UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
INSTITUTO METRÓPOLE DIGITAL
IMD0030 – LINGUAGEM DE PROGRAMAÇÃO I

Geração automática de documentação de código fonte com Doxygen

1. Objetivo

O objetivo deste documento é fornecer uma visão geral sobre a geração automática de documentação de código fonte na linguagem de programação C++ utilizando a ferramenta *Doxygen*. Vale salientar que este documento busca introduzir de forma breve apenas os elementos básicos para gerar documentação de código fonte utilizando o Doxygen. Caso seja de interesse do leitor aprender a utilizar funções mais avançadas para a documentação, recomenda-se a leitura do manual do Doxygen disponível através da seguinte URL: http://www.stack.nl/~dimitri/doxygen/manual/.

2. Sobre o Doxygen

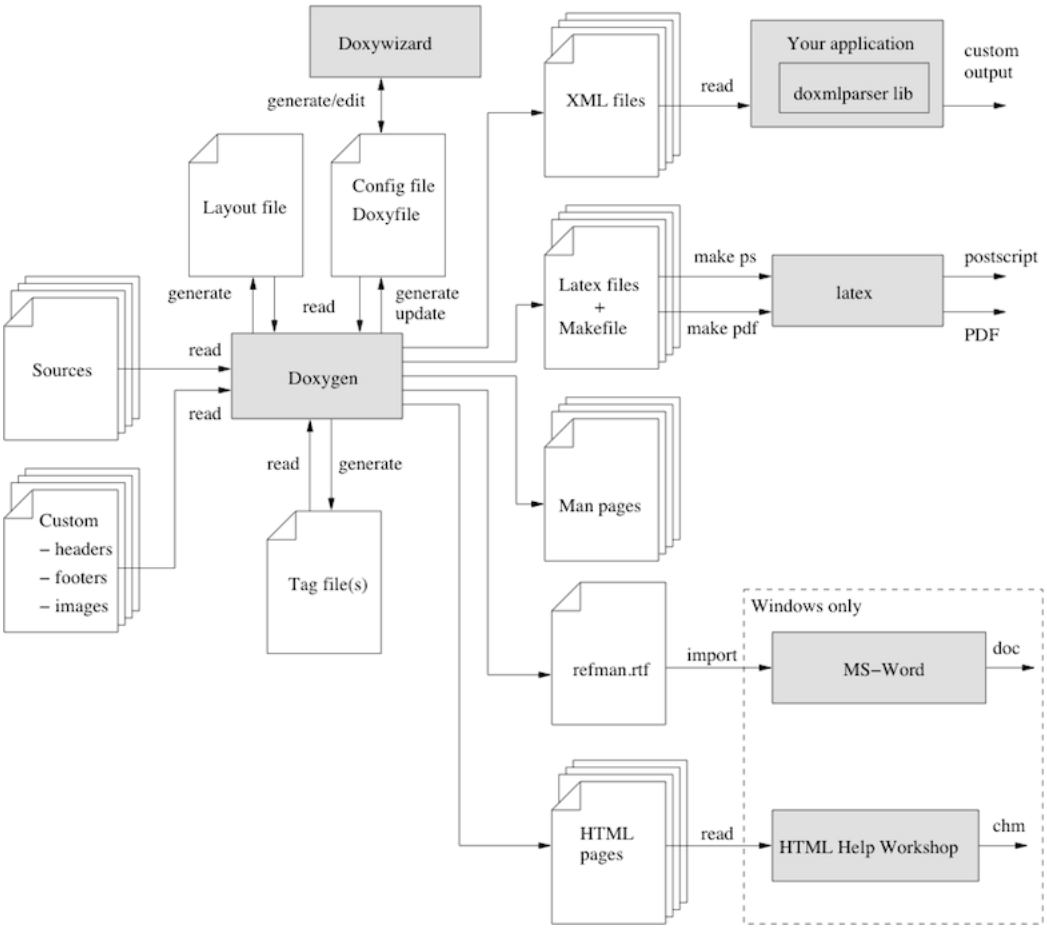
Nos dias atuais, é possível observar uma maior demanda por abordagens que permitam um maior e melhor (re)uso de *software* a fim de reduzir tempo e custo de desenvolvimento, manutenção e evolução de sistemas. Nesse contexto, um aspecto de contribuição fundamental para tal (re)uso é uma boa documentação, capaz de apresentar uma visão geral do *software* sem ser necessário expor detalhes da sua implementação. Mesmo assim, produzir uma documentação de qualidade pode se mostrar uma tarefa relativamente difícil e demandar certo tempo do processo de desenvolvimento.

