



# Doxxygen alapok

MOSZE előadás

Csapó Ádám Balázs –  
[csapo.adam@sze.hu](mailto:csapo.adam@sze.hu)

# Mi is az a Doxygen

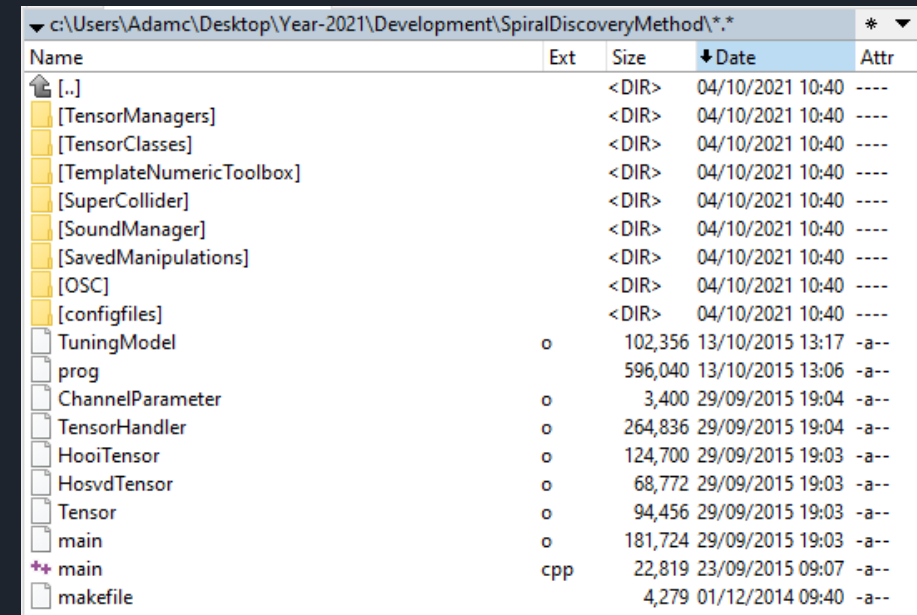
- 1997 óta létezik
- *"Doxygen is the de facto standard tool for generating documentation from annotated C++ sources, but it also supports other popular programming languages such as C, Objective-C, C#, PHP, Java, Python, IDL (Corba, Microsoft, and UNO/OpenOffice flavors), Fortran, VHDL and to some extent D."*
  - Más nyelvekre (pl. Javascript) is léteznek transzformátorok (pl. ami JS szintaxist C++ vagy Java-közeli szintaxisra konvertál, minekután használhatjuk a Doxygent – persze csak akkor, ha a JSDoc nem elég jó)
- Többféle célra használható:
  - Annotált források alapján dokumentáció generálása (html, latex -> pdf)
  - Nem annotált források alapján kódstruktúra felfedése, grafikus ábrázolása (melyik modul mit hív meg, stb.)

# Doxygen telepítése - <https://www.doxygen.nl/download.html>

- Alapvetően Mac OS X és Linux környezetekre fejlesztik, de szerencsére letölthető Windows-os telepítő is, nem kell forrásból buildelni akkor sem, ha Windowson vagyunk.
- Mac OS X rendszerre is letölthető .dmg fájl, Linuxon pedig a nagyobb csomagkezelők révén elérhető.
- Ha minden más kötél szakad, buildelhetjük is mi magunk, de kell hozzá g++, python, cmake, flex és bison.

