Programmazione procedurale e ad oggetti - 5/02/2018

Nome: .....
Cognome: .....
Matricola: .....

1. Illustrate le principali funzioni che il C offre per la gestione dei file.

Sol. Si vedano (dispense o libro di testo) le funzioni `fopen, fclose, fprintf, fscanf, fread, fwrite, fseek, ftell, rewind`.

2. Date le definizioni `int A[10][5]; int i,j;` scrivete un'espressione C che risulti equivalente a `A[i][j]=7` e che non faccia uso dell'operatore `[]`. Spiegatene il significato.

Sol. il nome di un array a 2 dimensioni corrisponde a un puntatore costante inizializzato con l'indirizzo del primo elemento della prima riga (la matrice è vista come un array di array). Un'espressione equivalente a quella indicata è quindi `*(*(A+i)+j)=7`, dove `*(A+i)` rappresenta l'indirizzo del primo elemento della riga di indice `i`, al quale viene sommato `j` per ottenere l'indirizzo `&A[i][j]`.

3. Supponete di utilizzare la seguente struttura per implementare liste concatenate,

```
struct nodo{char c; struct nodo *next}
```

Definite una procedura che riceve in ingresso una stringa `v` (un vettore di caratteri), e restituisce una lista di pari lunghezza contenente gli stessi caratteri nello stesso ordine.

```
struct nodo* StringToList(char str[])
{struct nodo *l, *pt; int i=1;
if(str[0]!='\0')
 {l=(struct nodo*)malloc(sizeof(struct nodo));
  l->c=str[0]; l->next=NULL;}
else return NULL;
pt=l;
while(str[i]!='\0')
 {pt->next=(struct nodo*)malloc(sizeof(struct nodo));
  pt=pt->next;
  pt->c=str[i]; i++;
 }
 pt->next=NULL;
 return l;
}
```

4. Supponete che un int occupi 4 byte e un puntatore 8. Date le seguenti definizioni

```
class A{
    int a=1;
    public:
    int get1(){return this->a;};
    int virtual get2(){return a;};
};
class B: public A{
    public:
    int a=2;
    int virtual get2(){return a;};
};
B bobj;
```

dite cosa viene stampato dalle espressioni

- `std::cout << bobj.get2()`
- `std::cout << bobj.get1()`
- `std::cout << sizeof(B)`

Motivate le risposte.

Sol. I valori stampati sono nell'ordine:

- 2 — `get2` è una funzione virtuale e quindi viene chiamata la versione più vicina alla classe dell'oggetto per cui è invocata. Poiché `bobj` è un oggetto della classe `B` che eredita da `A` e che fornisce una nuova versione di `get2`, viene chiamata quest'ultima, la quale accede al campo `a` della classe `B`;
- 1 — la funzione `get1` non è virtuale ed il suo codice prevede l'accesso al campo `a` di colui per cui è invocata. Il campo `a` in questione è quello che deriva dall'essere un oggetto di classe `A` e quindi viene letto il valore 1;
- 16 — un oggetto di classe `B` possiede due campi interi di nome `a` (`A::a` e `B::a`) e un puntatore per la funzione virtuale `get2`.

5. In C++ qual è la differenza tra un oggetto `x`, un riferimento a `x` e un puntatore a `x`? Supponendo che un oggetto di classe `X` occupi 12 byte e un puntatore 8, quanto spazio occuperebbero complessivamente le tre variabili così definite `X x; X& xref=x; X* xpunt=&x;`?

Sol. Mentre un oggetto occupa una quantità di memoria che dipende dalla classe di cui è istanza, un riferimento ad un oggetto, che equivale alla nozione di alias (ovvero introduce un secondo nome per lo stesso oggetto) non richiede spazio aggiuntivo. Un puntatore occupa sempre e comunque un numero di byte che non dipende dalla dimensione dell'oggetto puntato. Complessivamente, le tre variabili richiederanno quindi 20 byte.

6. Cosa si intende per genericità? È presente in C++?

Sol. Per genericità si intende la possibilità di usare tipi come variabili nella definizione di classi e funzioni. È una delle caratteristiche più interessanti dei linguaggi orientati agli oggetti in quanto favorisce il riutilizzo del software. È presente in C++ attraverso i Template.