UNIVERSITÀ
DI TRENTO

Dipartimento di Ingegneria e
Scienza dell'Informazione
DISI - Trento

Programmazione 1

04 - Esercitazione

Stefano Berlato - stefano.berlato-1@unitn.it
Andrea Mazzullo - andrea.mazzullo@unitn.it
Giovanna Varni - giovanna.varni@unitn.it

Anno Accademico 2023/2024

Nelle puntate precedenti

- **Le variabili**
  - Nome oppure identificatore (e.g., `carattere`, `numero`, ...)
  - Tipo (e.g., `int`, `bool`, `float`, `char`, ...)
  - Locazione di memoria, l-value oppure indirizzo
  - Valore oppure r-value

Nelle puntate precedenti

- **Stream di input/output**

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int main() {
5.    char carattere;
6.
7.    cout << "Inserisci un carattere: ";
8.    cin >> carattere;
9.    cout << "Il carattere inserito è: " << carattere << endl;
10.
11.   return 0;
12. }
```

Nelle puntate precedenti

- **Operatori misti aritmetica/assegnazione**

```
x += y; x -= y; x *= y; x /= y; x %= y;
```

- **Operatori di (pre/post)-incremento/decremento unitario**

```
x++; ++x; x--; --x;
int valore = 6;
int i = valore++; // j = 6, valore = 7
int j = ++valore; // j = 8, valore = 8
```

Nelle puntate precedenti

- **Operazioni booleane (&&, ||, >, <, >=, <=, !, ==, !=)**

```
bool maggiore = 5 > 6;
bool and = true && false;
```

(attenzione agli operatori bit-a-bit)

`numero_1 ^ numero_2` (xor bit-a-bit)

Nelle puntate precedenti

- **Il tipo** `char`
  - Sottoinsieme del tipo `int` (è definita un aritmetica);
  - Codifica ASCII;
  - Definite le relazioni di precedenza e consecutività;

```
char carattere = 'a';
cout << (int) 'a' << endl; // 97
carattere += 5; // f
```

Nelle puntate precedenti

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

00 - Let’s begin!

- Tipi di errori in C++
  - **Errori di sintassi (o di compilazione)**: accadono quando il codice da noi scritto viola la sintassi del C++;
  - **Errori di runtime**: avvengono durante l’esecuzione del programma, nonostante il processo di compilazione abbia dato un risultato positivo;
  - **Errori di linker**: in questo caso, una volta generati i file oggetto, il linker non è in grado di combinarli per creare l’eseguibile finale;

00 - Let’s begin!

- Tipi di errori in C++
  - **Errori di sintassi (o di compilazione)**: accadono quando il codice da noi scritto viola la sintassi del C++;
  - **Errori di runtime**: avvengono durante l’esecuzione del programma, nonostante il processo di compilazione abbia dato un risultato positivo;
  - **Errori di linker**: in questo caso, una volta generati i file oggetto, il linker non è in grado di combinarli per creare l’eseguibile finale;

00 - Let’s begin!

- Tipi di errori in C++
  - **Errori di sintassi (o di compilazione)**: accadono quando il codice da noi scritto viola la sintassi del C++;
  - **Errori di runtime**: avvengono durante l’esecuzione del programma, nonostante il processo di compilazione abbia dato un risultato positivo;
  - **Errori di linker**: in questo caso, una volta generati i file oggetto, il linker non è in grado di combinarli per creare l’eseguibile finale;

00 - Let’s begin!

- Tipi di errori in C++
  - **Errori di sintassi (o di compilazione)**: accadono quando il codice da noi scritto viola la sintassi del C++;
  - **Errori di runtime**: avvengono durante l’esecuzione del programma, nonostante il processo di compilazione abbia dato un risultato positivo;
  - **Errori di linker**: in questo caso, una volta generati i file oggetto, il linker non è in grado di combinarli per creare l’eseguibile finale;

00 - Let’s begin!

- Tipi di errori in C++
  - **Errori di sintassi (o di compilazione)**: accadono quando il codice da noi scritto viola la sintassi del C++;
  - **Errori di runtime**: avvengono durante l’esecuzione del programma, nonostante il processo di compilazione abbia dato un risultato positivo;
  - **Errori di linker**: in questo caso, una volta generati i file oggetto, il linker non è in grado di combinarli per creare l’eseguibile finale;

90%

00 - Let’s begin!

- **Errore di Compilazione**

```
foo.cc: In function ‘int main()’:
foo.cc:5:3: error: expected ‘,’ or ‘;’ before ‘return’
```

- **Warnings**

```
foo.cc: In function ‘int main()’:
foo.cc:7:12: warning: division by zero [-Wdiv-by-zero]
cout << a/0 << endl;
```

00 - Let’s begin!

- Errori di sintassi
  - **Punto e virgola (;) mancante alla fine delle istruzioni;**
  - **Utilizzare una variabile senza averla dichiarata;**
  - **Utilizzare una funzione senza aver incluso la libreria corrispondente;**
  - **Utilizzo errato delle parentesi;**
  - ….

00 - Let’s begin!

