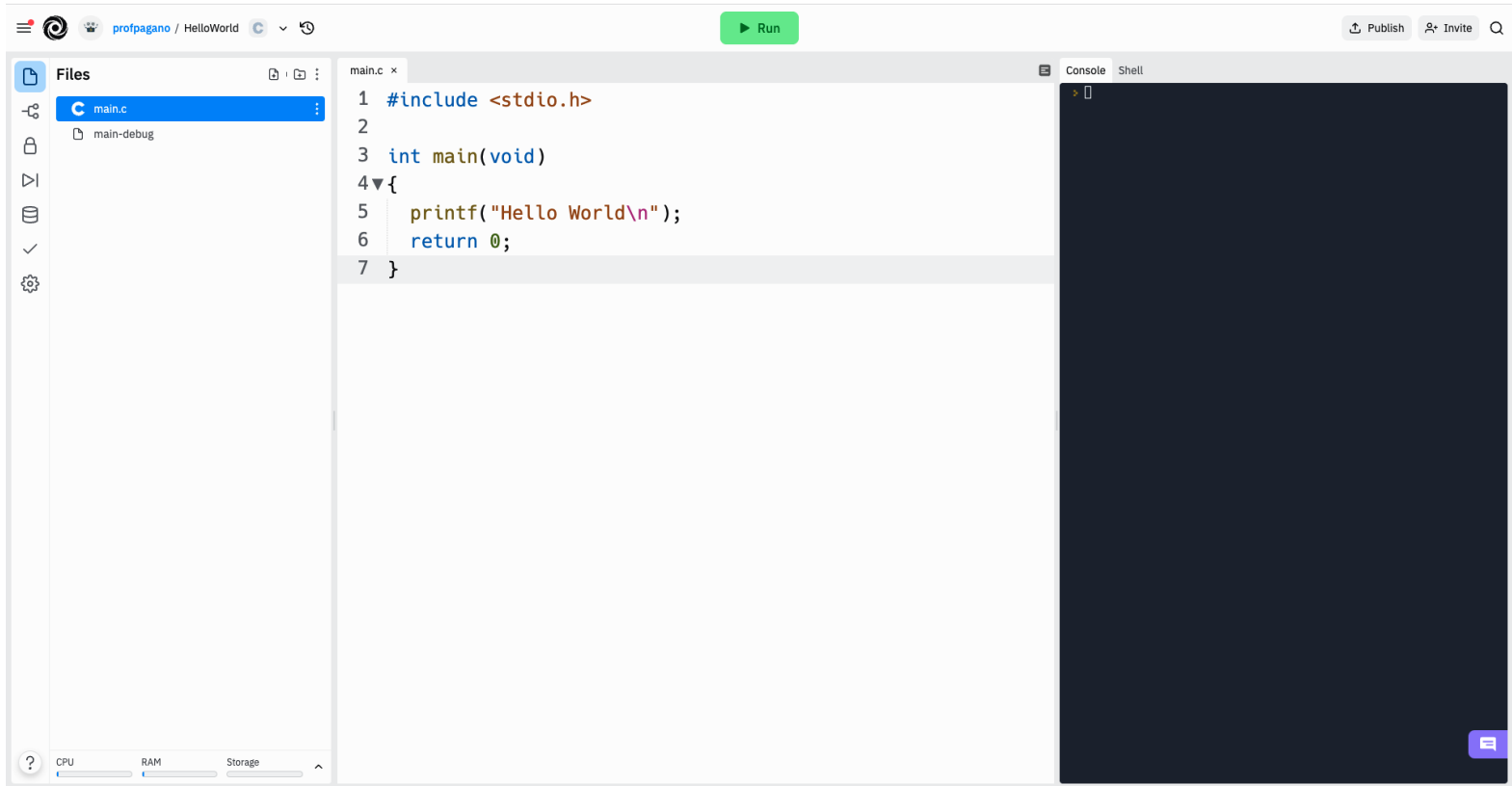
La programmazione in C

Sommario

- L’ambiente online Replit
- La struttura di base dei programmi in C
- I tipi di dati fondamentali
- Variabili
- Costanti

L’ambiente online Replit



La struttura di base dei programmi in C

L'esempio mostra la struttura di un tipico programma in C.

