

Combat Cryptography

Generated by Doxygen 1.9.8

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Transposition::Gridding Class Reference	7
4.1.1 Detailed Description	9
4.1.2 Constructor & Destructor Documentation	9
4.1.2.1 Gridding()	9
4.1.3 Member Function Documentation	9
4.1.3.1 crypto()	9
4.1.3.2 debug()	10
4.1.4 Member Data Documentation	10
4.1.4.1 grid	10
4.1.4.2 process_indicators	10
4.2 Transposition::Inversion Class Reference	10
4.2.1 Detailed Description	13
4.2.2 Constructor & Destructor Documentation	13
4.2.2.1 Inversion()	13
4.2.3 Member Function Documentation	13
4.2.3.1 crypto()	13
4.2.3.2 debug()	14
4.2.3.3 decrypto()	14
4.3 Method Class Reference	14
4.3.1 Detailed Description	16
4.3.2 Member Function Documentation	17
4.3.2.1 if_size_is_valid()	17
4.3.3 Member Data Documentation	17
4.3.3.1 keys	17
4.3.3.2 process_indicators	17
4.4 ProcessIndicator Class Reference	17
4.4.1 Detailed Description	18
4.4.2 Constructor & Destructor Documentation	18
4.4.2.1 ProcessIndicator() [1/2]	18
4.4.2.2 ProcessIndicator() [2/2]	18
4.4.3 Member Function Documentation	19
4.4.3.1 get_char()	19
4.4.4 Member Data Documentation	19

4.4.4.1 group	19
4.4.4.2 position	19
4.4.4.3 process_indicator	19
4.5 String Class Reference	20
4.5.1 Detailed Description	21
4.5.2 Constructor & Destructor Documentation	21
4.5.2.1 String() [1/2]	21
4.5.2.2 String() [2/2]	21
4.5.2.3 ~String()	22
4.5.3 Member Function Documentation	22
4.5.3.1 get_size()	22
4.5.3.2 get_string()	22
4.5.3.3 incremter()	22
4.5.4 Member Data Documentation	22
4.5.4.1 array	22
4.5.4.2 capacity	23
4.5.4.3 size	23
4.6 Transposition Class Reference	23
4.6.1 Detailed Description	23
5 File Documentation	25
5.1 src/Importations.hpp File Reference	25
5.1.1 Macro Definition Documentation	26
5.1.1.1 False	26
5.1.1.2 True	26
5.2 Importations.hpp	27
5.3 src/inversion.hpp File Reference	28
5.4 inversion.hpp	28
5.5 src/main.cpp File Reference	28
5.5.1 Function Documentation	29
5.5.1.1 main()	29
5.6 main.cpp	29
5.7 src/Transposition.hpp File Reference	30
5.7.1 Function Documentation	31
5.7.1.1 apply_inversion()	31
5.8 Transposition.hpp	31
Index	35

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Method	14
Transposition::Griding	7
Transposition::Inversion	10
ProcessIndicator	17
String	20
Transposition	23

Chapter 2

Class Index

2.1 Class List

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

Transposition::Griding	Responsible for grouping the functions of the grids method	7
Transposition::Inversion	Responsible for grouping the functions of the inversion method	10
Method	Abstract class to standardize method development	14
ProcessIndicator	Responsible for represents the process indicators	17
String	Abstraction of our string's characters. Default is that '\0' isnt a char in the string	20
Transposition	Grouping Transposition-based Encryption Methods	23

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

src/Importations.hpp	25
src/inversion.hpp	28
src/main.cpp	28
src/Transposition.hpp	30

