Programação de Computadores
Capítulo 7: O pré-processador de C++

Teresa Martinez Gomes


DEEC
DEPARTAMENTO DE ENGENHARIA
ELECTROTÉCNICA E DE COMPUTADORES

2013-2014

Plano: O pré-processador de C++

O Pré-Processador de C++

Introdução

- O pre-processador de C++ é um programa separado do compilador de C++, automaticamente executado pela "envolvente" do compilador de C++
- O pré-processador era muito importante em C, mas é menos relevante em C++.
- Pode dizer-se que o pré-processador é um processador de *macros* que trabalha com expressões.

  Exemplo de uma macro de substituição:

  ```
  #define TAMANHO 100
  ```

  Sempre que `TAMANHO` for encontrado será substituído, assim:

  ```
  num = TAMANHO * 2 ;
  ```

  será transformado em:

  ```
  num = 100 * 2;
  ```

O Pré-Processador de C++

A directiva `#define`. Macros

- A forma geral da instrução `#define` é:
  `#define Nome Texto_de_Substituição`
- `Nome`
  poderá ser qualquer identificador válido em C++
- `Texto_de_Substituição`
  pode ser qualquer linha de texto, com espaços, operadores, ou qualquer tipo de caracteres, desde que não ultrapasse uma linha

O Pré-Processador de C++

A directiva `#define`. Macros

- A instrução,
  `#define TAMANHO 100`
  é semelhante a:
  `const int TAMANHO = 100;`
- **Mas** o pré-processador não sabe nada sobre o alcance ou o tipo dos dados em C++! Além disso a correcção sintáctica da constante não é verificada pelo pré-processador.

  Logo uma declaração `const` é em geral preferível à utilização da instrução `define`.
- Outro exemplo:
  `#define CICLOETERNO for(;;)`
- As macros também podem ter argumentos:
  `#define PRINT(a,b) cout << (a) << ", "<< (b)`
- Segundo Bjarne Stroustrup a primeira regra acerca de macros é: Don't use them unless you have to.

O Pré-Processador de C++

Directivas

- Uma directiva (instrução) do pré-processador termina no final de uma linha – recorde que em C++ todas as instruções terminam com ';'
- Uma directiva que pretenda estender-se por mais do que uma linha precisa de uma barra à esquerda, \, no final da linha, que indica a continuação na linha seguinte.
- Coloque-se parêntesis em tudo.
- Se uma directiva tem mais do que uma instrução, coloquem-se dentro de chavetas.
- Não utilize "=" nem ";".
- É uma prática comum em programação usar apenas letras maiúsculas no nome das macros.
- A directiva `define` é importante na compilação condicional, como iremos ver.

O Pré-Processador de C++

Directivas. Compilação condicional.

```
#include <iostream>
using namespace std;

#define PRINT(a,b) cout << (a) << ", "<< (b)
#define DEBUG  /* Imprime todas as mensagens de DEBUG */
//#undef DEBUG  /* Nao imprime nehuma mensagem de DEBUG*/

int main(){
  int i = 1, j=2;
  PRINT(i,j);

  #ifdef DEBUG
  std::cout << "\nMostra i = " << i << endl;
  #endif /* DEBUG */

  return(0);
}
```

```
1, 2
Mostra i = 1
```

