono caratterizzate da
  - Nome (p.e. base, altezza) che la identifica
  - Tipo (p.e. float, int, char) che ne determina i possibili valori che può contenere e le possibili operazioni
  - L-VALUE, Riferimento o Indirizzo in RAM (p.e. F56A)
  - R-VALUE, Valore (p.e. 3.14, 7, 'c') il valore effettivo contenuto nello spazio di memoria della variabile
- Il nome può essere una qualsiasi sequenza di caratteri (senza spazi) con alcune eccezioni:
  - Il primo carattere è sempre alfanumerico (NO: 3gino, 44a)
  - No segni di interpunizione (NO: a^3, pippo+pluto) solo il carattere "_" (underscore) è permesso
- I tipi fondamentali sono: numero intero (int), numero reale (float,double), carattere (char), + void. (n.b. il linguaggio NON prescrive la dimensione in BYTE)

Tipi

- I problemi da risolvere fanno uso di valori di tipo diverso
- Accedere alle sequenze di bit in RAM nel modo giusto
- Permettere al compilatore di verificare (staticamente) se un programma è "sensato", ovvero se i valori sono usati in modo coerente con il tipo

- Vedremo in seguito come derivare dei tipi + sofisticati

- INTERI: con n bit riusciamo a rappresentare i numeri
  - -2^{n-1} … $2^{n-1}-1$ (complemento a 2 small endians)
  - Nell'header file <limits.h> si trovano definite le costanti INT_MIN e INT_MAX
- REALI: un bit per segno, x per la mantissa e y per l'esponente

- Dimensione in byte: sizeof(T) o sizeof(X) se X è di tipo T

Utilizzo delle variabili

- In C le variabili vanno prima di tutto DICHIARATE

  ```
  int a;
  float b;
  ```

  La dichiarazione serve in fase di esecuzione per "allocare" uno spazio di memoria adatto a contenere il valore della variabile (indicando il tipo) e ad associargli un nome.

- Generalmente vanno anche INIZIALIZZATE mediante un assegnamento

  ```
  a = 3;
  b = 3.14;
  ```

- Infine le si possono anche usare all'interno di espressioni usando opportuni operatori

L'area del rettangolo

```
/* Calcolo aerea rettangolo */

#include<iostream>
using namespace std;

main() {
  int base;
  int altezza;
  int aerea;

  base = 3;
  altezza = 7;
  area = base*altezza;

  cout << area << endl;
}
```

Dichiarazioni

Le costanti

- In alcuni casi, sappiamo a priori che l'informazione da memorizzare non cambia durante l'esecuzione del programma.

  P.e. Pi_greco = 3.14, LaEdiEulero= 2.8, …

- In questi casi, è più corretto (ma non è obbligatorio) utilizzare le cosiddette costanti

  const Pi_greco = 3.14;

  N.B. Nella definizione di costante la dichiarazione e l'inizializzazione sono contemporanee!!

Scope delle variabili

- I blocchi di istruzioni in C++ vengono definiti mediante le parentesi graffe.
- I blocchi possono essere *annidati* e se ne possono definire un numero arbitrario

- Le variabili sono *visibili* solo dal momento della dichiarazione fino al termine del blocco dove sono dichiarate e quindi anche in tutti i blocchi più interni
- Quindi via via che il programma procede solo alcune delle variabili dichiarate nell'intero programma sono visibili e utilizzabili.
- Attenzione: una variabile dichiarata dentro un blocco con lo stesso nome di una variabile dichiarata in un blocco più esterno, la *nasconde!*

Esempio di Scope

```
01: /* Esempio Scope */
02: #include<iostream.h>
03: using namespace std;
04: main() {
05:  int x;
06:  x=1;            // x in 05:
07:  {
08:    {
09:      x=x+2;      // x in 05:
10:    }
11:    int x=3;      // x in 11:
12:  }
13:  int x; // ILLEGALE!!!
14:  x=x*3;          // x in 05:
15: }
```

Dichiarazioni di variabili
tipo nome_della_variabile [= inizializzazione];

Esempi:
- int a;
- int a = 2;
- int a, b;
- float f = 2.345;
- char c = 'X';
- char messaggio[] = "Questa e' una stringa";

Facendo precedere il tipo dal qualificatore **const**, si ottiene una "variabile costante", il cui valore non potrà essere cambiato

const double pi = 3.141592653;