# Doxygen használata

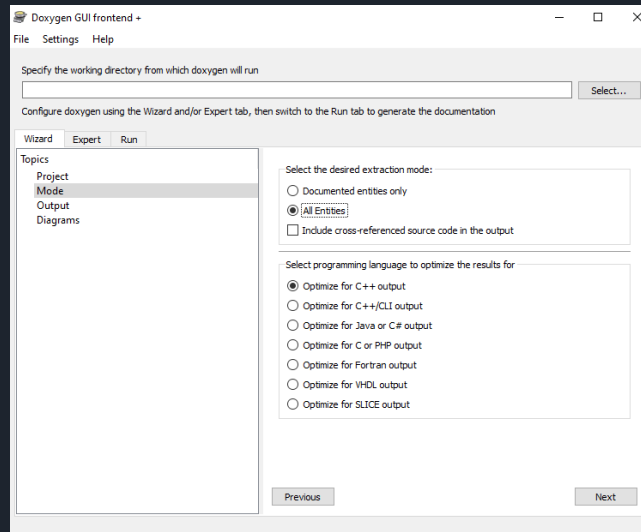
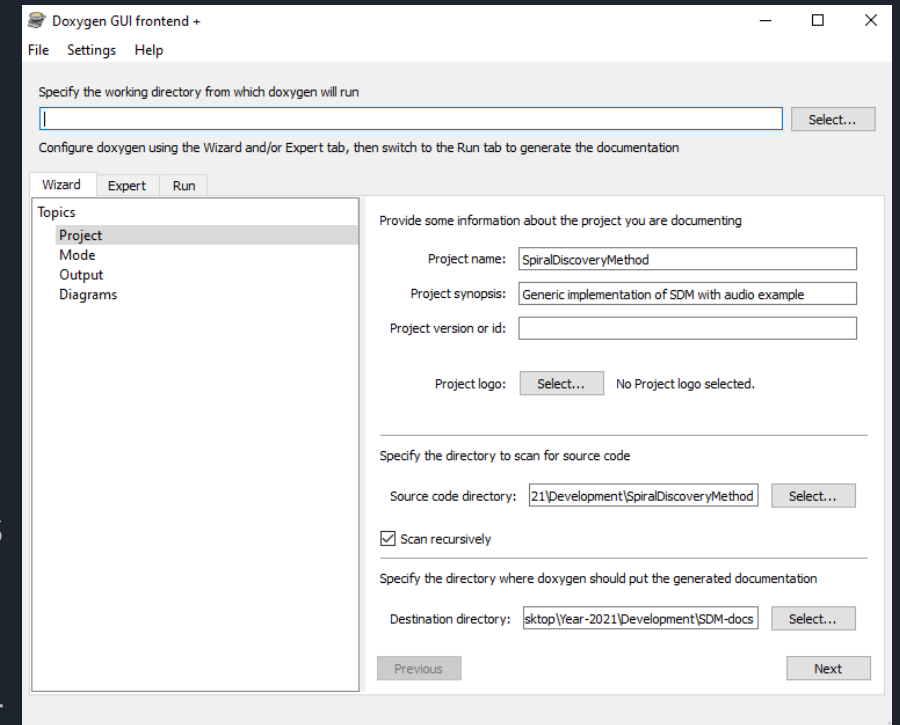
- Alapvetően konzolos alkalmazás, de van hozzá GUI frontend is (Doxywizard). A legkényelmesebb, ha ezt használjuk.
- Nézzük meg a doxygen használatát egy olyan esetre, amikor alig van doc string a kódunkban!
- (Itt egy olyan régi projektet vettem elő, melyben korlátozottan vannak dokumentációs sztringek, de jó részére ez nem jellemző)



Name	Ext	Size	Date	Attr
[..]	<DIR>		04/10/2021 10:40	----
[TensorManagers]	<DIR>		04/10/2021 10:40	----
[TensorClasses]	<DIR>		04/10/2021 10:40	----
[TemplateNumericToolbox]	<DIR>		04/10/2021 10:40	----
[SuperCollider]	<DIR>		04/10/2021 10:40	----
[SoundManager]	<DIR>		04/10/2021 10:40	----
[SavedManipulations]	<DIR>		04/10/2021 10:40	----
[OSC]	<DIR>		04/10/2021 10:40	----
[configfiles]	<DIR>		04/10/2021 10:40	----
TuningModel	o	102,356	13/10/2015 13:17	-a--
prog		596,040	13/10/2015 13:06	-a--
ChannelParameter	o	3,400	29/09/2015 19:04	-a--
TensorHandler	o	264,836	29/09/2015 19:04	-a--
HooiTensor	o	124,700	29/09/2015 19:03	-a--
HosvdTensor	o	68,772	29/09/2015 19:03	-a--
Tensor	o	94,456	29/09/2015 19:03	-a--
main	o	181,724	29/09/2015 19:03	-a--
*+ main	cpp	22,819	23/09/2015 09:07	-a--
makefile		4,279	01/12/2014 09:40	-a--

