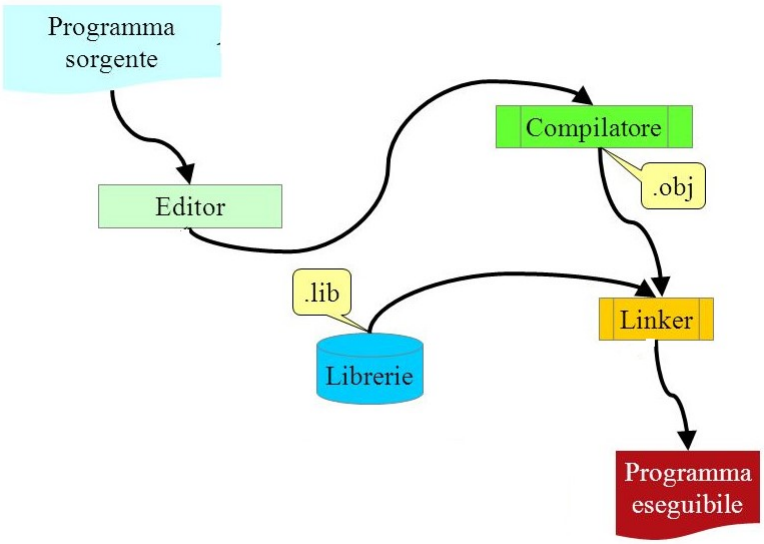
ITIS-LS "Francesco Giordani" Caserta

prof. Ennio Ranucci

a.s. 2019-2020

Namespace e creazione libreria C++

Esercitazioni in ambiente code::Blocks versione 17.12



Quando abbiamo due persone con lo stesso nome per differenziarli usiamo alcune informazioni aggiuntive insieme al loro nome, per esempio la zona in cui vivono.
Stessa situazione può verificarsi nelle applicazioni C++.
Ad esempio, potreste scrivere codice che ha una funzione chiamata xyz() e un'altra libreria disponibile contiene la stessa funzione xyz().
Uno spazio dei nomi è utile per superare questa ambiguità e viene utilizzato come informazione aggiuntiva per differenziare funzioni, classi, variabili che hanno lo stesso nome in librerie diverse.
L'operatore di risoluzione di scope, identificato dai caratteri :: ci consente di accedere ai simboli definiti entro i rispettivi namespace.

```
#include <iostream>

using namespace std;

namespace primoSpazio {

  void stampaNomeSpazio() {

    cout << "Nome spazio: primoSpazio" << endl;

  }

}

namespace secondoSpazio {

  void stampaNomeSpazio() {

    cout << "Nome spazio: secondoSpazio" << endl;

  }

}

int main () {

  primoSpazio::stampaNomeSpazio();

  secondoSpazio::stampaNomeSpazio();

  return 0;

}
```

La direttiva using namespace indica al compilatore che il codice successive utilizza i nomi nello spazio dei nomi specificato.

```
#include <iostream>

using namespace std;

namespace primoSpazio {
  void stampaNomeSpazio() {
    cout << "Nome spazio: primoSpazio" << endl;
  }
}


namespace secondoSpazio {
  void stampaNomeSpazio() {
    cout << "Nome spazio: secondoSpazio" << endl;
  }
}
using namespace primoSpazio;
int main () {
  stampaNomeSpazio();
  return 0;
}
```

Creazione di librerie

Suddivisione del codice in più file

Tutti i programmi che abbiamo visto finora erano composti da un unico file .cpp
Nel caso di programmi più grandi è conveniente suddividere il codice in più file:
– migliore suddivisione e organizzazione del codice
– facilità di manutenzione e correzione degli errori
– possibilità di riutilizzare il codice per progetti diversi
Il linguaggio C++ fornisce tre strade per suddividere il codice:
1. inclusione diretta di file sorgente e compilazione unica;
2. inclusione di header file e compilazione separata;
3. creazione di librerie statiche o dinamiche.

Inclusione diretta di file sorgente

- Effettuata dal preprocessore mediante la direttiva #include “myLib.cpp”

Prima della compilazione il preprocessore inserisce il codice contenuto in myLib.cpp nel programma.
È il metodo più semplice ma meno efficiente, la modifica di uno solo dei file richiede la ricompilazione completa di tutto il programma.