L'istruzione `` `int main(void)` `` dichiara una funzione che non è di una libreria: infatti **main** è la funzione principale, che dev'essere presente in ogni programma in C.

```c
#include <stdio.h>

int main(void)
{
  int eta;
  char pokemon[] = "Bulbasaur";
  eta = 6;

  printf("Il mio pokemon preferito è: %s", pokemon);
  printf(", da quando avevo %d anni.\n", eta);

  return 0;
}
```

La struttura di base dei programmi in C

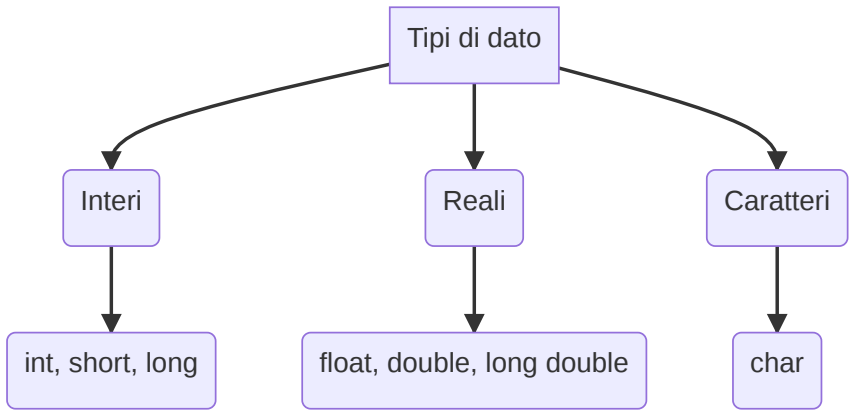
Un programma si compone di tre parti fondamentali:

```c
#include <stdio.h>

int main(void)
{
  int eta;
  char pokemon[] = "Bulbasaur";
  eta = 6;

  printf("Il mio pokemon preferito è: %s", pokemon);
  printf(", da quando avevo %d anni.\n", eta);

  return 0;
}
```

1. Librerie
2. Main
3. Corpo del programma

Ogni istruzione deve terminare con il punto e virgola (;).

I tipi di dati fondamentali



Interi

con segno (signed)

Tipo	N° bit	Minimo	Massimo
short	16	-32 768	+32 767
int	32	-2 147 483 648	+2 147 483 647
long	64	-9.223372e+18	+9.223372e+18

Interi

senza segno (unsigned)

Tipo	N° bit	Minimo	Massimo
unsigned short	16	0	65 5365
unsigned int	32	0	4 294 967 295
unsigned long	64	0	circa 1.84e+19

Reali

senza segno (unsigned)

Tipo	N° bit	Cifre significative
float	32	6-7
double	64	15-16
long double	128	?

Caratteri

Una variabile `char` occupa 8 bit di memoria.

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

Caratteri

```c
#include <stdio.h>

int main(void)
{
  char var1 = 'A';
  char var2 = 65;

  printf("Le due variabili hanno lo stesso contenuto!\n");
  printf("Contenuto di var1: %c\n", var1);
  printf("Contenuto di var2: %c\n", var2);

  return 0;
}
```

Variabili

Una variabile è un'area di memoria nella RAM caratterizzata da:

- tipo
- nome (*identificatore*)

Dichiarazione

Per poter utilizzare una variabile è necessario dichiararla, ovvero comunicare al compilatore che si intende utilizzare un dato di un certo tipo.

```
1   int numero, x, y;
2   float temperatura;
3   char car;
```

Inizializzazione

All'atto della dichiarazione, è possibile assegnare un valore alle variabili. (consigliato)

```
1   int numero, x = 0, y = 0;
```

Variabili

Altre operazioni:

Assegnazione di un valore

```
1   numero = 666; 🤟😈
```

Assegnazione di una variabile

```
1   numero = x;
```

Assegnazione di un'espressione

```
1   numero = 666 / 2;
2   y = numero + x;
```

È possibile eseguire operazioni arimetiche solo tra *variabili dello stesso tipo*, altrimenti il compilatore ci segnalerà un errore (studieremo più avanti alcune eccezioni!).

