Aula 9

Compilação e Execução

Prof. Sandino Jardim / CC-UFMT-CUA

Separando definição de implementação

A definição de classes em C++ é comumente dividida em dois arquivos separados (e ambos fora do arquivo de utilização destas):

- A "descrição" da classe, com seus atributos e protótipos dos métodos costuma vir num arquivo .h
- A implementação dos métodos vem num arquivo .cpp com o nome da classe

Benefícios:

- Se a implementação da classe não mudar, não precisa ser recompilada
- Separação entre implementação e utilização explícita
- Organização do código

Uso do #ifndef

1

Algumas vezes podemos incluir um arquivo de cabeçalho múltiplas vezes

2

O compilador pode entender como se estivéssemos tentando redefinir alguma coisa novamente

3

Solução: informar ao compilador (diretiva) que a biblioteca não deve ser redefinida se já o foi

Exemplo – Sem `#ifndef`

```
#include "Num.h"
class Foo {
 public:
  Num n;
};
```

```
#include <iostream>
#include "Num.h"
#include "Foo.h"
using namespace std;
int main() {
  Num n(35);
  cout << n.getNum() << endl;
  Foo f;  cout << f.n.getNum() << endl;
  return 0;
 }
```

```
In file included from Foo.h:1:0,
 from main.cpp:3:
Num.h:1:7: error: redefinition of ‘class Num’
In file included from main.cpp:2:0:
Num.h:1:7: error: previous definition of ‘class Num’
main.cpp: In function ‘int main()’:
main.cpp:13:13: error: ‘class Foo’ has no member named ‘num’
```

Separando de fato

Compile cada arquivo .cpp num arquivo object (.o) que contém o código de máquina para aquele arquivo:

```
g++ -c main.cpp Num.cpp
```

Realize o link entre o(s) arquivo(s) e o executável:

```
g++ main.o Num.o
```

Se apenas o executável mudar, compile apenas ele:

```
g++ -c main.cpp
```

Depois, linke-os novamente:

```
g++ main.o Num.o
```

Criando arquivo make

Arquivo que define as dependências do programa e o que precisa ser compilado

Processo automatizado por IDE's

Quando invocado, compila apenas o que foi modificado

Permite a limpeza dos arquivos de saída para recompilação

Arquivos - Exemplo

Time.cpp

```cpp
#include <iostream>
#include <iomanip>
#include "Time.h"

using namespace std;
static int x = 10;
namespace Tempo{
Time :: Time(const int h, const int m, const int s)
: hour(h), minute (m), second(s) { }

void Time :: setTime(const int h, const int m, const int s)
{
      hour = h;
      minute = m;
      second = s;
}
/*\...*/
```

Time.h

```cpp
#ifndef TIME_H
#define TIME_H
namespace Tempo{
class Time
{
private :
      int hour;
      int minute;
      int second;
public :
      //with default value
      Time(const int h = 0, const int m = 0, const int s =
      0);
      // setter function
      void setTime(const int h, const int m, const int s);
      // Print a description of object in " hh:mm:ss"
      void print() const;
      //compare two time object
      bool equals(const Time&);
};

#endif
```

main.cpp

```cpp
#include <iostream>
#include "Time.h"

using namespace std;
int main()
{
      int x = 20;
      using Tempo::Time;
      Time t1(20, 50, 59);
      t1.print(); // 10:50:59
      Time t2;
      t2.print(); // 06:39:09
      t2.setTime(8, 39, 9);
      t2.print(); // 06:39:09

      if(t1.equals(t2))
            cout << "Two objects are equal\n";
            else
            cout << "Two objects are not equal\n";

      return 0;
}
```

Time.cpp

```cpp
#include <iostream>
#include <iomanip>
#include "Time.h"

using namespace std;
static int x = 30;
namespace Tempo{
Time :: Time(const int h, const int m, const int s)
: hour(h), minute (m), second(s) { }

void Time :: setTime(const int h, const int m, const int s)
{
    hour = h;
    minute = m;
    second = s;
}
/*(...)*/
```

Time.h

```
#ifndef TIME_H
#define TIME_H
namespace Tempo{
class Time
{
private :
    int hour;
    int minute;
    int second;
public :
    //with default value
    Time(const int h = 0, const int m = 0, const int s = 0);
    // setter function
    void setTime(const int h, const int m, const int s);
    // Print a description of object in " hh:mm:ss"
    void print() const;
    //compare two time object
    bool equals(const Time&);
};

#endif
```

main.cpp

```cpp
#include <iostream>
#include "Time.h"

using namespace std;
int main()
{
    int x = 20;
    using Tempo::Time;
    Time t1(20, 50, 59);
    t1.print(); // 10:50:59
    Time t2;
    t2.print(); // 06:39:09
    t2.setTime(8, 39, 9);
    t2.print(); // 06:39:09

    if(t1.equals(t2))
        cout << "Two objects are equal\n";
        else
        cout << "Two objects are not equal\n";

    return 0;
}
```

Estrutura arquivo *Makefile*

```
CFLAGS = -O
CC = g++
TimeTest: main.o Time.o
 $(CC) $(CFLAGS) -o TimeTest main.o Time.o
main.o: main.cpp
 $(CC) $(CFLAGS) -c main.cpp
Time.o: Time.cpp
 $(CC) $(CFLAGS) -c Time.cpp
clean: rm -f core *.o
```

Compilação & Execução

```
cupertsj@cupertsj-Inspiron-3437:~/Prog II/aula9/$ make
g++ -O -c main.cpp
g++ -O -c Time.cpp
g++ -O -o TimeTest main.o Time.o
cupertsj@cupertsj-Inspiron-3437:~/Prog II/aula9/$ ./TimeTest
19 : 50 : 59
00 : 00 : 00
08 : 39 : 09
Two objects are not equal
```