File inclusioneDiretta.cpp

```
#include <iostream>
#include <myLib.cpp>
using namespace std;
int main()
{
   int num1=5, num2=10;
   cout <<"somma di "<<num1<<" "<<num2<<" = "<< somma(num1,num2) << endl;
   return 0;
}
```

File myLib.cpp da copiare nella directory “include” (C:\Program Files (x86)\CodeBlocks\MinGW\include)

```
int somma(int num1Par,int num2Par)
{
  return num1Par+num2Par;
}
int moltiplica(int num1Par,int num2Par)
{
  return num1Par+num2Par;
}
```

Header file e compilazione separata

Anziché includere tutto il sorgente si può includere un header file, cioè un file con estensione .h che contiene solamente inclusioni di altri header file, definizione di strutture e di tipi (typedef), prototipi di funzioni.
Si usa la direttiva #include "myLib.h"

- Il codice delle funzioni va scritto in un file .cpp separato che deve includere l'header
- Compilazione e linking devono essere fatti separatamente
- La modifica di uno dei file richiede di ricompilare il file oggetto corrispondente e di rifare il linking.

Compilazione non separata

Creare il file "myLib.h" il file "mylib.cpp" e il file "main.cpp".
Creare il progetto "Header e compilazione non separata.cbp" scegliere "Project" ->Add file ed aggiungere myLib.h e myLib.cpp (che devono essere memorizzata nella cartella del progetto)
Compilare ed eseguire

File main.cpp

```
#include <iostream>
#include "myLib.h"
using namespace std;
int main()
{
   int num1=5, num2=3;
   cout <<"somma di "<<num1<<" "<<num2<<" = "<< somma(num1,num2) << endl;
   return 0;
}
```

File myLib.cpp

```
#include "myLib.h"
int somma(int num1Par,int num2Par)
{
  return num1Par+num2Par;
}
int moltiplica(int num1Par,int num2Par)
{
  return num1Par+num2Par;
}
```

File myLib.h

```
int somma(int num1Par,int num2Par);
int moltiplica(int num1Par,int num2Par);
```

Compilazione separata della libreria

Aprire un nuovo progetto denominato "Cancellazione separata" scegliendo l'icona "Static Library"
nel main (rinominare il file main.c in main.cpp)scrivere il seguente codice:

```
#include "myLib.h"
int somma(int num1Par,int num2Par)
{
  return num1Par+num2Par;
}
int moltiplica(int num1Par,int num2Par)
{
  return num1Par+num2Par;
}
```

Aggiungere al progetto il file “myLib.h” selezionando “Project”->”add files”e compilare mediante la voce BUILD. Chiudere il progetto.
Nella cartella bin/debug troverete il file: libcompilazione separata.a che è la libreria appena creata.

Aprire un nuvo progetto “console application” denominato usaLibreriaStatica e scrivere il seguente codice nel main:

```
#include <iostream>
#include "myLib.h"
using namespace std;
int main()
{
   int num1=5, num2=3;
   cout <<"somma di "<<num1<<" "<<num2<<" = "<< somma(num1,num2) << endl;
   return 0;
}
```

Aggiungere la libreria: myLib.h selezionando la voce “add files” nel menu “Project”
myLib.h contiene il seguente codice:

```
int somma(int num1Par,int num2Par);
int moltiplica(int num1Par,int num2Par);
```

A questo punto dobbiamo aggiungere il percorso per trovare la nuova libreria che abbiamo creato:
Copiamo la libreria libcompilazione separata.a nella cartella del progetto “usaLibreriaStatica” e selezioniami “settings”->”Compiler”->”Linker setting” ->”Add”->selezioniamo la nostra libreria libcompilazione separarta.a->”OK”
Compiliamo ed eseguiamo il progetto.

Le librerie

_ La fase di Linking è l'ultimo passaggio nella creazione di un eseguibile:
– aggiunge ai file oggetto il codice delle librerie esterne necessarie per il programma
_ Il linking può essere di due tipi:
– Linking statico: l'eseguibile contiene una copia del codice delle librerie;
– Linking dinamico: l'eseguibile non contiene il codice, ma solo un riferimento al file della libreria.
_ Se l'eseguibile è stato "linkato dinamicamente", le librerie richieste devono essere presenti ed "installate" nel sistema per poterlo eseguire.
_ Se l'eseguibile è "linkato staticamente", la presenza delle librerie non è necessaria.