A fim de auxiliar o desenvolvedor na tarefa de produzir uma boa documentação de *software*, é comum fazer uso de *ferramentas de documentação automática*. Uma das principais vantagens da utilização desse tipo de ferramenta é permitir que o desenvolvedor se concentre na elaboração de uma boa documentação do código (conteúdo), ao invés de se preocupar com a aparência que tal documentação terá para o usuário final. Isso é possível graças a *marcações* (ou *anotações*) no código fonte, utilizadas pela ferramenta para gerar e organizar automaticamente a documentação final.

Dentre as ferramentas para documentação automática está o *Doxygen*, disponível através do endereço http://www.doxygen.org/. O Doxygen consiste numa ferramenta para documentação de programas implementados nas linguagens de programação C, C++, Java, Objective-C, Python, PHP, entre outras linguagens populares. Versões do Doxygen podem ser encontradas para os sistemas operacionais Microsoft® Windows, Linux e Mac OS X.

A figura a seguir, disponível no manual do Doxygen disponível na Internet, ilustra como a ferramenta funciona. Em essência, o Doxygen utiliza como entrada (1) um *arquivo de configuração* com um conjunto de opções que especificam alguns parâmetros referentes ao projeto em questão e como o Doxygen irá se comportar, e (2) os *arquivos fonte* comentados seguindo o estilo reconhecido pela ferramenta. Outros arquivos adicionais podem ser fornecidos como entrada para uma maior

customização da documentação a ser gerada, a exemplo de um *arquivo de layout* referente a como a documentação será organizada e disposta, bem como imagens, cabeçalhos, rodapés etc. Como saída, o Doxygen pode gerar documentação em formatos comumente utilizados, tais como páginas Web (HTML), LaTeX, RTF, XML e *man pages* usadas em sistemas Unix.



3. Instalando e configurando o Doxygen

Há diversas formas de instalar o Doxygen. Neste documento, o enfoque será na instalação da ferramenta no sistema operacional Linux, de modo que formas alternativas podem ser encontradas na página da ferramenta na Internet.

3.1 Instalando de um repositório Git

Provavelmente a forma mais indicada para instalação do Doxygen é através de um repositório Git hospedado no GitHub, que contém a versão mais atual da ferramenta. Tendo o Git instalado no sistema, deve-se clonar (operação *clone*) o repositório para obter os arquivos fonte:

```
$ git clone https://github.com/doxygen/doxygen.git
$ cd doxygen
```

Uma vez que os arquivos fonte foram devidamente baixados para o diretório `doxygen`, é preciso preparar o *Makefile* para a compilação e compilar o Doxygen com o comando `make`:

```
$ mkdir build
$ cd build
$ cmake -G "Unix Makefiles"...
$ make
```

Após a geração dos binários (localizados no diretório `doxygen/bin`), é possível instalar o Doxygen no sistema como o comando:

```
$ cd ..
$ make install
```

Nota: O acesso público ao repositório Git do Doxygen é apenas de leitura (isto é, *read-only*). Dessa forma, não é possível submeter alterações diretamente através de *commits*.

3.2 Instalando os binários

É possível também instalar o Doxygen já compilado, fazendo o *download* da versão correta disponível no endereço http://www.stack.nl/~dimitri/doxygen/download.html. Essa página também contém as instruções de instalação a serem seguidas. No caso do sistema operacional Linux, é possível instalar o Doxygen também através do instalador de pacotes `apt-get`, que instalará a ferramenta e respectivas dependências:

```
$ sudo apt-get install doxygen
```

Nota: Antes de instalar pacotes no sistema operacional Linux, é preciso ter permissões de super usuário (*super user*).

3.3 Compilando o código fonte

É possível ainda instalar o Doxygen a partir do código fonte da própria ferramenta. O código fonte também está disponível no endereço http://www.stack.nl/~dimitri/doxygen/download.html.

4. Anotando o código fonte

Um bloco de documentação no estilo Doxygen difere ligeiramente do padrão de comentário existente nas linguagens de programação C e C++, por exemplo, uma vez que alguns *marcadores* adicionais são requeridos. São justamente esses marcadores (também chamados de *anotações*) que permitem ao Doxygen reconhecer que aquela parte do arquivo deve ser utilizada no momento em que a documentação é gerada. A relação completa de marcadores pode ser encontrada no endereço https://www.stack.nl/~dimitri/doxygen/manual/commands.html.