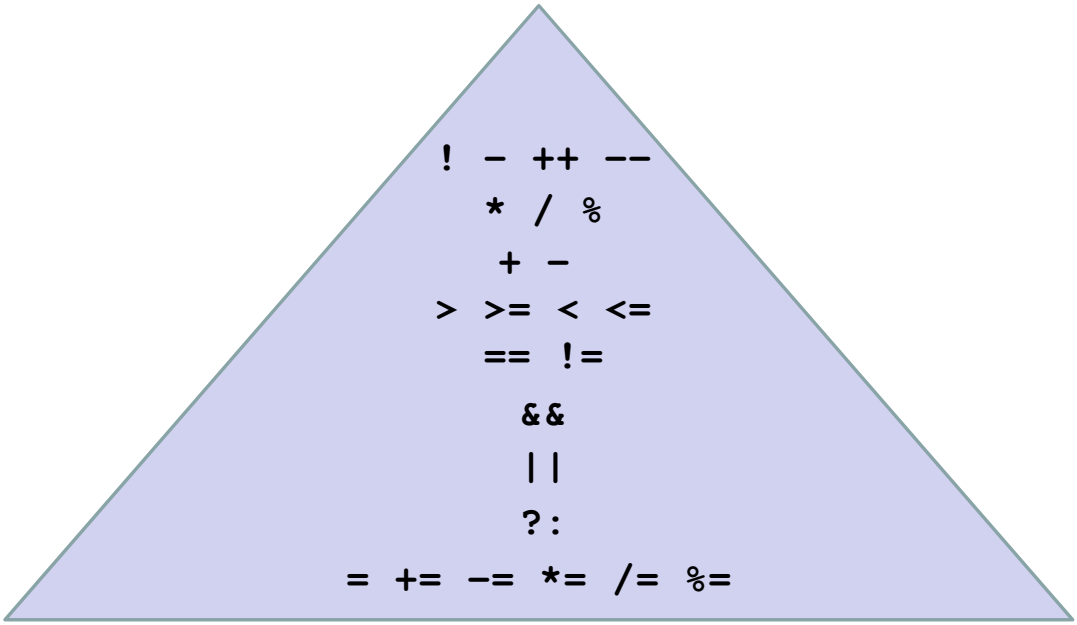
Operatori

Operatori senza effetti collaterali
- Aritmetici: +, -, *, /, %
- Relazionali: >, >=, <, <=, ==, !=
- Logici: &&, ||, !
- bitwise: &, |, ^, <<, >>, ~

Operatori con effetti collaterali
- Incremento e decremento: ++, --
- Assegnazione: =, +=, -=, *=, ecc.

Condizionale ternario: expr1 ? expr2 : expr

Precedenza Operatori



Operatori Logici

Expr1	Expr2	Expr1 && Expr2	Expr1 || Expr2	! Expr1
Zero	Zero	0	0	1
Zero	Non Zero	0	1	1
Non Zero	Zero	0	1	0
Non Zero	Non Zero	1	1	0

Valutazione short-circuit

La valutazione degli operatori logici && e || avviene in maniera short-circuit, cioè da sinistra verso destra e si interrompe non appena il risultato diventa noto.

(a != 0) && (++i < 10)
(a!=0) || (++i < 10)

Conversioni di Tipo

Quando un'espressione coinvolge valori di tipo diverso, essi vengono convertiti ad un tipo comune:

- Automaticamente ad un tipo “più grande”, che quindi non fa perdere informazione
- Esplicitamente, mediante un operazione di cast. Obbligatoria se non e' possibile una conversione implicita automatica.

Caratteri

In generale viene usata la **codifica standard ASCII**:

ogni carattere è rappresentato in 1 byte e quindi possiamo rappresentare 256 caratteri. Questo basta per:

a...z A...Z 0...9 . , ;: () etc

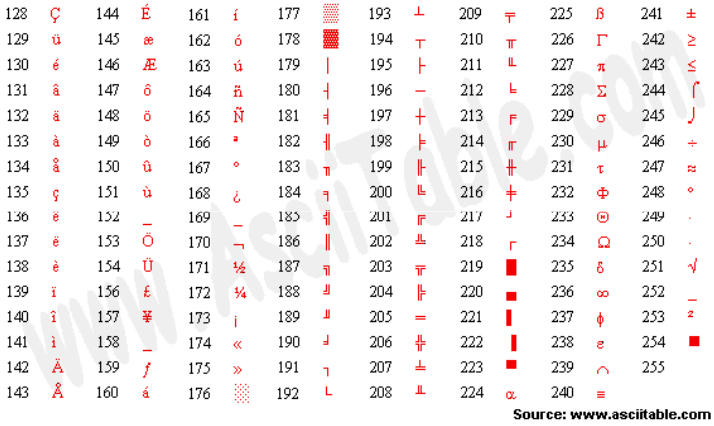
+ caratteri di controllo: Enter, Tab, etc

Tabella ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.asciitable.com

Tabella ASCII



128	Ç	144	É	161	í	177	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	163	ú	179	│	195	├	211	╙	227	π	243	≤
131	â	147	ô	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
134	å	150	û	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
136	ê	152	ÿ	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
137	ë	153	Ö	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	171	½	187	╗	203	╦	219	█	235	δ	251	√
139	ï	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
140	î	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
141	ì	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	
143	Å	160	á	176	░	192	└	208	╨	224	α	240	≡		

Source: www.asciitable.com