Chapter 4

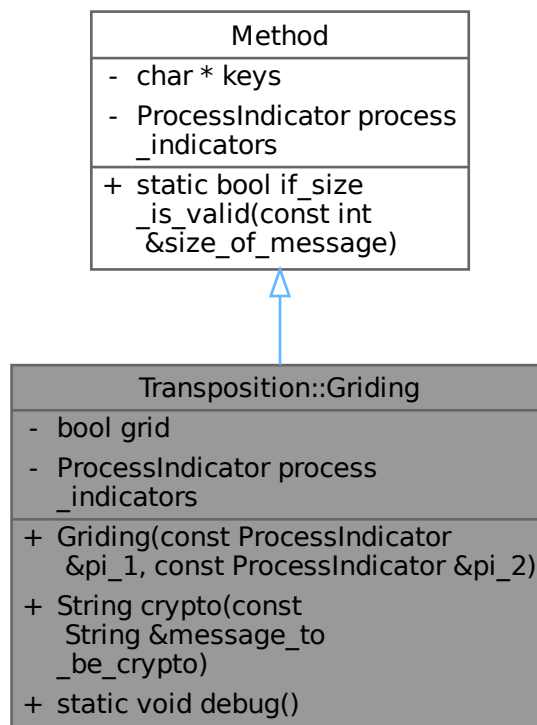
Class Documentation

4.1 Transposition::Gridding Class Reference

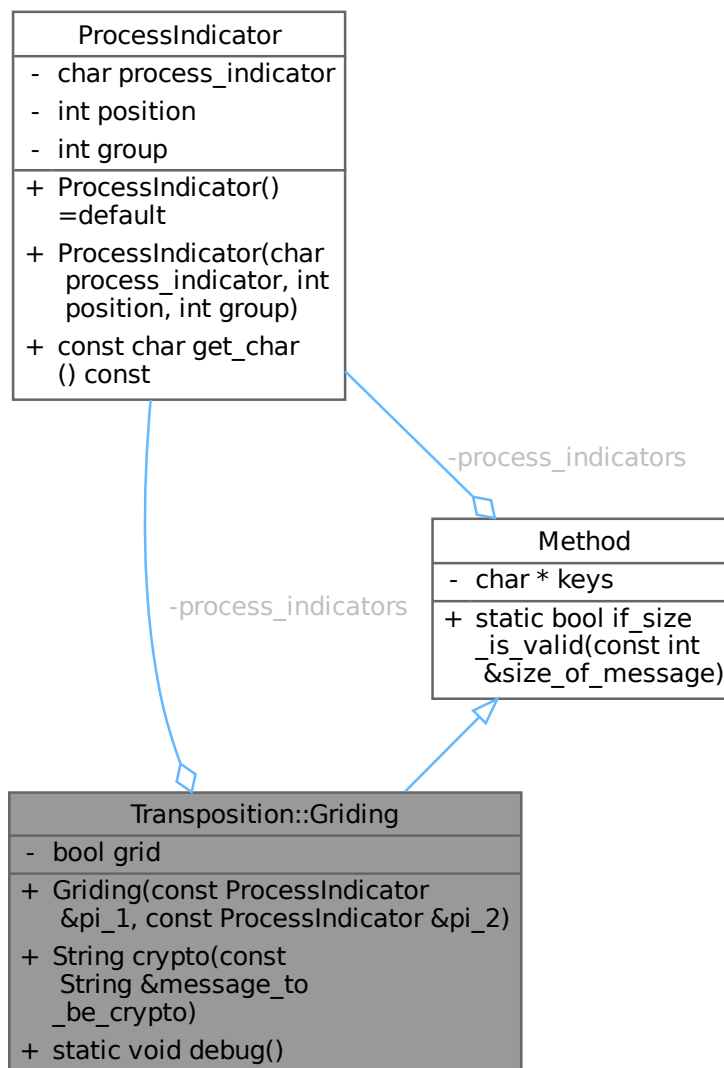
Responsible for grouping the functions of the grids method.

```
#include <Transposition.hpp>
```

Inheritance diagram for Transposition::Gridding:



Collaboration diagram for Transposition::Gridding:



Public Member Functions

- **Gridding** (const [ProcessIndicator](#) &pi_1, const [ProcessIndicator](#) &pi_2)
Funções Inerentes ao Método.
- **String crypto** (const [String](#) &message_to_be_crypto)
Encrypted function.