4.1 Adicionando descrições no código

Há dois tipos de descrição que juntos formam a documentação de um item de código fonte, a saber, uma *descrição breve* e uma *descrição detalhada*. Apesar de ambas serem opcionais, utilizar mais de uma descrição do mesmo tipo não é permitido. Uma descrição breve consiste num comentário de única linha, enquanto que a descrição detalhada, como o nome já denota, pode ocupar diversas linhas.

Uma descrição detalhada pode ser escrita de diferentes formas. Neste documento, será utilizado o estilo *JavaDoc*, similar a um bloco de comentário no estilo C/C++, porém com dois asteriscos na abertura. Cabe ressaltar que o uso do asterisco no início de cada linha intermediária do bloco de comentários é opcional.

```
/**
 * ... texto ...
 */
```

Devido à flexibilidade do Doxygen, há igualmente diferentes formas de adicionar uma descrição breve. A primeira delas é através da anotação `@brief` dentro de um bloco especial de comentário, como mostrado anteriormente. Uma descrição breve é separada da descrição detalhada por uma linha em branco.

```
/**
 * @brief Descricao breve
 *
 * A partir deste ponto, segue-se a descricao detalhada.
 */
```

Outra forma de especificar, explicitamente, a descrição detalhada é por meio da anotação @details dentro de um bloco especial de comentário:

```
/**
 * @brief   Descricao breve
 * @details Descricao detalhada.
 */
```

4.2 Documentando membros

Uma das tarefas fundamentais na documentação de código fonte nas linguagens de programação C e C++ é a clara definição dos membros de um arquivo, estrutura (*struct*), união (*union*), classe (*class*) ou enumeração (*enum*). Embora essas informações sejam muitas vezes adicionadas nos blocos de comentário (descrições), isso não dispensa a documentação individual de cada membro. Para isso, usa-se o símbolo < num bloco de comentário logo após a declaração de tal membro, como mostra o exemplo a seguir:

```
int qtAlunos;          /**< Define a quantidade de alunos. O valor maximo e... */
```

Outra alternativa é incluir o bloco de comentário antes da declaração do membro:

```
/** @brief Define a quantidade de alunos. O valor maximo e... */
int qtAlunos;
```

4.3 Documentando funções

Na documentação de funções, além de se fazer uso de descrições breves e detalhadas, é possível documentar os parâmetros de entrada e os retornos que tais funções possuem. Para documentar parâmetros de uma função, deve-se usar a anotação `@param` seguida do nome do parâmetro e sua descrição. Por sua vez, o retorno da função é especificado por meio da anotação `@return`. Opcionalmente, é possível também indicar a direção do parâmetro, se é um parâmetro de entrada (`[in]`), saída (`[out]`) ou entrada e saída (`[in,out]`), como mostra o exemplo a seguir:

```
/**
 * @brief Funcao que calcula o fatorial de um numero
 * @param[in] n Numero cujo fatorial sera calculado
 * @return Fatorial do numero
 */
long double fatorial(long double n);
```

4.4 Documentando arquivos fonte

Igualmente importante à documentação de membros, classes, funções, etc. é a documentação dos arquivos fonte propriamente ditos. Utilizando um bloco de comentário especial no início do arquivo em questão, pode-se fazer uso dos marcadores para descrições breve (`@brief`) e detalhada (`@details`) e dos seguintes marcadores úteis:

Marcador (anotação)	Descrição
`@file`	Documentação do arquivo fonte em questão
`@author`	Inserção do(s) nome(s) do(s) autor(es) do código fonte em questão. É possível separar os nomes de múltiplos autores por meio de vírgulas ou múltiplas anotações `@author` contendo o nome de um autor.
`@since`	Inserção da data de início da implementação
`@date`	Inserção de data (por exemplo, a data da última modificação do arquivo)
`@version`	Indicação da versão atual do arquivo
`@sa`	Inserção de referências cruzadas para classes, funções, métodos, variáveis, arquivos ou mesmo endereços da Internet

```
/**
 * @file    hello.cpp
 * @brief   Primeiro programa na linguagem de programacao C++
 * @author  Joao dos Anzois
 * @since   01/01/2016
 * @date    31/12/2016
 * @sa      http://www.google.com/
 */
```

5. Gerando documentação automaticamente com Doxygen

Tendo-se o Doxygen devidamente instalado, ele pode ser utilizado para a geração automática do código fonte. Primeiramente, é necessário criar um arquivo de configuração que conterá algumas opções que determinarão como a documentação deverá ser gerada. Apesar de ser possível criar manualmente tal arquivo de configuração, o Doxygen permite cria-lo automaticamente através do comando

```
$ doxygen -g <nome_arquivo>
```

que irá criar um arquivo com o nome que foi especificado no diretório no qual o Doxygen está sendo executado. Caso o nome não seja especificado na execução do comando, o Doxygen irá criar o arquivo com o nome `Doxyfile`.

