Esercitazione Programmazione Procedurale con Laboratorio

Puntatori

Esercizio 1 Valutazione espressioni con puntatori

Valutare e definire il tipo delle seguenti espressioni. Inserire le parentesi per sottolineare l'ordine della valutazione in base alla priorità degli operatori.

```
int i = 3, j = 5, *p = &i, *q = &j, *r;
double x;

p == & i;
* * & p;
r = &x;
7 * * p / * q + 7;
* (r = & j) *= *p;

```

Soluzione dell'esercizio 1

Le espressioni sono tutte di tipo intero:

- $p == (\&i)$, valore 1
- $^*(^*(\&p))$, valore 3
- $r = (\&x)$, il compilatore restituisce un *warning*, conversione implicita da * *double* a *int* *
- $(((7^*(^*p)))/(^*q) + 7$, valore 11
- $(^*(r = (\&j)))* = (^*p)$, valore 15

Ricordarsi $^*r$ è un *lvalue* dato che è equivalente alla variabile a cui punta, cioè j.

Esercizio 2 Assegnamenti legali e illegali con i puntatori

Valutare quali dei seguenti assegnamenti sono legali e quali sono illegali (warning/errori). Provare direttamente programmando l'esempio sottostante.

```
int *p;
float *q;
void *v;

p = 0;
p = 1;
p = (int*) 1;
p = v = q;
v = 1;
p = q;
p = (int*) p;

```

Soluzione dell'esercizio 2

Per illegale, in tutti gli esempi sotto si ottiene un warning. I warning, anche se non sono errori, denotano un possibile uso sbagliato del linguaggio. Per ognuno dei seguente punti illegali, basta convertire al tipo puntatore corretto: per esempio il punto tre corregge il punto due.

- Legale (a differenza dell'esempio sotto, 0 è un intero che può essere assegnato ad un puntatore, per esempio, anche *NULL* è definito come 0 in *stdlib.h*).

- Illegale (da *int* a $^*int$). In questo caso, il possibile uso sbagliato è dato dal fatto di assegnare al puntatore p una costante intera che in realtà non sappiano cosa riferisca in memoria.
- Legale
- Legale (assegnare un puntatore di qualsiasi tipo ad un puntatore a void o viceversa non provoca nessun warning, come invece accade nel penultimo caso tra $^*float$ a $^*int$)
- Illegale (da *int* a $^*void$)
- Illegale (da $^*float$ a $^*int$, qui i tipi dei puntatori sono diversi)
- Legale

Esercizio 3 Trova gli errori

Individuare quali comandi generano un errore o un *warning* a tempo di compilazione, e dire perché.

```c
#include <stdio.h>

int main(void) {

  int a= 6, b= 7;
  const int* p= &a;
  int* const q= &a;
  int* r= &b;

  *q= a+1;

  printf("a= %d\n", a);

  a= a++;

  printf("a= %d\n", a);

  *p= a+1;

  printf("a= %d\n", a);

  q= r;

  a= *r;

  printf("a= %d\n", a);

}
```

Soluzione dell'esercizio 3

Gli errori a tempo di compilazione sono:

- $a = a{+}{+}$; ritorna un *warning*: due effetti collaterali nello stesso comando (*warning: multiple unsequenced modifications to 'a'*).
- $^*p = a + 1$; ritorna un errore: p è dichiarato come un puntatore a costante e questo comando modifica la varbiabile da esso puntato (*error: read-only variable is not assignable*).
- $q = r$; ritorna un errore: q è dichiarato come una costante puntatore e questo comando assegna il valore di p a q (*error: read-only variable is not assignable*).

Esercizio 4 Trova gli errori

Trova gli errori in queste dichiarazioni di variabili.

```c
int main(void) {

  float p* = null;
  int a= 3;
  double a= 3.2;
  int b= 0;
  long int B= 0;

}
```

Soluzione dell'esercizio 4

```c
#include<stdlib.h>

int main(void) {

  float *p = NULL; // 2 errori di sintassi: l'asterisco dopo l'identificatore della variabile puntatore
     p, e null non esiste, esiste NULL
  int a= 3;
  double a= 3.2; // ci sono due variabili con lo stesso identificatore "a"
  int b= 0;
  long int B= 0;

}
```

Esercizio 5 Stringhe

Leggere una stringa da tastiera (*scanf()*), inserendola in un array *char stringa[50]*. Memorizzare la lunghezza della stringa letta (utilizzare la funzione *strlen()*, importando *string.h*) in una variabile *len*. Scandire l'array *stringa* dall'ultima posizione con un ciclo *for*; stampando al stringa al contrario, un carattere per ogni iterazione del *for*: esempio, se *stringa* contiene *Francesco*, stampare *ocsecnarF*. Nelle tre espressioni del *for*, nella prima inizializzare un puntatore *c* all'ultimo elemento da stampare, nell'espressione di controllo controllare che *c* non diventi minore di *stringa*, nella terza decrementare *c* di uno ad ogni iterazione del ciclo *for*. Questa rappresenta un uso dell'artimetica dei puntatori.