Static Public Member Functions

- static void **debug** ()
Debug function for tests

Static Public Member Functions inherited from [Method](#)

- static bool [if_size_is_valid](#) (const int &size_of_message)
Function that checks if the message size is valid

Private Attributes

- bool [grid](#) [30]
- [ProcessIndicator](#) [process_indicators](#) [2]

4.1.1 Detailed Description

Responsible for grouping the functions of the grids method.

Definition at line 81 of file [Transposition.hpp](#).

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Griding()

```
Transposition::Griding::Griding (
    const ProcessIndicator & pi_1,
    const ProcessIndicator & pi_2 ) [inline]
```

Funções Inerentes ao Método.

Constructor

Parameters

array_to_process_indicators	Array reporting process, position and group indicators for each indicator
---	---

Definition at line 102 of file [Transposition.hpp](#).

4.1.3 Member Function Documentation

4.1.3.1 crypto()

```
String Transposition::Griding::crypto (
    const String & message_to_be_crypto ) [inline]
```

Encrypted function.

Parameters

message_to_be_crypto	It is necessary to have 5m - 2 characters.
--------------------------------------	--

Definition at line 116 of file [Transposition.hpp](#).

4.1.3.2 debug()

```
static void Transposition::Gridding::debug ( ) [inline], [static]
```

Debug function for tests

Definition at line 138 of file [Transposition.hpp](#).

4.1.4 Member Data Documentation

4.1.4.1 grid

```
bool Transposition::Gridding::grid[30] [private]
```

Initial value:

```
= {  
    1, 0, 1, 1, 1, 0, 1, 1, 1, 1,  
    0, 1, 0, 1, 0, 1, 0, 1, 0, 1,  
    1, 0, 1, 0, 1, 1, 1, 0, 1, 0,  
}
```

Definition at line 84 of file [Transposition.hpp](#).

4.1.4.2 process_indicators

```
ProcessIndicator Transposition::Gridding::process_indicators[2] [private]
```

Definition at line 90 of file [Transposition.hpp](#).

The documentation for this class was generated from the following file:

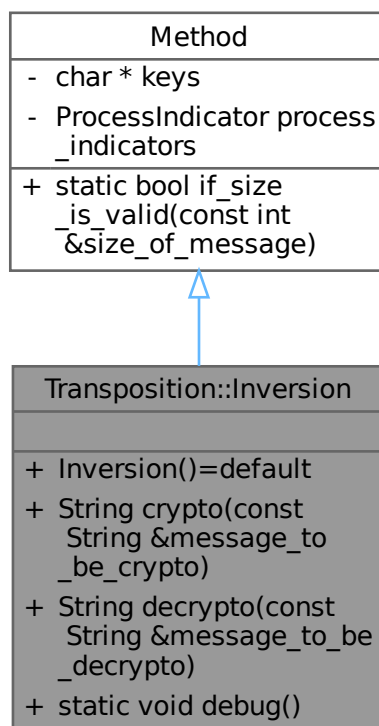
- [src/Transposition.hpp](#)