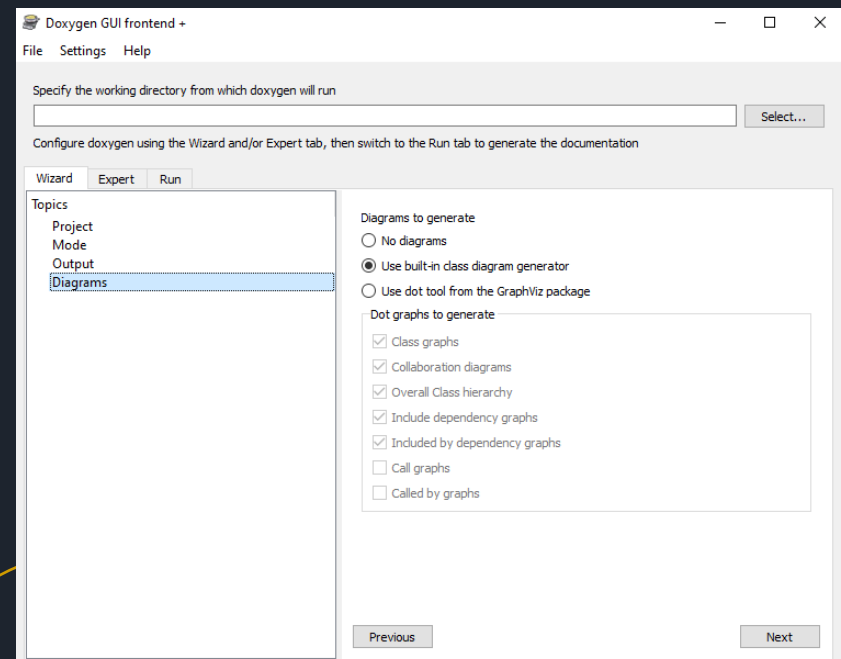
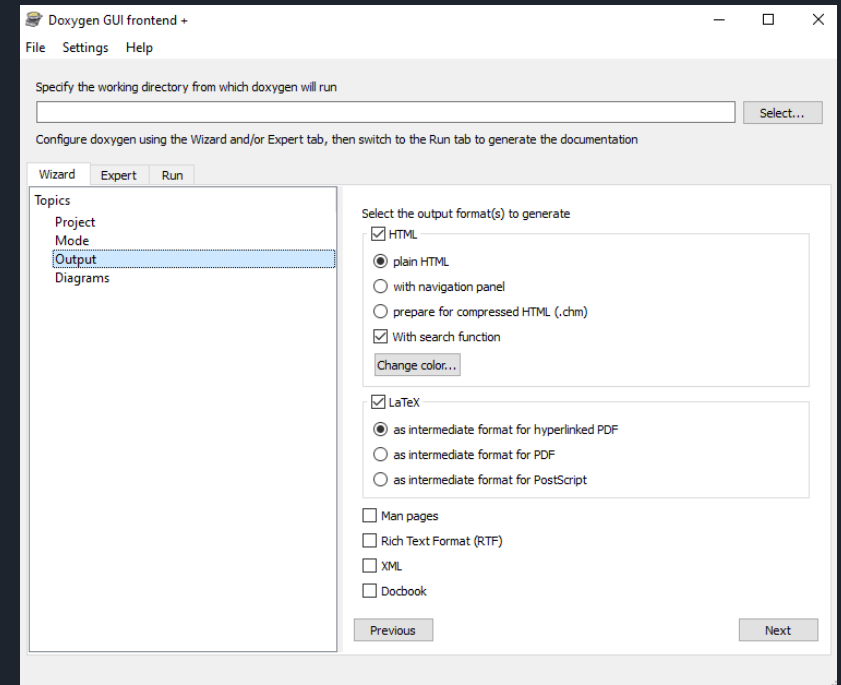
# Doxygen használata

- A Wizard panel Project topicjában megadhatjuk, mi a project neve, leírása, hol találhatóak a forrásfájlok és hova generálja le a dokumentációt.
- Ezt követően a mode topicban a forrás nyelvét konfigurálhatjuk.