O arquivo de configuração automaticamente gerado pelo Doxygen já contém uma série de opções estabelecidas por padrão, fazendo com que sejam necessárias pouquíssimas modificações. Dentre as diversas opções existentes no arquivo de configuração (também chamadas de *tags*), as principais são relacionadas a seguir. A relação de todas as opções possíveis para o arquivo de configuração do Doxygen encontra-se em https://www.stack.nl/~dimitri/doxygen/manual/config.html.

Opção (*tag*)	Descrição
`PROJECT_NAME`	Nome do projeto em questão, devendo vir entre aspas se contiver espaços em branco
`PROJECT_BRIEF`	Descrição breve do projeto em questão, devendo vir entre aspas se contiver espaços em branco
`OUTPUT_DIRECTORY`	Endereço do diretório no qual será armazenada a documentação gerada. Caso não seja especificado, o diretório atual no qual o Doxygen está sendo executado será utilizado por padrão.
`OUTPUT_LANGUAGE`	Idioma no qual a documentação será gerada. O valor padrão para esta *tag* é `English`, referindo-se à documentação em Inglês. Para geração de documentação em Português do Brasil, o valor para esta *tag* deve ser `Brazilian`.
`EXTRACT_ALL`	Determina se deverá ser gerada documentação para todos os membros de uma entidade, mesmo que não haja documentação disponível. O valor padrão para esta *tag* é `NO`.
`EXTRACT_PRIVATE`	Determina se deverá ser gerada documentação para membros privados de uma classe. O valor padrão para esta *tag* é `NO`.
`EXTRACT_STATIC`	Determina se deverá ser gerada documentação para membros estáticos de um arquivo. O valor padrão para esta *tag* é `NO`.
`RECURSIVE`	Determina se subdiretórios deverão ser pesquisados por arquivos de código fonte. O valor padrão para esta *tag* é `NO`.
`GENERATE_HTML`	Determina se deverá ser gerada documentação em formato de páginas Web (HTML). O valor padrão para esta *tag* é `YES`.
`HTML_OUTPUT`	Diretório no qual deverá ser armazenada a documentação em formato de páginas Web (HTML), dentro do diretório especificado por meio da *tag* `OUTPUT_DIRECTORY`. O valor padrão para esta *tag* é `html` e requer que a *tag* `GENERATE_HTML` possua valor `YES`.

`GENERATE_LATEX`	Determina se deverá ser gerada documentação em formato LaTeX. O valor padrão para esta *tag* é `YES`.
`LATEX_OUTPUT`	Diretório no qual deverá ser armazenada a documentação em formato LaTeX, dentro do diretório especificado por meio da *tag* `OUTPUT_DIRECTORY`. O valor padrão para esta *tag* é `latex` e requer que a *tag* `GENERATE_LATEX` possua valor `YES`.
`GENERATE_RTF`	Determina se deverá ser gerada documentação em formato RTF. O valor padrão para esta *tag* é `NO`.
`RTF_OUTPUT`	Diretório no qual deverá ser armazenada a documentação em formato RTF, dentro do diretório especificado por meio da *tag* `OUTPUT_DIRECTORY`. O valor padrão para esta *tag* é `rtf` e requer que a *tag* `GENERATE_RTF` possua valor `YES`.
`GENERATE_MAN`	Determina se deverá ser gerada documentação em formato de *man pages*. O valor padrão para esta *tag* é `NO`.
`MAN_OUTPUT`	Diretório no qual deverá ser armazenada a documentação em formato de *man pages*, dentro do diretório especificado por meio da *tag* `OUTPUT_DIRECTORY`. O valor padrão para esta *tag* é `man` e requer que a *tag* `GENERATE_MAN` possua valor `YES`.