4.2 Transposition::Inversion Class Reference

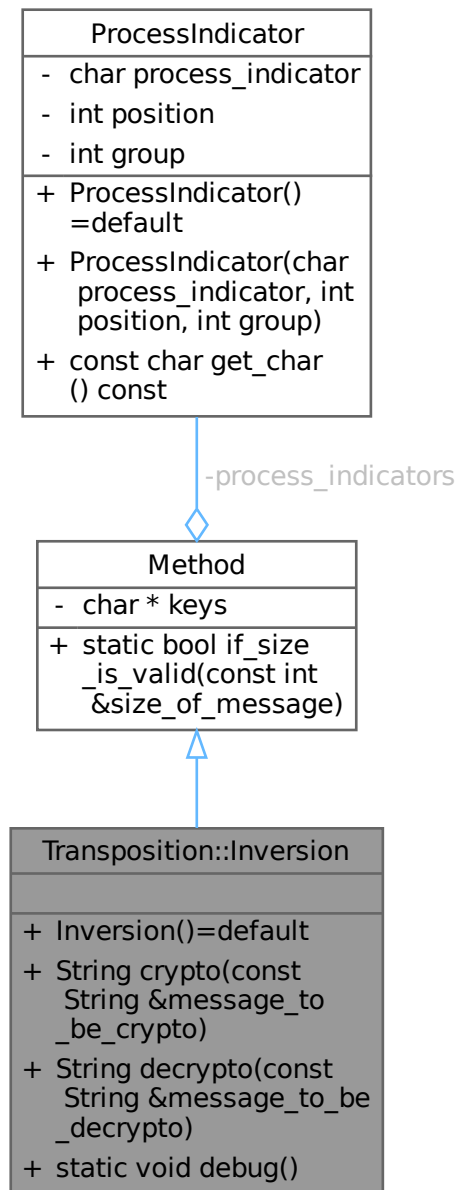
Responsible for grouping the functions of the inversion method.

```
#include <Transposition.hpp>
```

Inheritance diagram for Transposition::Inversion:



Collaboration diagram for Transposition::Inversion:



Public Member Functions

- [Inversion](#) ()=default
Constructor
- [String crypto](#) (const [String](#) &message_to_be_crypto)
Encryption function.
- [String decrypto](#) (const [String](#) &message_to_be_decrypto)
Decryption function.

Static Public Member Functions

- static void [debug](#) ()
Debug function.

Static Public Member Functions inherited from [Method](#)

- static bool [if_size_is_valid](#) (const int &size_of_message)
Function that checks if the message size is valid

4.2.1 Detailed Description

Responsible for grouping the functions of the inversion method.

Definition at line 15 of file [Transposition.hpp](#).

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Inversion()

```
Transposition::Inversion::Inversion ( ) [default]
```

Constructor

4.2.3 Member Function Documentation

4.2.3.1 crypto()

```
String Transposition::Inversion::crypto (
    const String & message_to_be_crypto ) [inline]
```

Encryption function.

Parameters

<i>message_to_be_crypto</i>	Self Explained
-----------------------------	----------------

Returns

message_crypted

Definition at line 29 of file [Transposition.hpp](#).

4.2.3.2 debug()

```
static void Transposition::Inversion::debug ( ) [inline], [static]
```

Debug function.

Definition at line 68 of file [Transposition.hpp](#).

4.2.3.3 decrypto()

```
String Transposition::Inversion::decrypto (
    const String & message_to_be_decrypto ) [inline]
```

Decryption function.

Parameters

<i>message_to_be_decrypto</i>	Self Explained
-------------------------------	----------------

Returns

message_decrypted

Note that, by isomorphism, it is the same encryption function.

Definition at line 56 of file [Transposition.hpp](#).

The documentation for this class was generated from the following file:

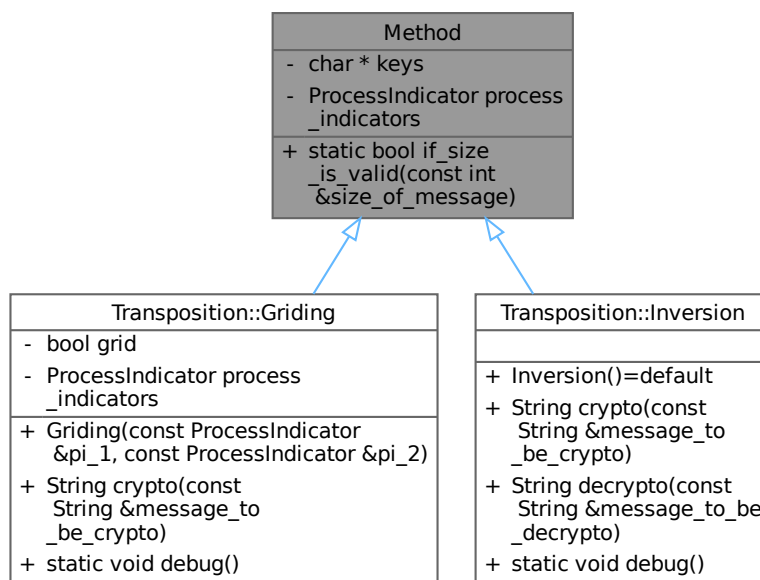
- [src/Transposition.hpp](#)

