Lezione 4

Elementi lessicali e espressioni logiche

Linguaggi di Programmazione I

Matricole 2-3

Linguaggi di Programmazione I

Elementi lessicali

• il linguaggio C ha un suo "vocabolario di base" i cui elementi sono detti token

- esistono 6 tipi di token:
 - parole chiave
 - identificatori
 - costanti
 - costanti stringa
 - -operatori
 - -simboli di interpunzione

Linguaggi di Programmazione I



Elementi lessicali

- Il primo passo del processo di compilazione consiste nell'*analisi lessicale* del programma.
- Durante l'analisi lessicale
 - i caratteri del programma vengono raggruppati in token
 - vengono rimossi i commenti (ciascun commento è sostituito con uno spazio)
- La sequenza di token prodotta durante l'analisi lessicale viene poi sottoposta all'analisi sintattica per controllare che formi una stringa legale secondo la sintassi del linguaggio

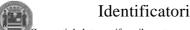
Linguaggi di Programmazione I



La parole chiave

 Parole riservate al linguaggio che non possono essere ridefinite o usate in altri contesti dal programmatore

| auto | do | goto | signed | unsigned |
|----------|--------|----------|---------|----------|
| break | double | if | sizeof | void |
| case | else | int | static | volatile |
| char | enum | long | struct | while |
| const | extern | register | switch | |
| continue | float | return | typedef | |
| default | for | short | union | |



Composti da lettere, cifre e il carattere _

- il primo carattere deve essere una lettera o il carattere _
- Usati per definire nomi unici per oggetti in un programma (variabili, funzioni etc.)
- · Lunghezza degli identificatori:
- ANSI C: 31 caratteri (almeno)
- E' buona regola usare identificatori con un significato:

- esempio: $x=y^*z$; NO! area = base *altezza; OK!

 Gli identificatori che iniziano con il carattere _ dovrebbero essere utilizzati esclusivamente dai programmatori di sistema

Linguaggi di Programmazione I



Costanti

· Rappresentano valori

• Costanti intere: 50 -7

• Costanti virgola mobile: 13.5 9.0

• Costanti carattere: 'a' ';' '\n'

Linguaggi di Programmazione I



Costanti stringa

sequenze di caratteri racchiuse tra doppie virgolette

- "Hello world!"
- \bullet per utilizzare il carattere " all'interno di una stringa bisogna farlo precedere da \backslash
 - "Antonella gli chiese : \" come ti chiami?\" "
- \bullet per utilizzare il carattere \setminus all'interno di una stringa bisogna farlo precedere da un altro \setminus
 - "il carattere \\ è chiamato backslash"
- le sequenze di caratteri che potrebbero avere un significato al di fuori della costante stringa vengono considerate semplici sequenze di caratteri all'interno delle doppie virgolette:
 - "a=1*2;" non esegue alcuna operazione

Linguaggi di Programmazione I

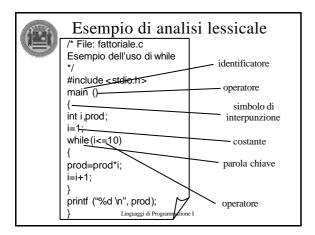


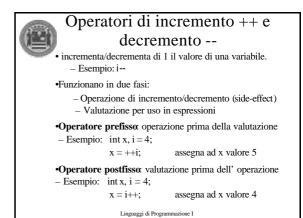
Simboli di interpunzione e operatori

- I **segni di interpunzione** permettono al compilatore di separare e distinguere gli elementi del linguaggio
 - Esempi : () ; , { }
- •Gli operatori sono caratteri a cui è associata una operazione
 - Esempi: gli operatori aritmetici: + * / %
- Gli operatori separano gli identificatori. Comunque separare operatori con spazi migliora la leggibilità del programma

 $x=z^*(3.16+y/2)$; $x=z^*(3.16+y/2)$;

- Le parentesi () dopo il nome di una funzione vengono trattate come operatori $\,$







Priorità ed associatività degli operatori

- Regole per stabilire l'ordine di applicazione degli operatori
- Operatori con priorità più alta vengono applicati prima di quelli a priorità più bassa
- •A parità di priorità si segue la associatività (che può essere da sinistra o da destra)

Linguaggi di Programmazione I



Priorità ed associatività degli operatori

• Di seguito sono elencati alcuni operatori in ordine di priorità decrescente. Gli operatori sulla stessa riga hanno uguale priorità. Per ciascuna riga è indicata l'associatività degli operatori che si trovano su quella riga:

| Operatori | Associatività |
|-----------------------------------|---------------|
| () ++ (postfisso) (postfisso) | Da sinistra |
| + -(unari) ++(prefisso)(prefisso) | Da destra |
| * / % | Da sinistra |
| + - | Da sinistra |
| = += -= *= /= %= | Da destra |
| I inquaggi di Programmaziona I | |