Linking statico vs. linking dinamico

Linking statico
Vantaggi:
– non necessita della presenza delle librerie
– è più veloce nel caricamento del programma
Svantaggi:
– maggior consumo di memoria: se più programmi usano la stessa libreria, essa viene caricata in memoria più volte
Linking dinamico
Vantaggi:
– minor consumo di memoria: se più programmi usano la stessa libreria, essa viene caricata in memoria una volta sola.
Svantaggi:
– necessita della presenza delle librerie nel sistema
– caricamento del programma più lento

Librerie statiche e Librerie dinamiche (dll) – ulteriori differenze

La differenza fra le 2 librerie e che la libreria dinamica, Dynamic Link Library (librerie condivise) sono dei software che vengono caricati dinamicamente in fase di esecuzione invece di essere collegate staticamente ad un eseguibile in fase di compilazione, la separazione del codice in librerie dinamiche permette di spezzare i programmi in parti separate e queste parti verranno caricate solo se effettivamente e necessario e inoltre una singola libreria può essere caricata in memoria una sola volta ed utilizzata da più programmi è questo consente di risparmiare risorse, questo metodo di caricamento viene chiamato ***Loading Demand*** *(caricamento su richiesta), consente inoltre di installare parzialmente dei sistemi software perché alcune parti che sono le nostre librerie potrebbero essere già presenti nella memoria di massa e quindi non richiede la reinstallazione. La libreria statica consente invece di collegare questo software che contiene le nostre funzioni staticamente ad un eseguibile, questo collegamento avviene in fase di compilazione.*

Creare una libreria dinamica

Una ***dynamic-link library*** (**collegamento dinamico**) indica una libreria software che viene caricata dinamicamente in fase di esecuzione, invece di essere collegata staticamente a un eseguibile in fase di compilazione. Queste librerie sono note con l'acronimo **DLL**, che è l'estensione del file che hanno nel sistema operativo Microsoft Windows, o anche con il termine **librerie condivise** (da *shared library*, usato nella letteratura dei sistemi Unix).
Per la creazione di librerie dinamiche Codeblocks mette a disposizione "Shared Library".
Dopo averlo selezionato aggiungere la libreria: myLib.h selezionando la voce "add files" nel menu "Project"
myLib.h contiene il seguente codice:

```
int somma(int num1Par,int num2Par);
int moltiplica(int num1Par,int num2Par);
```

Scrivere nel main il seguente codice:

```
#include "myLib.h"
extern "C++"
int somma(int num1Par,int num2Par)
{
  return num1Par+num2Par;
}
int moltiplica(int num1Par,int num2Par)
{
  return num1Par+num2Par;
}
```

BUILD. Chiudere il progetto.
Nella cartella bin/debug troverete i files: liblibreriaDinamica.def, liblibreriaDinamica.dll, liblibreriaDinamica.a
Aprire un nuvo progetto “console application” denominato usaLibreriaDinamica e scrivere il seguente codice nel main:

```
#include <iostream>
#include "myLib.h"
using namespace std;
int main()
{
   int num1=5, num2=3;
   cout <<"somma di "<<num1<<" "<<num2<<" = "<< somma(num1,num2) << endl;
   return 0;
}
```

Aggiungere la libreria: myLib.h selezionando la voce “add files” nel menu “Project”
myLib.h contiene il seguente codice:

```
int somma(int num1Par,int num2Par);
int moltiplica(int num1Par,int num2Par);
```

A questo punto dobbiamo aggiungere il percorso per trovare la nuova libreria che abbiamo creato: Copiamo la libreria libcompilazione separata.a nella cartella del progetto “usaLibreriaDinamica” e selezioniami “settings”->”Compiler”->”Linker setting” ->”Add”->selezioniamo la nostra libreria liblibreriaDinamica.dll ->”OK” Copiamo la libreria dinamica liblibreriaDinamica.dll nella cartella principale del progetto oppure in bin/debug
Compiliamo ed eseguiamo il progetto. Se eliminiamo il file dll la compilazione termina con successo mentre l’esecuzione si interrompe.