Templates – Parte 2

Paulo Ricardo Lisboa de Almeida

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Non-type parameters

- Um Template pode ser definido para receber um non-type parameter
  - Um valor integral (ex.: 3.1415, 700, 'B', …) ou uma enumeração
  - Um ponteiro ou referência para
    - Um objeto
    - Uma função

Exemplo

- Na aula passada, a Pilha implementada possuir um valor limite para a quantidade de itens armazenados

```
template<typename T>
class Pilha{
	public:
		static const int MAX_PILHA{10};

		Pilha():topo{-1}{
		}

		~Pilha(){
		}
		//...
	private:
		T pilha[MAX_PILHA];
		int topo;
};
```

Exemplo

- Podemos usar um non-type parameter para especificar o tamanho máximo da pilha
  - **Sem overheads!!!**
  - O non-type parameter é substituído em tempo de compilação onde necessário, de forma similar a um *#define* em *C*

Exemplo

non-type parameter

Uma pilha de até 5 elementos

No main

```
int main(){
	int retorno;
	Pilha<int, 5> p;
	p.push(1);
	p.push(2);
	p.push(3);

	while(!p.estaVazia()){
		p.pop(&retorno);
		std::cout << retorno << std::endl;
	}

	return 0;
}
```

```
template<typename T, int MAX_PILHA>
class Pilha{
	public:
		Pilha():topo{-1}{
		}

		~Pilha(){
		}

		//...

	private:
		T pilha[MAX_PILHA];
		int topo;
};
```

UDESC

Uma solução melhor

size_t é definido em cstddef

Cast e comparação extras necessárias pois estamos comparando um valor com sinal e um sem

```
#include <cstddef>

template<typename T, size_t MAX_PILHA>
class Pilha{
        public:
                Pilha():topo{-1}{
                }
                ~Pilha(){
                }
                bool estaCheia() const{
                        if(topo > -1 && (size_t)topo >= MAX_PILHA -1)
                                return true;
                        return false;
                }
                //...
        private:
                T pilha[MAX_PILHA];
                int topo;
};
#endif
```

UDESC

size_t

- size_t é substituído por algum tipo que possui ao menos 16 bits (a partir da especificação C++11) e representa o maior valor tamanho teórico possível de um objeto, incluindo vetores
  - A partir do C++14, uma tipo em que o tamanho não pode ser representado por um size_t é considerado mal formado
- Vantagem do size_t
  - É substituído pelo tipo correto durante a compilação
  - O tipo correto depende da arquitetura da máquina
    - Exemplo: o tamanho máximo em um x86 é diferente do de um microcontrolador
    - Melhoramos a **portabilidade**
    - Em meu x86-64, o size_t é traduzido para um *unsigned long*

Array STL

- A classe Array da STL utiliza non-type parameters em sua implementação
  - Com isso a classe array é muito similar a um array convencional
  - www.cplusplus.com/reference/array/array/

```
#include <iostream>
#include <array>

int main (){
  std::array<int,5> myarray = { 2, 16, 77, 34, 50 };

  std::cout << "myarray contains:";
  for ( auto it = myarray.begin(); it != myarray.end(); ++it )
    std::cout << ' ' << *it;
  std::cout << '\n';

  return 0;
}
```

Argumentos default nos Templates

- Os templates podem aceitar valores Default
  - Mesmo conceito com parâmetros *default* de funções
  - Caso nada seja especificado, o *default* é usado
  - Disponível a partir do **C++11**

Exemplo

Por padrão, T é int, e MAX_PILHA é 10

Pilha “Padrão”

```
#include <iostream>

#include "Pilha.hpp"

int main(){
	int retorno;
	Pilha<> p;
	Pilha<double, 30> p2;

	//...


	return 0;
}
```

Pilha de doubles com no máximo 30 itens

```
#include <cstddef>

template<typename T = int, size_t MAX_PILHA = 10>
class Pilha{
	//...
};
```

UDESC

Exemplo

```
#include <iostream>

#include "Pilha.hpp"

int main(){
	int retorno;
	Pilha<> p;
	Pilha<double, 30> p2;

	//...


	return 0;
}
```

Mesmo a pilha padrão **precisa** dos **< >** em sua definição. Essa verborreia foi eliminada no C++17, com a introdução do *class template argument deduction*

en.cppreference.com/w/cpp/language/class_template_argument_deduction

UDESC

Exercício

1. Modifique a classe Fila da aula passada. Internamente nessa classe você provavelmente utilizou um deque ou uma list para representar a fila. Defina **o tipo** da representação interna (deque, fila, array, …) via um Template, e esse template deve ter um valor default (exemplo: caso nada seja especificado, sempre usar uma fila).

Referências

- en.cppreference.com/w/cpp/language/template_parameters
- en.cppreference.com/w/cpp/types/size_t
- Stroustrup, B. **The Design and Evolution of C++.** Pearson Education. 1994. ISBN 9780135229477
- DEITEL, P.; DEITEL, H. **C++ how to Program**. [S.l.]: Pearson, 2017. ISBN 9780134448237
- STROUSTRUP, B. **The C++ Programming Language**. Pearson Education, 2013. ISBN 9780133522853.