

**Università di degli Studi della Campania**  
**Luigi Vanvitelli**  
**Dipartimento di Ingegneria**

**Programmazione ad Oggetti**

*a.a. 2020-2021*

**C++**

Docente: Prof. Massimo Ficco  
E-mail: [massimo.ficco@unicampania.it](mailto:massimo.ficco@unicampania.it)

1

1

## Il linguaggio C++



- C++ è un linguaggio di programmazione general-purpose che supporta:
  - la programmazione procedurale (è un C “migliore”);
  - la programmazione orientata agli oggetti;
  - la programmazione generica.
- C++ è quindi un linguaggio ibrido, nel senso che supporta più paradigmi di programmazione



2

# Programmazione generica V:

La *programmazione generica* consiste nella creazione di costrutti di programmazione che possano essere utilizzati con molti tipi di dati diversi

Ad esempio, la classe `ArrayList` sfrutta le tecniche della programmazione generica: come risultato, è possibile creare vettori che contengano elementi di tipi diversi



# Programmazione con oggetti V:

Esempio con programmazione con oggetti tramite l'utilizzo del costrutto **struct** e **funzioni c**.

```
struct stack {  
    int TOS;  
    data * buffer;  
    int size;  
};  
  
boolean push(struct stack * ptr, data value) {  
    ....  
}  
  
boolean push(struct stack * ptr, data value) {  
    ....  
}
```



# Programmazione orientata agli oggetti

## Le classi in C++

- Il linguaggio C++ supporta esplicitamente la dichiarazione e la definizione di tipi astratti da parte dell'utente mediante il costrutto `class`; le istanze di una classe vengono dette oggetti.
- In una dichiarazione `class` occorre specificare sia la struttura dati che le operazioni consentite su di essa. Una classe possiede, in generale, una sezione pubblica ed una privata.
- La sezione pubblica contiene tipicamente le operazioni (dette anche metodi) consentite ad un utilizzatore della classe. Esse sono tutte e sole le operazioni che un utente può eseguire, in maniera esplicita od implicita, sugli oggetti.
- La sezione privata comprende le strutture dati e le operazioni che si vogliono rendere inaccessibili dall'esterno.



## Definizione di una classe C++

### Esempio classe in c++

```
class stack
{ int TOS;
  data * buffer;
  int size;

  boolean push(data value);
  boolean pop(data * value);
};

boolean stack::push(data value) {
  .....
}

boolean stack::pop(data * value) {
  .....
}
```



# Implementazione di una classe V:

Implementazione di un metodo:

```
tipo_restituito nome_classe :: nome_metodo (parametri) {...}
```

dove l'operatore `::` viene denominato operatore di **scope**

```
oggetto.attributo;  
oggetto.metodo();
```



## Visibilità

# V:

```
class Contatore {  
    public:  
        void Incrementa(); // operazione incremento  
        void Decrementa(); // operazione decremento  
        unsigned int give_value(); // restituisce il valore  
                                   // corrente  
    private:  
        unsigned int value; // valore corrente  
        const unsigned int max; // valore massimo  
};
```

Interfaccia {

Implementazione {



# Creazione

V:

## Esempio con **struct c**:

```
void client (void) {  
    .....  
    stack s; // variabile struct– viene allocata staticamente nello stack e  
             // ha il tempo di vita della funzione in cui è dichiarata  
    .....  
}
```

## Esempio con **classi c++**:

```
void main (void) {  
    stack *sptr;  
    .....  
    sptr = new stack; // istanza di un oggetto – viene allocata dinamicamente nella  
                     // area heap e distrutte solo esplicitamente tramite il costrutto  
                     // DELETE o automaticamente alla fine del programma  
    .....  
    delete sptr;  
}
```



# Costruttori e Distruttori

V:

- **new**, che alloca la memoria necessaria all'istanziamento dell'oggetto e ne ritorna la relativa locazione di memoria.
- **delete**, che servirà per liberare la memoria utilizzata per l'oggetto, una volta che non ci servirà più



# Uso dei puntatori

V:

*Nome\_Classe \*nome\_Oggetto;*

Avendo a che fare con un puntatore all'oggetto, cambierà anche la modalità con la quale facciamo riferimento ai suoi metodi o ai suoi attributi (o proprietà).

Invece del punto utilizziamo l'operatore freccia (->)

**Riferimento statico**

**oggetto.attributo;**

**oggetto.metodo();**

**Puntatore all'oggetto**

**oggetto->attributo;**

**oggetto->metodo();**

```
void main (void) {  
    stack S;  
    stack *Sptr;  
    ....  
    Sptr=&S;  
    ....  
    S.push(13);  
    ....  
    Sptr->push(13);  
}
```



# Implementazione di una classe V:

Potremmo scrivere la definizione dell'interfaccia e l'implementazione dei metodi direttamente nello stesso file, ma è buona norma di inserire la definizione di una classe (l'interfaccia) in un file detto di header (intestazione, con estensione **.h**) e le implementazioni dei metodi della classe in file con estensione **.cpp**.



## Produzione e uso di una classeV:

Utente.cpp	Contatore.h	Contatore.cpp
<pre>// Modulo utilizzatore del modulo // Contatore #include "Contatore.h"  unsigned int i; Contatore cont1, cont2;  . . .  cont1.incrementa(); cont2.decrementa();  i = cont1.get_value();</pre>	<pre>// Interfaccia del // modulo Contatore  class Contatore { . . . };</pre>	<pre>// Implementazione del modulo Contatore  #include "Contatore.h"  Contatore::incrementa() { . . }  Contatore::decrementa() { . . }</pre>

## Produzione e uso di una classeV:

Cliente.h

```
// Semplice esempio di una classe C++

class Cliente
{
public:
    char nome[20];
    char cognome[20];
    char indirizzo[30];
    void inserisci_nome( );
    void inserisci_cognome( );
    void inserisci_indirizzo( );
};
```

# Produzione e uso di una classeV:

cliente.cpp

```
#include <iostream.h>
#include "cliente.h"

void Cliente::inserisci_nome( )
{
    cout << "Inserire il nome del dipendente: ";
    cin >> nome;
    cout << endl;
}

void Cliente::inserisci_cognome( )
{
    cout << "Inserire il cognome del dipendente: ";
    cin >> cognome;
    cout << endl;
}

void Cliente::inserisci_indirizzo( )
{
    cout << "Inserire l' indirizzo del dipendente: ";
    cin >> indirizzo;
    cin >> get(newline); //elimina il Carriage Return
}

main()
{
```



15

# Produzione e uso di una classeV:

cliente.cpp

```
main()
{
    Cliente* cliente;

    cliente = new Cliente( );

    cliente->inserisci_nome( );
    cliente->inserisci_cognome( );
    cliente->inserisci_indirizzo( );

    cout << "Il nome del cliente inserito è: " << cliente->nome << endl;
    cout << "Il cognome del cliente inserito è: " << cliente->cognome << endl;
    cout << "L' indirizzo del cliente inserito è: " << cliente->indirizzo << endl;

    delete cliente;
}
```



16