```
1.  using namespace std;
2.  int Main(
3.  {
4.   int a = 0;
5.   char = "a";
6.
7.   court << a << endl;
8.   cout << char << endl;
9.
10.  return 0
11.  }
```

00 - Let’s begin!

```
1.  using namespace std;
2.  int Main(
3.  {
4.   int a = 0;
5.   char = "a";
6.
7.   court << a << endl;
8.   cout << char << endl;
9.
10.  return 0
11. }
```

1. Manca #include <iostream>
2. Parentesi mancante, nome main errato
3.
4.
5. Nome variabile, assegnamento errato
6.
7. Nome istruzione errato
8. Istruzione errata
9.
10. Punto e virgola mancante
11.

00 - Let’s begin!

- Errori di sintassi
  - **Punto e virgola (;) mancante alla fine delle istruzioni;**
  - **Utilizzare una variabile senza averla dichiarata;**
  - **Utilizzare una funzione senza aver incluso la libreria corrispondente;**
  - **Utilizzo errato delle parentesi;**
  - ….

Il compilatore ci avviserà di questi errori.
Spesso però i messaggi che si ottengono sono abbastanza “criptici”.

00 - Let’s begin!

- Errori di runtime
  - **Divisione per 0 o altre operazioni che producono valori non validi** (e.g., `-inf`, `+inf`, `NaN`, etc.);
  - **Dare input errati al programma** (mancati controlli);
  - **Overflow delle variabili** (e.g., numeri troppo grandi per essere rappresentati dal tipo int).

Gli errori di runtime sono più difficili da identificare e sono anche in grado di causare i danni maggiori (vedi la fallita Missione Cluster ESA del 1996).

00 - Let’s begin!

```
1.   int a = 5;
2.   cout << a/0 << endl;
3.
4.   int a = 2147483647;
5.   cout << a+100 << endl;
6.
7.   int x;
8.   int y = x * 2;
9.
10.  int x;
11.  x == 5;
12.  cout << x;
```

00 - Let's begin!

```
1.   int a = 5;
2.   cout << a/0 << endl;
3.
4.   int a = 2147483647;
5.   cout << a+100 << endl;
6.
7.   int x;
8.   int y = x * 2;
9.
10.  int x;
11.  x == 5;
12.  cout << x;
```

1.
2. Divisione per 0
3.
4.
5. Overflow della variabile int
6.
7.
8. Uso di variabile non inizializzata (non sempre risulta in errore a run-time, ma più generalmente è un comportamento indefinito)
9.
10.
11. Non un errore di run-time, ma comunque un errore

00 - Let’s begin!

- `g++ -Wall -o output.out codice.cc`
  - L’istruzione `-Wall` istruisce il compilatore in modo da segnalare ogni possibile “warning” nel codice che avete scritto (e.g., variabile non utilizzate, conversione tra tipi errate, etc.)

1 - Implicazione

Scrivere un programma che calcoli la tabella di verità dell'operatore implicazione (P $\longrightarrow$ Q). Il Programma riceve il valore di P e Q in input dall'utente

P	Q	$P \rightarrow Q$
T	T	T
T	F	F
F	T	T
F	F	T

1 - Implicazione

Scrivere un programma che calcoli la tabella di verità dell'operatore implicazione (P $\longrightarrow$ Q). Il Programma riceve il valore di P e Q in input dall'utente

$$(\neg P \vee Q)$$

```
(! A || B)
```

2 - Valore Assoluto

Scrivere un programma che, dati in input due numeri, a e b, calcoli il risultato, in valore assoluto, dell'operazione (a-b).
(senza utilizzare funzioni di libreria o istruzioni if-else)

3 - Maggiore e minore

Scrivere un programma che, dati in input due numeri, a e b, li salvi in due variabili distinte, max e min, in cui la prima conterrà il numero maggiore tra i due e la seconda il più piccolo.
(senza utilizzare funzioni di libreria o istruzioni if-else o operatore ternario o cicli)

Stampare poi queste variabili a video.

3 - Maggiore e minore

Scrivere un programma che, dati in input due numeri, a e b, li salvi in due variabili distinte, max e min, in cui la prima conterrà il numero maggiore tra i due e la seconda il più piccolo.
(senza utilizzare funzioni di libreria o istruzioni if-else)

Stampare poi queste variabili a video.

```
#include <cmath>
float assoluto = abs(-5.0) // 5.0
```

Reference alla libreria `<cmath>`
http://www.cplusplus.com/reference/cmath/

4 - Uguali

Scrivere un programma che, dati in input due numeri interi, a e b, controlli se i due numeri sono uguali. Il programma deve stampare a video il risultato del confronto tramite una variabile booleana
(senza utilizzare funzioni di libreria o istruzioni if-else oppure “==”)