Capturando abandonos com TRAP

Introdução

TRAP é o mecanismo implementado pelo Symbian para capturar abandonos em aplicações c++. Seriam análogos ao try/catch de c++ padrão uma vez que permite ao programador detectar erros em tempo de execução e tratá-los da maneira devida.

Quando um código abandona é gerado um código de erro e o fluxo de execução da aplicação é desviado para a TRAP mais próxima que irá receber o código de erro gerado. Duas macros são utilizadas para capturar abandonos: TRAP e TRAPD. Ambas funcionam de forma semelhante, a única diferença está na forma como a variável que recebe o código do erro do abandono é declarada. Ao usar TRAP a variável deve ser declarada previamente enquanto que na TRAPD ela é declarada durante a execução da TRAPD.

Nota: O escopo da variável declarada pela TRAPD somente é válido no mesmo escopo da TRAPD.

Definição de TRAP e TRAPD

```
TRAP(_r, _s)
```

```
TRAPD(_r, _s)
```

Onde:

_r = é um inteiro que irá receber o código do abandono ou KErrNone caso a execução da função seja feita com sucesso.
_s = Conjunto de instruções c++ que possam gerar um abandono. É possível especificar mais de uma separando-as por vírgula mas não é aconselhável para manter a legibilidade do código.

Exemplos de como utilizar TRAP e TRAPD

Exemplo com TRAP:

```
TInt err;
TRAP(err, FuncaoQuePodeAbandonarL());
if( err == KErrNone ) // não ocorreu abandono
{
        // implementação
}
else // Ocorreu abandono
{
        // implementação
}
```

Exemplo com *TRAPD*:

```
TRAPD(err, FuncaoQuePodeAbandonarL());
if( err == KErrNone ) // não ocorreu abandono
{
        // implementação
}
else // Ocorreu abandono
{
        // implementação
}
```

OBS: A utilização de TRAPs deve ser feita de maneira cautelosa pois introduzem certo *overhead* e aumento do binário da aplicação.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.