Alcuni esempi

int a=3, b=4, c=2;

| espressione | espressione equivalente | valore |
|-------------|-------------------------|--------|
| a*b/c | (a*b)/c | 6 |
| a+b*c | a+(b*c) | 11 |
| b/c-++a | ((b)/c)-(++a) | -2 |
| a c * ++ b | a - ((-c)*(++b)) | 13 |
| a +=b=3+ c | a+=(b=(3+c)) | 8 |
| | Times and di December 1 | |



Espressioni logiche

- espressioni logiche (booleane): assumono valori di true o
- Il C non definisce un tipo di dati booleano.
 - si usa il tipo di dati int
 - false è 0
 - true è qualsiasi valore diverso da 0
- nelle espressioni logiche vengono utilizzati gli operatori relazionali, di uguaglianza e logici

Linguaggi di Programmazione I



Operatori relazionali, di uguaglianza

- Operatori relazionali:
 - maggiore
 - minore
 - maggiore o uguale
 - minore o uguale
- Operatori di uguaglianza
- == uguale
- != diverso

Linguaggi di Programmazione I



Operatori relazionali

>, <, >= ,<=

- \bullet operazioni binarie che restituiscono un risultato di tipo int che può essere $0\ o\ 1$
- · associatività da sinistra
 - attenzione!

i=9;

printf ("%d\n", 5 < i < 8)

stampa 1 in quanto 5< i < 8 è equivalente a (5<i) <8

- a < b è equivalente a (a-b) < 0
 - attenzione! Il risultato dipende dalla precisione della

macchina:

y = 0.0001;

printf("%d\n", x < x + y)

può stampare 0 in quanto x e x+y potrebbero essere uguali in base alla precisione della macchina Linguaggi di Programmazione I



Operatori di uguaglianza

- == (uguale), != (diverso)
- · associatività da sinistra
- a != b è equivalente a !(a == b)
- a == b è implementata come (a-b) == 0;
- alle espressioni che sono operandi di operatori di uguaglianza vengono applicate le usuali regole di conversione aritmetica
 - -1'espressione 0.0 == 0 ha valore 1

Linguaggi di Programmazione I



Operatori logici

- Operatori logici
 - ! NOT logico (unario)
 - OR logico (binario)
 - && AND logico (binario)

Linguaggi di Programmazione I



NOT logico

- Il NOT logico (negazione logica) restituisce un valore *int*.
- !expr è uguale a:
 - 0 (false) se espr è 1 (true)
 - 1 (true) se esprè 0 (false)
- Il NOT logico è associativo a destra:
 - !!expr è equivalente a !(!expr)

Linguaggi di Programmazione I



OR logico

- L' OR logico restituisce un valore int.
- expr1 || espr2 è uguale a:
 - 0 (false) se espr1 e expr2 sono entrambe 0 (false)
 - 1 (true) se almeno una tra espr1 e expr2 è 1 (true)
- L' OR logico è associativo a sinistra:
 - expr1 || expr2 || expr3 è equivalente a(expr1 || expr2) || expr3



AND logico

- L' AND logico restituisce un valore int.
- expr1 && espr2 è uguale a:
 - -0 (*false*) se almeno una tra **espr1** e **expr2** è **0** (*false*)
 - -1 (true) se espr1 e expr2 sono entrambe 1 (true)
- L' AND logico è associativo a sinistra:
 - expr1 && expr2 && expr3 è equivalente a (expr1 && expr2) && expr3

Linguaggi di Programmazione I

Priorità ed associatività degli operatori relazionali, di uguaglianza e logici

| Operatori | Associatività |
|-------------------------------------|---------------|
| () ++ (postfisso) (postfisso) | Da sinistra |
| + -(unari) ++(prefisso)(prefisso) ! | Da destra |
| * / % | Da sinistra |
| + - | Da sinistra |
| < <= > >= | Da sinistra |
| == != | Da sinistra |
| && | Da sinistra |
| | Da sinistra |
| = += -= *= /= %= | Da destra |
| | |

Linguaggi di Programmazione I



Alcuni esempi

int a=3, b=4, c=2; double d = 0.0; char x = 'B';

| espressione | espressione equivalente | valore | | | |
|-----------------------------------|---------------------------------|--------|--|--|--|
| ! x | ! x | 0 | | | |
| ! d | ! d | 1 | | | |
| a && c && d | (a && c) && d | 0 | | | |
| $d \parallel a \; \&\& \; b-4$ | $d \parallel (a \&\& (b-4))$ | 0 | | | |
| $3 < = b - 2 \parallel a + 4 > 3$ | > 6 (3 <= (b-2)) ((a+4) > 6) |) 1 | | | |
| Linguaggi di Programmazione I | | | | | |

1

Lo short-circuit della valutazione

- Efficienza del C: la valutazione di espressioni che sono operandi di \parallel e && la valutazione si arresta non appena risulti noto il valore del risultato.
- Esempio:
- nell' espressione expr1 && expr2

expr2 viene valutata solo se expr1 è true

- nell' espressione expr1 || expr2

expr2 viene valutata se expr1 è false