# Doxygen használata

- A kimenet formátuma lehet html és / vagy latex, melyből külön generálhatunk pdf-et.
- A diagramoknál beállíthatjuk, hogy milyen grafikus reprezentációkat szeretnék a kódunk részeinek összefüggéseiről generálni. A végén Next, majd Run Doxygen!
- A diagrammokhoz érdemes telepítenünk a különálló Graphviz csomagot is, ami igen sokféle reprezentációt lehetővé tesz
- Ehhez menjünk el a <http://www.graphviz.org/> oldalra, telepítsük a csomagot, majd itt jobb oldalon az Expert fülön a Dot topicban pipáljuk be a HAVE\_DOT pipadobozt, és a telepítés útvonalát is írjuk be! (ha rátesszük a path-re a Graphviz-t, ez utóbbira nincs szükség...)



# Doxygen használata

- Ha nem használunk Graphviz-t, akkor is tud egyszerű öröklési diagrammokat generálni a Doxygen.
- Graphvizzel viszont lehet hívási diagrammokat, osztálydiagrammokat stb. is generálni.

**SpiralDiscoveryMethod**  
Generic implementation of SDM with audio example

Main Page | Namespaces ▾ | Classes ▾ | Files ▾ | Examples

SdmTensor > TuningModel

### SdmTensor::TuningModel Class Reference

Implements the SDM tuning model. The class is ultimately a **Tensor** object that inherits through **HosvdTensor** and **HoolTensor**, so that it can be HOSVDed, HOOled and HOOI compensated. More...

```
#include <TuningModel.hpp>
```

Inheritance diagram for SdmTensor::TuningModel:

```
graph BT; SdmTensor_TuningModel[SdmTensor::TuningModel] --> SdmTensor_HoolTensor[SdmTensor::HoolTensor]; SdmTensor_HoolTensor --> SdmTensor_HosvdTensor[SdmTensor::HosvdTensor]; SdmTensor_HosvdTensor --> SdmTensor_Tensor[SdmTensor::Tensor];
```

# Az eredmény

- Látható, hogy ez nem éppen egy szét-dokumentált project, de azért hasznos információkhoz jutottunk így is.

## SpiralDiscoveryMethod

Generic implementation of SDM with audio example

[Main Page](#) [Namespaces](#) [Classes](#) [Files](#) [Examples](#)

### Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

[detail level 1 2 3]

▼ <b>N</b> JAMA	
C Cholesky	
C Eigenvalue	
C LU	
C QR	
C SVD	
▼ <b>N</b> oscpkt	
▼ <b>C</b> Message	
C ArgReader	
C PacketReader	
C PacketWriter	
C PodBytes	
C SockAddr	
C Storage	
C TimeTag	
C UdpSocket	
C Uri	
▼ <b>N</b> SdmTensor	
C HooiTensor	
C HosvdTensor	
C Tensor	
C TuningModel	Implements the SDM tuning model. The class is ultimately a <b>Tensor</b> object that inherits through <b>HosvdTensor</b> and <b>HooiTensor</b> , so that it can be HOSVDed, HOOled and HOOI compensated
▼ <b>N</b> SuperColliderWrapper	
C SoundManager	

