primo appello
prova scritta

23.1.2001

Correzione di alcuni esercizi

(gli altri sono analoghi a esercizi svolti nelle prove parziali, o desumibili direttamente dalla dispensa)

Comando switch

- Qual è il valore della variabile **x** al termine del seguente comando?

```
int y = 1;
int x = ++y;
switch (x){
  case 1: case 2: x++;
  case 3: case 4: if (x-- == 3) x*=2;
  default:        y *= x;
}
```

- Bisognava aver studiato la semantica del comando switch, per la quale, in assenza di break, tutti i comandi successivi al primo caso soddisfatto vengono eseguiti.
  Quindi la risposta corretta è 4 (ed **y** varrà 8).

Comandi di ripetizione

- Usando solo dichiarazioni, comandi di assegnamento e `while`, scrivere una sequenza di comandi equivalente al seguente comando `for`:

```
int i,j;
int z = 0;
for (i=0, j=10;  i<j;  i++, j--)
   z += i*j;
```

- Risposta:

```
int i=0;
int j=10;
int z=0;
while (i<j){
   z += i*j;
   i++; j--;
}
```

Espressioni booleane

- Scrivere un'espressione booleana che testa se il valore di una variabile x di tipo **char** è una cifra.
- Risposta:

```
(x>='0' && x<='9')
```

- Scrivere un'espressione booleana che testa se il valore di una variabile x di tipo **char** è una lettera compresa tra 'a' e 'z'.
- Risposta:

```
(x>='a' && x<='z')
```

Ricorsione

- Scrivere la definizione di una procedura ricorsiva **`scrivi(int n)`** che dato numero intero positivo n scrive su una stessa riga i numeri da **`0`** a **`4*n`** (estremi inclusi).
- Per ottenere una procedura ricorsiva dobbiamo porci due domande:
  1. Qual'è il caso base? E' 0. E in questo caso il problema si risolve facilmente, scrivendo "0 ".
  2. Se so risolvere il problema per n-1, cosa devo fare per risolverlo per il caso n? Sead esempio n=5, scrivi(n-1) mi scriverebbe 0 1 … 16. Per arrivare fino a 16 basterà aggiungere 17 18 19 20, ovvero scrivere 4*(n-1)+1 … 4*(n-1) +4

Ricorsione

- In conclusione, questo era il codice della procedura richiesta dall'esercizio…

```
void scrivi(int n){
  int i;
  if (n==0)
       printf("0 ");
  else{
       scrivi(n-1);
       for (i=1; i<5; i++)
            printf("%d ", 4*(n-1) + i);
  }
}
```

Ricorsione

- Scrivere la definizione di una procedura ricorsiva `scrivi(int n)` che dato numero intero positivo `n` scrive su una stessa riga i numeri da `-2n` a `2n` (estremi inclusi).
- Risposta

```
void scrivi(int n){
  if (n==0)
       printf("0");
  else{
       printf("%d %d ", -2*n, -(2*n - 1));
       scrivi(n-1);
       printf(" %d %d", 2*n - 1, 2*n );
  }
}
```

Tipi riferimento

- La seguente espressione di inizializzazione è corretta? Perché?

```
double **d = &(*d);
```

- Dal punto di vista dei tipi non ci sono problemi: l'espressione a destra dell'assegnamento è di tipo double**, e verrebbe assegnata a d che è pure di tipo double**. Ma l'inizializzazione NON è corretta, in quanto nel valutare l'espressione a destra dell'assegnamento si dereferenzia la variabile d, alla quale non è ovviamente stato ancora assegnato nessun indirizzo di memoria "valido".

Procedure

- Una procedura che ha tipo di ritorno `int*` può restituire l'indirizzo di una qualsiasi variabile di tipo `int`? Perché?
- No: può restituire solo l'indirizzo di una variabile globale o di una variabile dinamica (allocata nello heap).
  Non avrebbe alcun senso, infatti, che restituisse l'indirizzo di una variabile locale alla procedura, in quanto tale variabile "sparisce" dallo stack all'uscita della chiamata della procedura stessa.

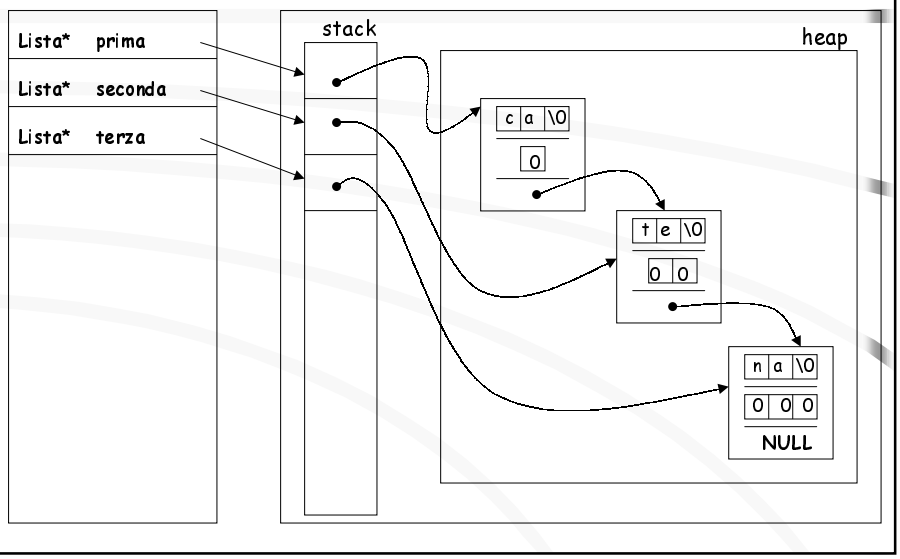
Strutture dinamiche

- Scrivere una serie di dichiarazioni e di comandi che allochino nello heap una lista semplice di 3 elementi. Gli elementi di questa lista devono essere dei record con tre campi: `a`, `b` e `next`. Il campo `a` è di tipo `char*`, e nei tre elementi della lista questo campo deve essere inizializzato, rispettivamente, con la stringa "ca", con la stringa "te" e con la stringa "na". Il tipo del campo `b` è un array di interi, e deve essere inizializzato con array di dimensione 1, 2 e 3, rispettivamente, i cui elementi sono tutti uguali a 0. Il campo `next`, di tipo puntatore, deve puntare al prossimo elemento della lista (a `NULL` nel caso del terzo ed ultimo elemento).

- Risposta:

```
typedef struct Lista{
  char* a;
  int* b;
  struct Lista* c;
} Lista;
Lista *prima, *seconda, *terza;
prima = (Lista*) malloc(sizeof(Lista));
seconda = (Lista*) malloc(sizeof(Lista));
terza = (Lista*) malloc(sizeof(Lista));
prima->a = "ca"; seconda->a = "te"; terza->a ="na";
prima->b = (int*) malloc(sizeof(int)*1);
seconda->b = (int*) malloc(sizeof(int)*2);
terza->b = (int*) malloc(sizeof(int)*3);
(prima->b)[0]=0;
(seconda->b)[0]=0; (seconda->b)[1]=0;
(terza->b)[0]=0; (terza->b)[1]=0; (terza->b)[2]=0;
prima->next = seconda;
seconda->next = terza;
terza->next = NULL;
```

Modello Ambiente-Memoria



Procedure

- Si considerino le seguenti definizioni di costante di procedura:

```
int prima(int a, int b){ return (a*b); }
int seconda(int b, int c){ return (2*b + c); }
```

- Si definisca il tipo `F` al quale appartengono sia `prima` che `seconda`.

- Risposta:

```
typedef int (*F)(int x, int y);
```