Funções inline

Paulo Ricardo Lisboa de Almeida

UDESC
UNIVERSIDADE
DO ESTADO DE
SANTA CATARINA

Overhead

- As funções membro são excelentes ideias do ponto de vista do reuso e da engenharia de software
  - Mas geram overheads
    - Por quê?

Overhead

- As funções membro são excelentes ideias do ponto de vista do reuso e da engenharia de software
  - Mas geram overheads
    - Chamadas de função podem custar caro para a CPU
      - Atualizar a pilha
      - Redirecionar para outro trecho do programa
        - Os dados podem não estar na cache

Inline

- C++ possui o conceito de *inline*
  - Uma função marcada com inline é injetada nos trechos onde ela é chamada pelo compilador
    - O compilador não chama a função, mas coloca uma cópia (otimizada) da função nos trechos onde ela é chamada

Inline

- Para declarar uma função como *inline*, basta adicionar *inline* na frente da declaração da função
- A função membro precisa ser **implementada** no .hpp
  - Na verdade podemos implementar a função no .cpp, mas isso vai nos gerar muitas dores de cabeça
    - Veremos os motivos adiante na disciplina

Faça o teste

A implementação da função inline fica no .hpp

Note o **const**. Indica que o parâmetro é somente leitura. Isso pode auxiliar o compilador a gerar o inline de forma eficiente

```
#ifndef PESSOA_H
#define PESSOA_H

#include<string>

class Pessoa{
	public:
		inline unsigned char getIdade(){
			return idade;
		}
		inline void setIdade(const unsigned char novaIdade){
			idade = novaIdade;
		}

		std::string getNome();
		void setNome(std::string novoNome);

		unsigned long getCpf();
		bool setCpf(unsigned long novoCpf);
	private:
		// ... variáveis e métodos privados
};
#endif
```

UDESC

Faça o teste

- No .cpp, remova a implementação dos gets e sets de idade (que agora são inline)

Atenção

- Ao declarar uma função como inline (ou remover o inline), você deve apagar todos os arquivos objeto (.o) já gerados
  - Use *make clean*
  - Remova os .gch caso exista algum (deixado caso você tenha gerado apenas o Assembly)

Compile e execute

- Edite o main para realizar alterações na idade
  - Compile e execute

Pare para pensar

- Quais os problemas que podem ser gerados por funções *inline*?

Pare para pensar

- Quais os problemas que podem ser gerados por funções *inline*?
  - O trecho da função é copiado para todos os lugares onde ela é chamada
    - O programa fica maior
      - Aumento de pressão na cache de instruções
        - Pode ter o efeito contrário ao desejado, deixando o programa mais lento
      - Ao alterar a função, todos os “clientes” que usam a função precisam ser recompilados
        - Recompilar apenas a função não surtirá efeito
      - Funções inline não podem ser virtuais, não aceitando sobrecarga
        - A sobrecarga é um dos pontos fortes da orientação a objetos
          - Estudaremos no futuro

Novos testes

- Compile uma versão do programa com os *gets* e *sets* de idade *inline*, e uma sem eles estarem *inline*
  - Compile usando o make
  - Depois de usar o make, os arquivos objeto (.o) estarão no diretório de compilação
  - Utilize o comando *g++ -S main.cpp* para gerar o assembly do main, com a linkedição já feita
  - Será criado um arquivo main.s, com o assembly x86 do main compilado
  - Compare o assembly com e sem o *inline*
  - Dica
    - Utilize uma ferramenta visual para comparação de arquivos, como o meld
    - sudo apt-get install meld
    - meld assemblySemInline.s assemblyComInline.s
  - **Se você não aprender assembly ainda, não se preocupe**
    - Somente verifique que os assemblys gerados são diferentes para as funções com e sem inline

Diferenças de desempenho

- No nosso programa, provavelmente você não notará diferenças de desempenho
  - O programa é muito pequeno
    - Tudo cabe na cache
    - A pilha do programa é pequena
    - A sua cpu x86 é muito bom em prever saltos (ex.: chamadas de função), principalmente quando temos poucas funções e chamadas
    - O compilador realiza diversas otimizações automaticamente
      - Nosso programa é simples e o compilador g++ é muito sofisticado
- Mas essas otimizações podem fazer toda a diferença em um programa grande e complexo

Compiladores

- Os compiladores atuais são sofisticados
  - Aplicam *inline* mesmo que você não solicite
  - Podem ignorar o *inline* que você pediu
    - Por exemplo, caso o compilador chegue a conclusão que a pressão na cache de instruções está muito grande
  - Podemos desabilitar esses comportamentos com opções avançadas
    - Veja as opções do g++ digitando *man g++* no terminal
      - Complicado, principalmente as opções que ajustam a pressão nos registradores
        - Um ajuste incorreto e o seu C++ se torna em um Java (ok, nem tanto)
- Dica de compilação
  - A opção -O3 (ó maiúsculo 3) quando passada para o gcc ou g++, realiza otimizações máximas no seu programa
    - O programa pode ser tornar muito mais rápido
      - Mas pode ter efeitos colaterais

Compiladores

- Dica de compilação
  - A opção -O3 (ó maiúsculo 3) quando passada para o gcc ou g++, realiza otimizações máximas no seu programa
    - O programa pode ser tornar muito mais rápido
    - Pode ter efeitos colaterais
      - O gcc/g++ vai analisar loops por exemplo, e se ele considerar que um loop não está fazendo trabalho útil, ele será removido
      - Funções não usadas são removidas
        - E se outro programa tentar fazer a linkedição dessa função?
      - Ordem das operações podem mudar
    - No entanto, é muito raro a otimização gerar problemas reais em nossos programas

Para a próxima aula

1)Estude sobre as diretivas de otimização do gcc/g++
O, O1, O2 e O3