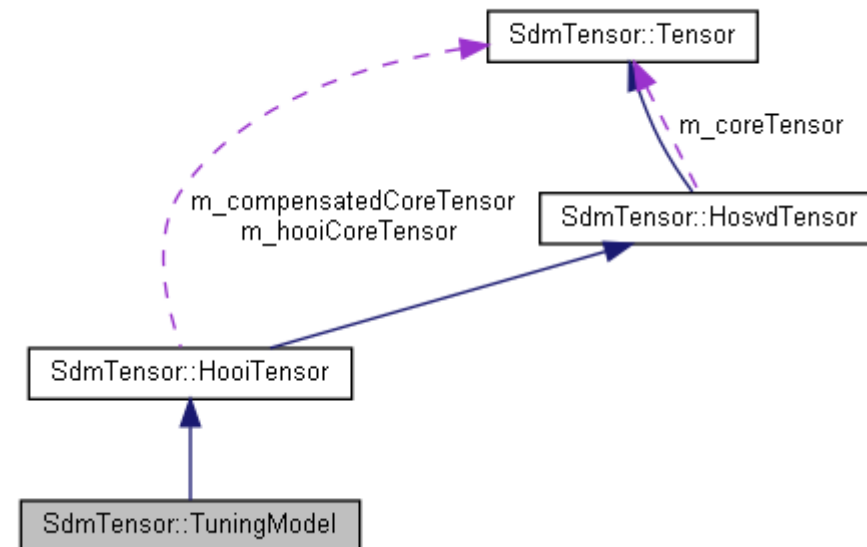


# Mi a helyzet a Graphvizzel?

- Linuxon a package managerek segítségével telepíthető (apt, yum), ahogy Mac-en is (port, brew).
- Szerencsére Windows-ban is létezik installer.
- A Graphviz általánosan használatos szoftver gráfokat tartalmazó ábrák készítéséhez. Tartozik hozzá egy absztrakt nyelv, a "Dot", aminek segítségével az ilyen gráfok szöveggel leírhatóak (ezeket konvertálja kimeneti képekre a Graphviz).
- Még egyszer: ha a Graphviz-t global path-ra tettük, elég csak a HAVE\_DOT pipadobozt bepipálni, path-t nem kell megadni. Viszont ha már futott a Doxywizard, lehetséges, hogy újra kell indítani

# Néhány Graphvizzel generált ábra

Collaboration diagram for SdmTensor::TuningModel:



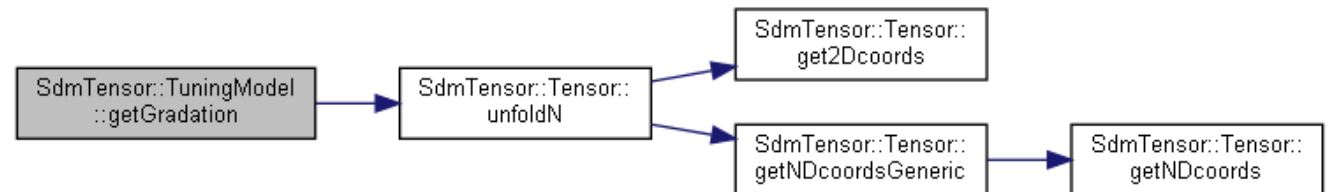
- Ha valami nem világos, kattintsunk a gráf alatti “legend” feliratú linkre...

# Néhány Graphvizzel generált ábra

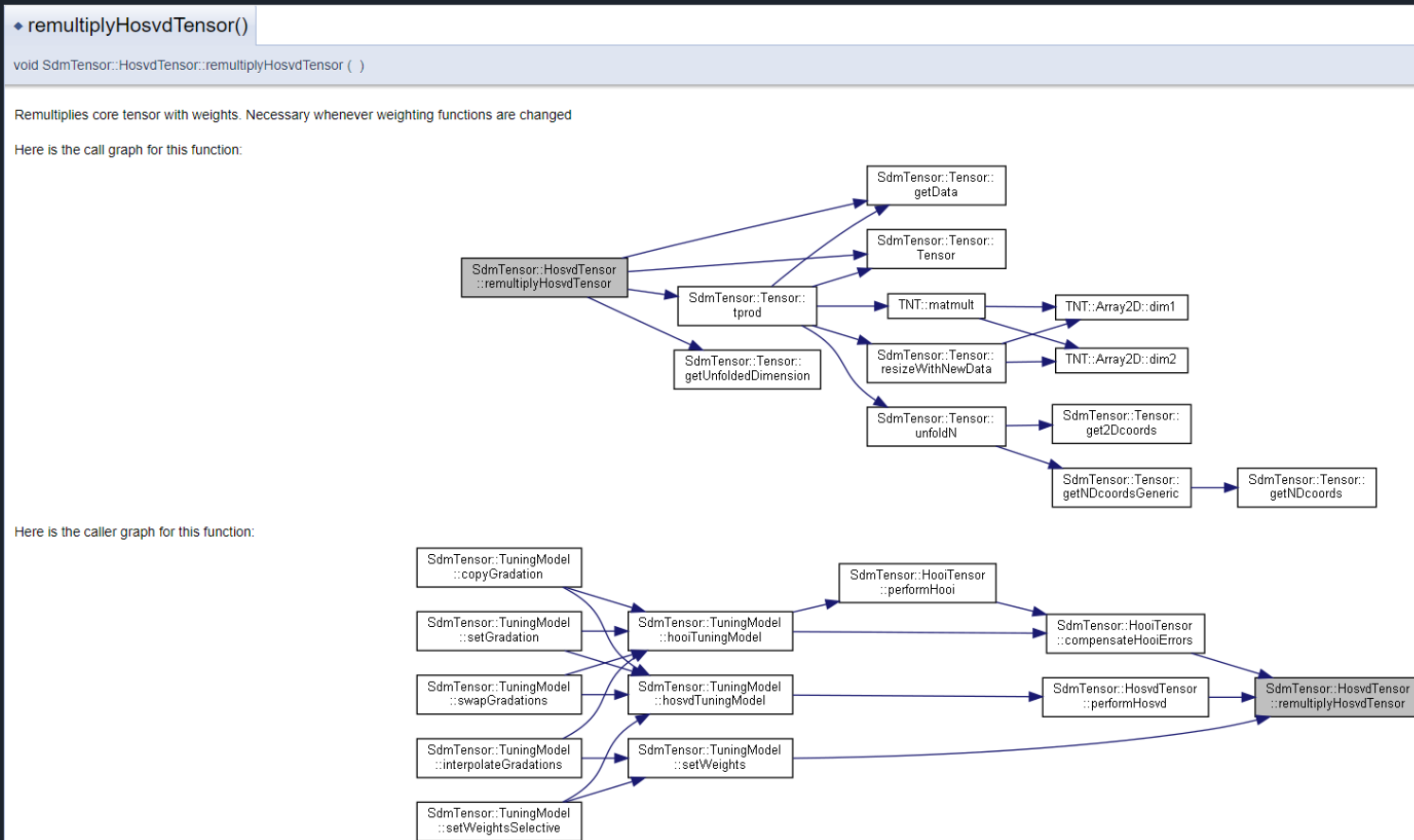
## ◆ getGradation()

```
void SdmTensor::TuningModel::getGradation ( int          gradation,  
                                             std::vector< double > & params  
                                             )
```