O formato para especificação das opções (*tags*) no arquivo de configuração é:

```
TAG = <valor>
```

Uma vez criado o arquivo de configuração do Doxygen, a documentação pode ser gerada automaticamente executando-se, no diretório que contém os arquivos fonte, o comando

```
$ doxygen <nome_arquivo>
```

em que `nome_arquivo` é o nome do arquivo de configuração criado. Caso o nome do arquivo tenha sido mantido com o padrão `Doxyfile`, basta executar o comando

```
$ doxygen
```

6. Um exemplo

A título de exemplo, considere o trecho de código a seguir, referente à implementação de um simples programa que realiza o cálculo do índice de massa corporal (IMC) de um indivíduo a partir dos dados de altura e peso fornecidos pelo usuário via linha de comando. O programa, implementado por meio de um arquivo fonte com nome `imc.cpp`, contém, além da função principal, a definição de duas funções, uma que calcula o IMC propriamente dito (`imc`) a partir do peso e da altura do indivíduo e outra que determina o grau de obesidade com base no valor do IMC (`grau_obesidade`). Como resultado, o programa exibe o valor do IMC e a respectiva classificação de obesidade:

```
$ g++ -Wall -pedantic imc.cpp -o imc
$ ./imc 65 1.65
Indice de massa corporal: 23.8751
Grau de obesidade: Individuo com peso considerado normal
```

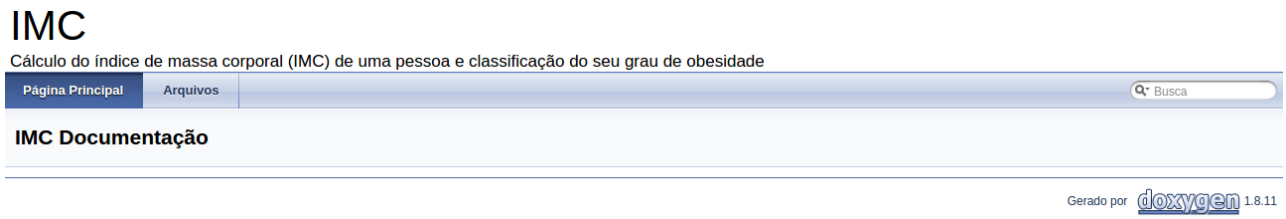
```
/**
 * @file     imc.cpp
 * @brief    Programa que calcula o indice de massa corporal (IMC) de uma pessoa
 * @details  O IMC e uma medida internacional usada para calcular se uma pessoa
esta
 *           no peso ideal. Essa medida e determinada pela divisão do peso da
pessoa
 *           (em quilogramas) pelo quadrado de sua altura (em metros)
 * @author   Pedro Paulo Pereira
 * @since    01/01/2016
 * @date     01/02/2016
 * @sa       https://pt.wikipedia.org/wiki/Índice_de_massa_corporal
 */

#include <cmath>
#include <cstdlib>
#include <iostream>
#include <string>

using namespace std;


/**
 * @brief    Funcao que calcula o IMC de um individuo adulto a partir do seu peso
e
 *           de sua altura
 * @param    peso Peso em quilogramas
 * @param    altura Altura em metros
 * @return   IMC do individuo
 */
float imc(float peso, float altura) {
    return peso / pow(altura, 2);
}


/**
 * @brief    Funcao que determina o grau de obesidade de um individuo com base em
seu IMC
 * @details  A classificação do grau de obesidade de um individuo adulto e
determinada
 *           por uma tabela de referencia internacional com base no IMC
 * @param    imc IMC do individuo
 * @return   Grau de obesidade do individuo
 * @sa       http://apps.who.int/bmi/index.jsp?introPage=intro_3.html
 */
string grau_obesidade(float imc) {
    if (imc < 18.5) {
        return "Individuo abaixo do peso";
    } else if (imc >= 18.5 && imc < 25) {
       return "Individuo com peso considerado normal";
    } else if (imc >= 25 && imc < 30) {
       return "Individuo com sobrepeso";
    } else if (imc >= 30 && imc < 35) {
       return "Individuo obeso";
```

```
/**
 * @brief   Funcao principal
 * @details Calculo do IMC do individuo a partir dos dados de altura e peso
 *          fornecidos pelo usuário via linha de comando e classificacao do grau
 *          de obesidade com base no IMC. O primeiro argumento refere-se ao peso
 *          (em quilogramas) e o segundo a altura (em metros).
 * @param   argc Numero de argumentos fornecidos via linha de comando
 * @param   argv Argumentos fornecidos via linha de comando
 */
int main(int argc, char* argv[]) {
    float peso = atof(argv[1]);           // Peso do individuo
    float altura = atof(argv[2]);         // Altura do individuo

    // Calculo do IMC
    float valor_imc = imc(peso, altura);
    cout << "Indice de massa corporal: " << valor_imc << endl;

    // Classificacao do grau de obesidade
    cout << "Grau de obesidade: " << grau_obesidade(valor_imc) << endl;

    return 0;
}
```