Soluzione dell'esercizio 5

```c
#include <stdio.h>
#include <string.h>

int main() {
char stringa[50];

scanf("%s", stringa);

int len= strlen(stringa);

for(char* c = stringa + len -1; c >= stringa; c--)

putchar(*c);

putchar('\n');
return 0;
}

```

Esercizio 6 Aritmetica dei puntatori

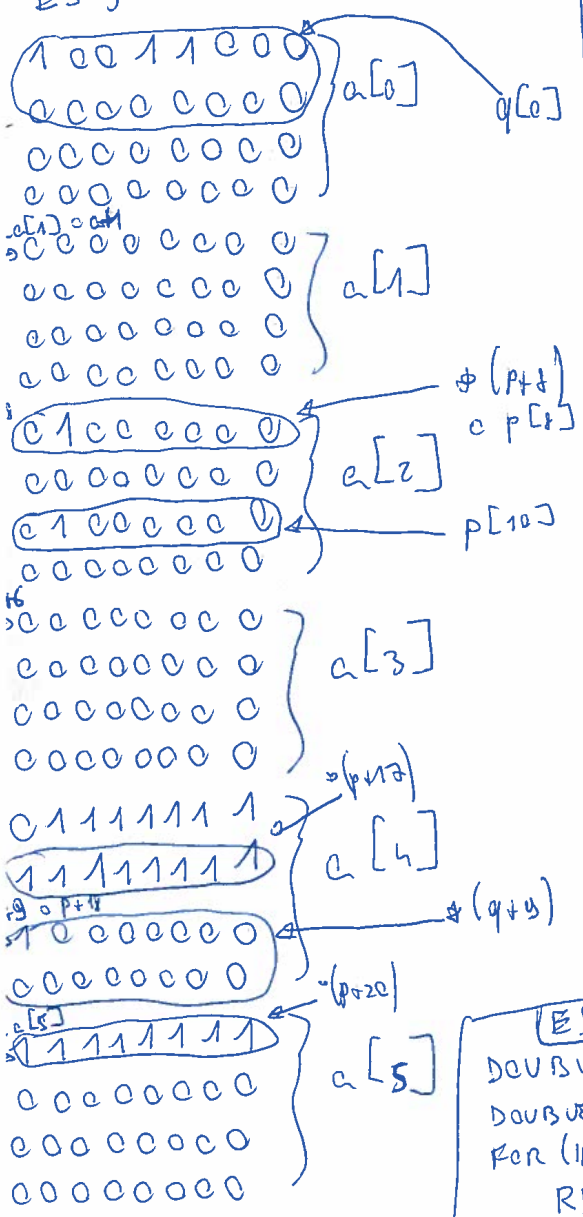
Cerchiare le affermazioni vere dato
*int a[6]= {25,[2]=131074,[4]= 131070, 255}; char *p = (char*) a; short int *q = (short*) a;*
sapendo che i tre tipi usati occupano 4, 1, e 2 byte, e $131072 = 2^{17}$ (valori rappresentati in *little endian*). Autovalutare il risultato programmando l'esempio direttamente.

- $^*(p+8) + q[0] > 24$;
- $p[8] - p[10] > 0$;
- $((int)(q+6) - (int)(\&a[1]) - {}^*(q+9))\%2 == 1$;

- $\&a[5] - (a+1) + {}^*(p+18) <= 5;$
- ${}^*(p+20) - {}^*(p+17) < 0).$
- ${}^*(p+17) >= 0.$

Soluzione dell'esercizio 6

ES 9



A VERA
*(p+8) == 2
q[0] = 25
*(p+8) + q[0] = 27

B FALSA
p[8] == 2 p[8] - p[10] = 0
p[10] == 2

C VERA
SUPPONENDO CHE L'ARRAY È MEMORIZZATO DA INDIRIZZO DI MEMORIA 1000
q+6 == 1012 INT(q+6) - INT(a[1]) = 8
a[1] == 1004
*(q+9) == 1
8-1 == 7 % 2 == 1

D VERA
a[5] - (a+1) == 4 È LA DIFFERENZA IN NUMERO DI ELEMENTI NELL'ARRAY
*(p+12) == 1
4+1 == 5 <= 5

E FALSA
*(p+20) == *(p+12) LA LORO DIFFERENZA È 0

F FALSA
*(p+12) BIT PIÙ SIGNIFICATIVO A 1 -> NUMERO NEGATIVO

ES 8

```
DOUBLE A_ARMONICA (INT* ARRAY, INT N){
DOUBLE RES = 0.0
FOR (INT i = 0; i<N; i++)
    RES += (DOUBLE) 1/ARRAY[i]

RES = N / RES ;
RETURN (RES);
}
```