O Pré-Processador de C++

Directivas: #ifdef, #ifndef, ...

- ```
  #ifdef NOME
  # ...
  :
  #endif

  #ifdef OUTRO
  # ...
  :
  #else
  # ...
  :
  #endif
  ```

```
#ifndef NOME
# ...
:
#endif

#ifndef OUTRO
# ...
:
#else
# ...
:
#endif
```
```

O Pré-Processador de C++

Directivas: #include

- Quando escrevemos:
  `#include <iostream>`

  Isto indica ao pré-processador que deve pegar no ficheiro *iostream* (*header file*) e inseri-lo no programa actual.
- Ficheiros (com declarações ou definições) que são incluídos noutros ficheiros são designados de *header files* (ficheiros de cabeçalho).

  No unix esse ficheiros estão em /usr/include
- Ficheiros locais podem ser incluídos colocando o seu nome entre aspas `""`. Embora o nome possa ser qualquer é boa prática que a extensão seja "*.h*".

  O nome do ficheiro pode ser local, incluir o caminho completo, ou ser relativo ao ponto de compilação.

O Pré-Processador de C++

Directivas: `#include`

- Um ficheiro pode incluir mais do que um ficheiro *header*. Um ficheiro *header* pode ser incluído em mais do um ficheiro.
  Inclusões sucessivas podem causar problemas, quando a mesma declaração ou definição surge em duplicado.
- Por exemplo se o ficheiro `param.h` contiver definições ou declarações que são comuns a outros dois ficheiros, `a.h` e `b.h` (ou seja em ambos ocorre a instrução `#include param.h`) e num outro programa escrevermos:

  ```
  #include a.h
  #include b.h
  ```

  quando o ficheiro `b.h` é incluído o ficheiro `param.h` seria incluído pela segunda vez e isso poderia causar problemas de compilação.

O Pré-Processador de C++

Directivas: #include

- Este tipo de problemas resolve-se com inclusão condicional.
- Em cada ficheiro a ser incluído, colocam-se algumas instruções, directivas do pré-processador, que permitem verificar se esse ficheiro já foi incluído ou não.
- Por exemplo, no ficheiro `param.h`, para verificar se já foi incluído pode escrever-se:

```
#ifndef _PARAM_H
/* Conteúdo de param.h */

#define _PARAM_H
#endif /* _PARAM_H
```

- Se esta estratégia for adoptada em todos os ficheiros ".h" que podem ser incluídos, o problema da dupla (ou múltipla) inclusão não poderá ocorrer.

O Pré-Processador de C++

Compilação de um programa que se estende por vários ficheiros

- Considere que foi criado o ficheiro `geometria.h`, com o seguinte conteúdo:

```
#ifndef _GEOMETRIA_H
// Calcula a area de um rectangulo
float areaR(float largura, float comprimento);
// Calcula a hipotenusa de um triangulo rectangulo
float hipotenusa(float cateto1, float cateto2);
// Calcula a área e devolve a hipotenusa de um triangulo rectangulo
float recto(float cateto1, float cateto2, float & area);
#define _GEOMETRIA_H

#endif /* _GEOMETRIA_H */
```

- Considere agora que existe um ficheiro, `geometria.cpp`, com o seguinte conteúdo:

```
#include "geometria.h"
// Calcula a area de um rectangulo
float areaR(float largura, float comprimento){ ... }
// Calcula a hipotenusa de um triangulo rectangulo
float hipotenusa(float cateto1, float cateto2){ ... }
// Calcula a area e devolve a hipotenusa de um triangulo rectangulo

float recto(float cateto1, float cateto2, float & area){ ... }
```

- Finalmente um programa principal, que inclui `geometria.h` e que pode ser ligado com `geometria.cpp` para criar um executável.

O Pré-Processador de C++

Compilação de um programa que se estende por vários ficheiros

- Temos então os ficheiros `principal.cpp`, `geometria.cpp`, `geometria.h`, que queremos compilar e ligar para criar um executável, `principal.exe`.
- Podemos verificar a correcção sintáctica, e criar os ficheiros *object*, fazendo:

  `g++ -c geometria.cpp` <*enter*>
  `g++ -c principal.cpp` <*enter*> Criando `geometria.o` e `principal.o`
- Podemos em seguida ligar tudo, utilizando os ficheiros *object* criados:

  `g++ -o principal.exe principal.o geometria.o`
- Ou podemos criar o executável utilizando o código fonte:

`g++ -o principal.exe principal.cpp geometria.cpp`

- Como pode verificar o ficheiro `geometria.h` não aparece mencionado em nenhuma linha de comando (é incluído pelo pré-processador de C++).