No código exemplo acima, note que cada função é antecedida por um bloco de comentário especial com as anotações de documentação reconhecidas pelo Doxygen. Como descrito anteriormente, antes de gerar a documentação propriamente dita, é necessário criar automática ou manualmente o arquivo de configuração que o Doxygen utiliza como entrada. Por questões de simplicidade, considere o seguinte trecho como parte de um arquivo com nome `Doxyfile` contendo as opções de configuração necessárias à geração da documentação:

```
PROJECT_NAME       = "IMC"
PROJECT_BRIEF      = "Cálculo do índice de massa corporal (IMC) de uma pessoa e
classificação do seu grau de obesidade"
OUTPUT_DIRECTORY   =
OUTPUT_LANGUAGE    = Brazilian
RECURSIVE          = YES
GENERATE_HTML      = YES
HTML_OUTPUT        = doc
GENERATE_LATEX     = NO
```

Finalmente, executando-se o Doxygen com o arquivo `Doxyfile` acima, será criado um diretório com nome `doc` (conforme especificado no arquivo de configuração) contendo todos os arquivos de documentação do programa, na forma de páginas Web. As figuras a seguir mostram o resultado produzido pelo Doxygen.



Página principal da documentação gerada pelo Doxygen (`index.html`)

IMC

Cálculo do índice de massa corporal (IMC) de uma pessoa e classificação do seu grau de obesidade

Página Principal | Arquivos | Busca

Lista de Arquivos | Membros dos Arquivos

Lista de Arquivos

Esta é a lista de todos os arquivos documentados e suas respectivas descrições:

imc.cpp	Programa que calcula o indice de massa corporea (IMC) de uma pessoa

Gerado por doxygen 1.8.11

Página de listagem dos arquivos contidos no projeto (link `Arquivos`)

IMC

Cálculo do índice de massa corporal (IMC) de uma pessoa e classificação do seu grau de obesidade

Página Principal | Arquivos | Busca

Lista de Arquivos | Membros dos Arquivos

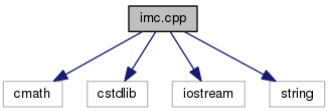
Funções

Referência do Arquivo imc.cpp

Programa que calcula o indice de massa corporea (IMC) de uma pessoa. Mais...

```
#include <cmath>
#include <cstdlib>
#include <iostream>
#include <string>
```

Gráfico de dependência de inclusões para imc.cpp:



Funções

float	**imc** (float peso, float altura) Funcao que calcula o IMC de um individuo adulto a partir do seu peso e de sua altura. Mais...
string	**grau_obesidade** (float **imc**) Funcao que determina o grau de obesidade de um individuo com base em seu IMC. Mais...
int	**main** (int argc, char *argv[]) Funcao principal. Mais...

Descrição Detalhada

Programa que calcula o indice de massa corporea (IMC) de uma pessoa.

O IMC e uma medida internacional usada para calcular se uma pessoa esta no peso ideal. Essa medida e determinada pela divisão do peso da pessoa (em quilogramas) pelo quadrado de sua altura (em metros)

Autor

Pedro Paulo Pereira

Desde

01/01/2016

Data

01/02/2016

Funções

string grau_obesidade (float imc)

Funcao que determina o grau de obesidade de um individuo com base em seu IMC.

A classificação do grau de obesidade de um individuo adulto e determinada por uma tabela de referencia internacional com base no IMC

Parâmetros

imc IMC do individuo

Retorna

Grau de obesidade do individuo

Veja também

http://apps.who.int/bmi/index.jsp?introPage=intro_3.html

float imc (float peso,
float altura
)

Funcao que calcula o IMC de um individuo adulto a partir do seu peso e de sua altura.

Parâmetros

Peso em quilogramas

Altura em metros

Retorna

IMC do individuo

int main (int argc,
char * argv[]
)

Funcao principal.

Calculo do IMC do individuo a partir dos dados de altura e peso fornecidos pelo usuário via linha de comando e classificacao do grau de obesidade com base no IMC

Parâmetros

argc Numero de argumentos fornecidos via linha de comando

argv Argumentos fornecidos via linha de comando

Gerado por doxygen 1.8.11

Documentação completa do arquivo `imc.cpp` *gerada pelo Doxygen na forma de uma página Web*