DIETI UNINA UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

Corso di Laurea in Ingegneria Informatica

Corso di Ingegneria del Software

Prof. Roberto Pietrantuono

Richiami di Programmazione Tecniche di programmazione a oggetti

Problema – 1/2

- Si realizzi un semplice software in grado di gestire un conto corrente.
- È richiesto che:
- Si tenga traccia del saldo del conto corrente;
- Sia possibile effettuare un prelievo dal conto corrente;
- Sia possibile effettuare un versamento sul conto corrente.

Problema – 2/2

- È possibile realizzare tale software facendo uso di diverse tecniche di programmazione.
- Si illustrano le tecniche di programmazione:

- Con oggetti;
- Con tipi di dati astratti (ADT);
- Con tipi strutturati;
- Con tipi strutturati con metodi;
- Con classi.

Programmazione con oggetti – 1/5

- Se ho bisogno di un solo oggetto può essere superflua la definizione di un tipo di dati
- È possibile realizzare l'oggetto contocorrente in C/C++ utilizzando:
  - Compilazione separata
  - File di intestazione
  - Prototipi di funzioni
  - Variabili membro protette

Programmazione con oggetti – 2/5

- Il file di intestazione contiene solo le dichiarazioni dei metodi sull'oggetto
- Le variabili che costituiscono la struttura dati concreta dell'oggetto sono incapsulate nel modulo e protette con il qualificatore `static`

main.cpp

```
//Modulo utilizzatore dell'
//oggetto contocorrente

#include "contocorrente.h"
```

contocorrente.h

```
//Interfaccia dell'oggetto
//contocorrente
```

contocorrente.cpp

```
//Implementazione
//dell'oggetto contocorrente

#include "ContoCorrente.h"
```

Programmazione con oggetti – 3/5

```
//File contocorrente.h
//Specifica dell'oggetto. Dichiarazioni di variabili e metodi
using namespace std;  // Da usare in C++


void inizializza();

void prelievo(const int);

void versamento(const int);

int getSaldo();

void setSaldo(const int);
```

Programmazione con oggetti – 4/5

```
//File contocorrente.cpp
//Implementazione dell'oggetto. Definizioni di variabili e metodi
#include "contocorrente.h"

static int saldo;     //struttura dati dell'oggetto,
                      //incapsulata e protetta
void inizializza(){
     saldo=0;}

void prelievo(const int importo){
       saldo -= importo;}

void versamento(const int importo){
       saldo += importo;}

int getSaldo(){
    return saldo;}

void setSaldo(const int importo){
    saldo=importo;}
```

Programmazione con oggetti – 5/5

```
//File main.cpp
//Programma di prova per l'oggetto contocorrente
#include <iostream>
#include "contocorrente.h"

int main(int argc, char *argv[]){
      inizializza();
      setSaldo(10000);
      cout<<"Saldo al 01/01/2011: "<<getSaldo()<<endl;
      prelievo(1000);
      cout<<"Saldo al 01/02/2011: "<<getSaldo()<<endl;
      prelievo(3500);
      cout<<"Saldo al 15/03/2011: "<<getSaldo()<<endl;
      versamento(5000);
      cout<<"Saldo al 05/04/2011: "<<getSaldo()<<endl;
      system("PAUSE");
      return EXIT_SUCCESS;
}
```

Programmazione con ADT – 1/5

- Altra possibile realizzazione fa uso del Tipo di Dato Astratto (*Abstract Data Type*, ADT)
- Si vuole definire il tipo ContoCorrente
  - Occorre un costruttore di tipo
    - Non necessariamente devono essere utilizzati record o classi
    - Si può usare il costruttore di tipo `typedef`

main.cpp

```
//Modulo utilizzatore del
//Tipo di Dato Astratto
//ContoCorrente

#include "ContoCorrente.h"
…
```

ContoCorrente.h

```
//Interfaccia del Tipo di Dato
//Astratto ContoCorrente
#ifndef CONTOCORRENTE_H_
#define CONTOCORRENTE_H_

…
#endif /* CONTOCORRENTE_H_ */
```

ContoCorrente.cpp

```
//Implementazione del Tipo di
//Dato Astratto ContoCorrente

#include "ContoCorrente.h"
…
```

Programmazione con ADT – 2/5

- È possibile realizzare l'ADT ContoCorrente utilizzando:
  - Compilazione separata
  - File di intestazione (l'intestazione contiene solo le dichiarazioni del tipo e delle operazioni)
  - Prototipi di funzioni
  - Un costruttore di tipo (`typedef`)
  - Compilazione condizionale (mediante direttive al preprocessore per evitare definizioni multiple)