Identificatori

Alcune regole da rispettare per scegliere i nomi delle variabili/costanti.

Un identificatore può essere composto da:

- lettere maiuscole e minuscole
- numeri
- simbolo di underscore

```
Pippo, x1, x2, mano_dx, mano_sx ✅
```

Un identificatore non può:

- iniziare con un numero
- essere una parola chiave del linguaggio C (es: main, return …)

```
10_x, main, return, ❌
```

Il linguaggio C è sensibile alle maiuscole/minuscole, quindi l'identificatore `id` è diverso da `ID`.

Calcolo dell’area di un rettangolo

```c
#include <stdio.h>

int main(void)
{
  float area;
  float base;
  float altezza;

  base = 3;
  altezza = 5;
  area = base * altezza;

  printf("Area rettangolo: %f\n", area);

  return 0;
}
```

Cenni sulla funzione printf

Ci permette di comunicare in output (sulla console) i risultati dell'elaborazione.

```c
#include <stdio.h>

int main(void)
{
  int i = 2;
  float r = 7.8;
  char car = '@';

  printf("Questa è una stringa 123@! \n");
  printf("Numero intero: %d \n", i);
  printf("Numero reale: %f \n", r);
  printf("Nome: %s \n", "Salvo");
  printf("Carattere: %c \n", car);


  return 0;
}
```

stringa di caratteri

%d per stampare un intero

%f per stampare un reale

%c per stampare un carattere

%s per stampare una stringa

\n per andare a capo

Scambio di variabili

```c
int main(void)
{
  float a = 3;
  float b = 5;
  float temp;

  printf("Valore a prima: %f \n", a);
  printf("Valore b prima: %f \n", b);

  temp = a;
  a = b;
  b = temp;

  printf("Valore a dopo: %f \n", a);
  printf("Valore b dopo: %f \n", b);

  return 0;
}
```

Costanti

Una costante è un’area di memoria nella RAM che identifica dati che **non possono essere modificati**.

Nel linguaggio C esistono due modi per definire una costante:

- attraverso la parola chiave `` `const` ``
- attraverso la direttiva al preprocessore `` `#define` ``

Nel primo caso, la costante viene vista come una "variabile con valore fisso".

Nel secondo caso, il compilatore sostituirà a tutte le occorrenze dell’identificatore il valore associato, senza che vi sia alcuna locazione di memoria associata alla costante.

Area del cerchio

```c
#include <stdio.h>

int main(void)
{
  const float PIGRECO = 3.1415;
  float raggio = 5;
  float area;

  area = raggio * raggio * pigreco;
  printf("Area del cerchio: %f \n", area);

  return 0;
}
```

const per definire una costante

convenzione: costanti con identificatori maiuscoli

Area del cerchio

```c
#include <stdio.h>
#define PIGRECO 3.1415

int main(void)
{
  float raggio = 5;
  float area;

  area = raggio * raggio * PIGRECO;
  printf("Area del cerchio: %f \n", area);

  return 0;
}
```

define per definire una costante

convenzione: costanti con identificatori maiuscoli

Operatori aritmetici

Operatore	Esempio	Significato
+	a + b	Somma tra a e b
-	a - b	Differenza tra a e b
*****	a * b	Moltiplicazione tra a e b
/	a / b	Divisione tra a e b
%	a % b	Resto della divisione intera tra a e b