PROGRAMMIAMO

C++ - Incapsulamento

Un esempio: programma per il calcolo della differenza fra due orari

Consideriamo ora nuovamente il programma per il calcolo della differenza in minuti fra due orari:

```
#include <cstdlib>
#include <iostream>

using namespace std;

struct orario {
int ora;
int min;
};

int main(int argc, char *argv[])
{
int dif;
orario ora1, ora2;

cout<<"Fornisci l'orario iniziale (ore e minuti): ";
cin>>ora1.ora;
cin>>ora1.min;

cout<<"Fornisci l'orario finale (ore e minuti): ";
cin>>ora2.ora;
cin>>ora2.min;

dif = (ora2.ora*60+ora2.min) - (ora1.ora*60+ora1.min);

cout<<"La differenza fra le "<<ora1.ora<<":"<<ora1.min<<" e le "<<ora2.ora<<":"<<ora2.min<<" e' "<<dif<<"
minuti\n";

system("PAUSE");
return EXIT_SUCCESS;
}
```

Il programma usa le *struct* allo scopo di creare un nuovo tipo di dati aggregato di nome *orario*. In questo modo i dati risultano organizzati in modo più chiaro ed efficace, poiché il programma può trattare il nuovo tipo come uno qualsiasi dei tipi di variabili predefiniti in C.

Per esempio possiamo sviluppare ulteriormente il nostro programma introducendovi alcune funzioni:

```
#include <cstdlib>
#include <iostream>

using namespace std;

struct orario {
int ora;
int min;
};

orario acquisisci();
void visualizza (orario h);
int minuti (orario h);

int main(int argc, char *argv[])
{
int dif;
orario ora1, ora2;

cout<<"Fornisci l'orario iniziale (ore e minuti): ";
ora1 = acquisisci();

cout<<"Fornisci l'orario finale (ore e minuti): ";
ora2 = acquisisci();

dif = minuti(ora2) - minuti(ora1);

cout<<"La differenza fra le ";
visualizza(ora1);
cout<<" e le ";
visualizza(ora2);
cout<<" e' "<<dif<<" minuti\n";

system("PAUSE");
return EXIT_SUCCESS;
}

orario acquisisci()
{
orario h;
cin>>h.ora;
cin>>h.min;
return h;
}

void visualizza (orario h)
{
cout<<h.ora<<":"<<h.min;
}

int minuti (orario h)
{
```

```
 return h.ora*60+h.min;
}
```

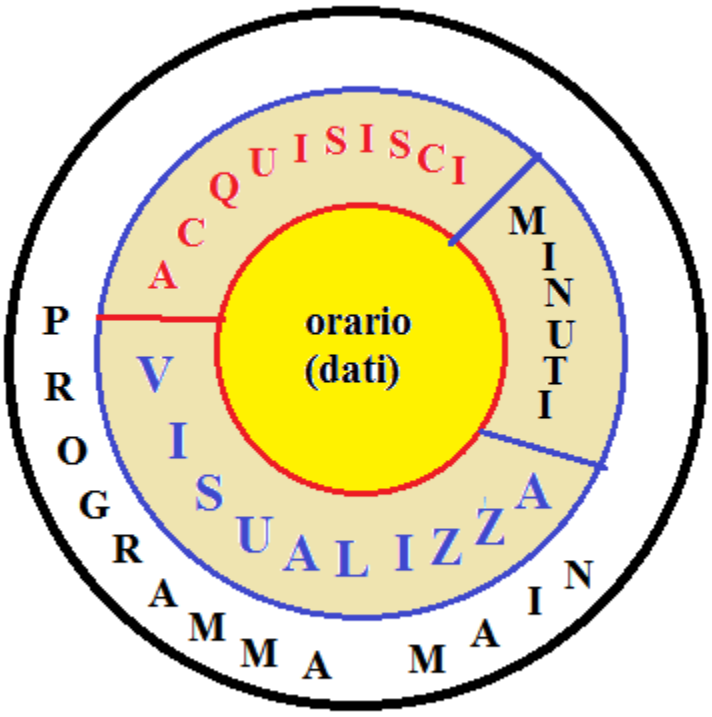
In questa nuova versione del programma, tutte le operazioni sui dati di tipo *orario* sono effettuate per mezzo di funzioni. In particolare le funzioni sono usate per:

- acquisire i dati (funzione *acquisisci*);
- visualizzare i dati (funzione *visualizza*);
- effettuare calcoli sui dati (funzione *minuti*).

Dato che praticamente tutte le operazioni importanti del programma sono svolte da funzioni, il programma principale *main* è diventato molto semplice e lineare. Questa modalità di programmazione, consistente nello svolgere tutte o quasi le operazioni all'interno di funzioni, è effettivamente quella più comunemente utilizzata nella costruzione di programmi complessi.

Incapsulamento (encapsulation)

La tecnica di programmazione in base alla quale il programma non accede direttamente ai dati, ma lo fa usando delle funzioni, si chiama **incapsulamento** (*encapsulation*). Nel nostro esempio, l'incapsulamento può essere rappresentato graficamente nel seguente modo:



Il guscio di funzioni (*acquisisici*, *visualizza*, *minuti*) che racchiude i dati (*orario*) suggerisce appunto l'incapsulamento. L'idea è che il main può accedere ai dati solo passando attraverso le funzioni.

Vantaggi dell'incapsulamento

L'incapsulamento presenta diversi potenziali vantaggi. Il primo è la protezione dei dati nei confronti di modifiche errate o accidentali. Se i dati possono essere modificati solo attraverso funzioni opportunamente testate e progettate, non si corre il rischio di effettuare operazioni scorrette. Questa tecnica è nota anche come **information hiding**, cioè nascondere le informazioni, rendendo disponibile all'esterno solo un accesso limitato e regolato ad esse. Questo problema non appare così significativo nel nostro esempio, ma si pensi per un attimo a una struttura dati più complessa, quale può essere un archivio dati (*database*). In questo caso è essenziale avere la certezza che le operazioni di modifica dell'archivio (es. l'inserimento o la cancellazione di un dato) siano consistenti e corrette.

Un altro vantaggio dell'incapsulamento è che è facile introdurre modifiche nel modo con cui il programma manipola i dati, poiché è sufficiente cambiare solo le corrispondenti funzioni. Per fare un semplice esempio, si supponga di voler modificare il programma in modo che la scrittura dei risultati non venga fatta sul monitor del PC ma su un file sul disco. Questa operazione può essere realizzata nel nostro caso modificando solo la funzione *visualizza* e lasciando immutato il resto del programma.

Infine l'incapsulamento consente facilmente di esportare le proprie strutture dati in altri programmi (anche scritti da altre persone): basta infatti unire ai dati le relative funzioni di accesso ed ecco che si potrà quindi esportare un "pacchetto" già pronto all'uso.

successiva