Programmazione con ADT – 3/5

```
//File ContoCorrente.h
//Dichiarazione del ADT ContoCorrente e dei metodi

using namespace std;

#ifndef CONTOCORRENTE_H_
#define CONTOCORRENTE_H_

typedef int ContoCorrente; //Definizione del nome di tipo

//Dichiarazioni delle operazioni dell'ADT
void inizializza(ContoCorrente&);
void prelievo(ContoCorrente&, const int);
void versamento(ContoCorrente&, const int);
int getSaldo(const ContoCorrente&);
void setSaldo(ContoCorrente&, const int);

#endif /* CONTOCORRENTE_H_ */
```

Programmazione con ADT – 4/5

```
//File ContoCorrente.cpp
//Implementazione dell'ADT ContoCorrente. Definizione dei metodi
#include "ContoCorrente.h"

void inizializza(ContoCorrente& saldo){
     saldo=0;}

void prelievo(ContoCorrente& saldo, const int importo){
       saldo -= importo;}

void versamento(ContoCorrente& saldo, const int importo){
       saldo += importo;}

int getSaldo(const ContoCorrente& saldo){
    return saldo;}

void setSaldo(ContoCorrente& saldo, const int importo){
    saldo=importo;}
```

Programmazione con ADT – 5/5

```
//File main.cpp
//Programma di prova dell'ADT ContoCorrente
#include <iostream>
#include "ContoCorrente.h"
int main(int argc, char *argv[]){
       ContoCorrente cc; //Creazione di un oggetto
       inizializza(cc);  //Uso dell'oggetto
       setSaldo(cc, 10000);
       cout<<"Saldo al 01/01/2011: "<<getSaldo(cc)<<endl;
       prelievo(cc, 1000);
       cout<<"Saldo al 01/02/2011: "<<getSaldo(cc)<<endl;
       prelievo(cc, 3500);
       cout<<"Saldo al 15/03/2011: "<<getSaldo(cc)<<endl;
       versamento(saldo, 5000);
       cout<<"Saldo al 05/04/2011: "<<getSaldo(cc)<<endl;
       system("PAUSE");
       return EXIT_SUCCESS;
}
```

Programmazione di un ADT con tipo strutturato – 1/4

- Similmente si può procedere utilizzando un record contenente la variabile saldo.
- Sono definiti anche metodi di accesso e assegnazione di valore ai dati del tipo strutturato.

main.cpp

```
// Modulo utilizzatore del
//TDA ContoCorrente

#include "ContoCorrente.h"
```

ContoCorrente.h

```
//Interfaccia del TDA con
//struct ContoCorrente
#ifndef CONTOCORRENTE_H_
#define CONTOCORRENTE_H_

…
#endif /* CONTOCORRENTE_H_ */
```

ContoCorrente.cpp

```
//Implementazione TDA con
//struct ContoCorrente

#include "ContoCorrente.h"
```

Programmazione di un ADT con tipo strutturato – 2/4

```
//File ContoCorrente.h
//Dichiarazione della struct ContoCorrente e dei metodi

using namespace std;

#ifndef CONTOCORRENTE_H_
#define CONTOCORRENTE_H_

struct ContoCorrente{
       int saldo;
};
void inizializza(ContoCorrente&);
void prelievo(ContoCorrente&, const int);
void versamento(ContoCorrente&, const int);
void setSaldo(ContoCorrente&, const int);
int getSaldo(const ContoCorrente&);

#endif /* CONTOCORRENTE_H_ */
```

Programmazione di un ADT con tipo strutturato – 3/4

```
//File ContoCorrente.cpp
//Definizione dei metodi
#include "ContoCorrente.h"

void inizializza(ContoCorrente& conto){
     conto.saldo=0;}

void prelievo(ContoCorrente& conto, const int importo){
       conto.saldo -= importo;}

void versamento(ContoCorrente& conto, const int importo){
       conto.saldo += importo;}

void setSaldo(ContoCorrente& conto, const int importo){
     conto.saldo=importo;}

int getSaldo(const ContoCorrente& conto){
    return conto.saldo;}
```

Programmazione di un ADT con tipo strutturato – 4/4

```
//File main.cpp
//Programma di prova della struct ContoCorrente
#include <iostream>
#include "ContoCorrente.h"
int main(int argc, char *argv[]){
      ContoCorrente cc;  //Creazione di un oggetto
      inizializza(cc);   //Uso dell'oggetto
      setSaldo(cc, 10000);
      cout<<"Saldo al 01/01/2011: "<<getSaldo(cc)<<endl;
      prelievo(cc, 1000);
      cout<<"Saldo al 01/02/2011: "<<getSaldo(cc)<<endl;
      prelievo(cc, 3500);
      cout<<"Saldo al 15/03/2011: "<<getSaldo(cc)<<endl;
      versamento(cc, 5000);
      cout<<"Saldo al 05/04/2011: "<<getSaldo(cc)<<endl;
      system("PAUSE");
      return EXIT_SUCCESS;
}
```