Here is the call graph for this function:



# Néhány Graphvizzel generált ábra



- Ahol ennek értelme van, a call graph mellett caller graph-et is le tud generálni a Doxygen / Graphviz

# Kód dokumentálása

- Ezzel pedig gyakorlatilag már mindent tudunk a Doxygenről (majdnem...)
- Csak az van hátra, hogy megtanuljuk, hogyan dokumentálhatjuk a saját kódunkat.
- Őszinte leszek, és bevallom, hogy ezt fejből magam sem tudom.
  - De nem baj, ezért van olyan, hogy dokumentáció! - <https://www.doxygen.nl/manual/docblocks.html>
  - A dokumentáció azért is fontos, mert nem mindegy, hogy milyen nyelven dolgozunk. Python, Fortran, VHDL speciálisabb, de a C/C++ "típusú" nyelveknél azonos az elvárt formátum.

# Kód dokumentálása

```
#include <string>

/*! \brief Point class. Ez egy brief description.
 *      Brief description continued, egészen az új sorig bezárólag.
 *
 * Ez pedig már a részletes leírashoz tartozik.
 */
struct Point {
    /*! \brief x coordinate of Point - ez egy brief leírás... */
    double x;
    double y; /*!< y coordinate of Point - ez egy detailed leírás */
    std::string name; /*!< Ha nincs záró csillag meg per jel, akkor brief leírás
    Point() : x(0), y(0), name("") {} /*!< Default constructor for new[]
    Point(double x, double y) : x(x), y(y), name("") {} /*!< Constructor with 2D params
};
```

# Kód dokumentálása

```
class Path {  
    const int n; //!< pointCount - mivel privat, nem generalja le!  
    Point* pts;  
public:  
    Path(int ptCount); //!< A constructor  
    Path(const Path&); //!< Copy constructor  
    ~Path(); //!< Destructor  
    Path& operator=(const Path&); //!< Copy assignment  
    void setPoint(int index, Point value); // ehhez majd ld. a definicional...  
    Point getPoint(int index) const; //!< getter  
    int getPointCount() const; //!< Ezek mind const metodusok, mert ... miért is?  
    double getLength() const; //!< getter  
    void print(std::ostream&) const; //!< Ennek atadhatjuk pl. az std::cout objektumot  
};
```

# Kód dokumentálása

```
//! Sets point at a given index. Backslash sa stands for 'see also'
/*!
  \param index Index value ranging from 0 to n-1
  \param value A point object
  \return Does not return anything
  \sa Point
*/
void Path::setPoint(int index, Point value) {
    *(pts + index) = value;
}
```

- A teljes példa a leggenerált doksival megtalálható itt:
- <https://github.com/csapoadam/mosze-projects/tree/master/2021>