Programmazione di un ADT con tipo strutturato con metodi – 1/4

- Si può procedere utilizzando una `struct` con variabile privata, metodi set/get e i metodi richiesti di prelievo e versamento.

main.cpp

```
// Modulo utilizzatore del
//TDA con struct ContoCorrente

#include "ContoCorrente.h"
```

ContoCorrente.h

```
//Interfaccia del TDA con
//struct ContoCorrente
```

ContoCorrente.cpp

```
//Implementazione TDA con
//struct ContoCorrente

#include "ContoCorrente.h"
```

Programmazione di un ADT con tipo strutturato con metodi – 2/4

```
//File ContoCorrente.h
//Dichiarazione della struct ContoCorrente e dei metodi
using namespace std;

#ifndef CONTOCORRENTE_H_
#define CONTOCORRENTE_H_

struct ContoCorrente{
    void inizializza();
    void prelievo(const int);
    void versamento(const int);
    void setSaldo(const int);
    int getSaldo();
private:
       int saldo;
};

#endif /* CONTOCORRENTE_H_ */
```

Programmazione di un ADT con tipo strutturato con metodi – 3/4

```
//File ContoCorrente.cpp
//Definizione dei metodi
#include "ContoCorrente.h"

void ContoCorrente::inizializza(){
     saldo=0;}

void ContoCorrente::prelievo(const int importo){
       saldo -= importo;}

void ContoCorrente::versamento(const int importo){
       saldo += importo;}

void ContoCorrente::setSaldo(const int importo){
     saldo=importo;}

int ContoCorrente::getSaldo(){
    return saldo;}
```

Programmazione di un ADT con tipo strutturato con metodi – 4/4

```
//File main.cpp
//Programma di prova della struct ContoCorrente
#include <iostream>
#include "ContoCorrente.h"

int main(int argc, char *argv[]){
	ContoCorrente cc;
	cc.inizializza();
	cc.setSaldo(10000);
	cout<<"Saldo al 01/01/2011: "<<cc.getSaldo()<<endl;
	cc.prelievo(1000);
	cout<<"Saldo al 01/02/2011: "<<cc.getSaldo()<<endl;
	cc.prelievo(3500);
	cout<<"Saldo al 15/03/2011: "<<cc.getSaldo()<<endl;
	cc.versamento(5000);
	cout<<"Saldo al 05/04/2011: "<<cc.getSaldo()<<endl;
	system("PAUSE");
	return EXIT_SUCCESS;
}
```

Programmazione con classi – 1/4

- Infine, è possibile definire il TDA ContoCorrente attraverso l'utilizzo del costrutto classe ContoCorrente.

main.cpp

```
//Modulo utilizzatore della
//classe ContoCorrente

#include "ContoCorrente.h"
```

ContoCorrente.h

```
//Interfaccia della classe
//ContoCorrente
```

ContoCorrente.cpp

```
//Implementazione della classe
//ContoCorrente

#include "ContoCorrente.h"
```

Programmazione con classi – 2/4

```
//File ContoCorrente.h
//Dichiarazione della classe ContoCorrente

using namespace std;

class ContoCorrente {
 public:
    ContoCorrente(int);
    void prelievo(const int);
    void versamento(const int);
    int getSaldo();
 private:
    int saldo;
};
```

Programmazione con classi – 3/4

```
//File ContoCorrente.cpp
//Realizzazione della classe ContoCorrente

#include "ContoCorrente.h"

ContoCorrente::ContoCorrente(int s):saldo(s){ };

void ContoCorrente::prelievo(const int importo){
       saldo -= importo;
}

void ContoCorrente::versamento(const int importo){
       saldo += importo;
}

int ContoCorrente::getSaldo(){return saldo;}
```

Programmazione con classi – 4/4

```
//File main.cpp
//Programma di prova della classe ContoCorrente
#include <iostream>
#include "ContoCorrente.h"

int main(int argc, char *argv[]){

      ContoCorrente cc(10000);
      cout<<"Saldo al 01/01/2011: "<<cc.getSaldo()<<endl;
      cc.prelievo(1000);
      cout<<"Saldo al 01/02/2011: "<<cc.getSaldo()<<endl;
      cc.prelievo(3500);
      cout<<"Saldo al 15/03/2011: "<<cc.getSaldo()<<endl;
      cc.versamento(5000);
      cout<<"Saldo al 05/04/2011: "<<cc.getSaldo()<<endl;
      system("PAUSE");
      return EXIT_SUCCESS;
}
```