4.3 Method Class Reference

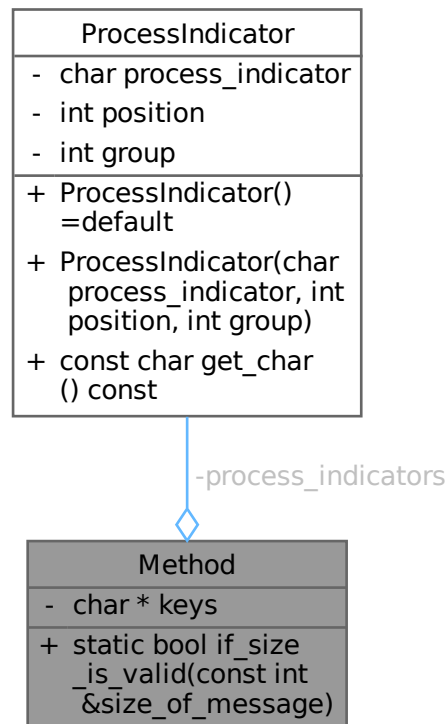
Abstract class to standardize method development.

```
#include <Importations.hpp>
```

Inheritance diagram for Method:



Collaboration diagram for Method:



Static Public Member Functions

- static bool [if_size_is_valid](#) (const int &size_of_message)
Function that checks if the message size is valid

Private Attributes

- char * [keys](#) [2] = {nullptr, nullptr}
- [ProcessIndicator process_indicators](#) [2]

4.3.1 Detailed Description

Abstract class to standardize method development.

There's a pattern for method's class:

- `cripto` Where a message is cripto.
- `decripto` Where a message is decripto
- `debug` Where is possible to see a example

Definition at line [149](#) of file [Importations.hpp](#).

4.3.2 Member Function Documentation

4.3.2.1 if_size_is_valid()

```
static bool Method::if_size_is_valid (
    const int & size_of_message ) [inline], [static]
```

Function that checks if the message size is valid

Definition at line 163 of file [Importations.hpp](#).

4.3.3 Member Data Documentation

4.3.3.1 keys

```
char* Method::keys[2] = {nullptr, nullptr} [private]
```

Definition at line 152 of file [Importations.hpp](#).

4.3.3.2 process_indicators

```
ProcessIndicator Method::process_indicators[2] [private]
```

Definition at line 153 of file [Importations.hpp](#).

The documentation for this class was generated from the following file:

- [src/Importations.hpp](#)

4.4 ProcessIndicator Class Reference

Responsible for represents the process indicators.

```
#include <Importations.hpp>
```

Collaboration diagram for ProcessIndicator:

ProcessIndicator
<ul style="list-style-type: none"> - char process_indicator - int position - int group
<ul style="list-style-type: none"> + ProcessIndicator() =default + ProcessIndicator(char process_indicator, int position, int group) + const char get_char() const

Public Member Functions

- [ProcessIndicator](#) ()=default
- [ProcessIndicator](#) (char process_indicator, int position, int group)
Constructor that will provide creation and validation methods.
- [const char get_char](#) () const

Private Attributes

- [char process_indicator](#)
- [int position](#)
- [int group](#)

4.4.1 Detailed Description

Responsible for represents the process indicators.

Definition at line 96 of file [Importations.hpp](#).

4.4.2 Constructor & Destructor Documentation

4.4.2.1 ProcessIndicator() [1/2]

```
ProcessIndicator::ProcessIndicator ( ) [default]
```

4.4.2.2 ProcessIndicator() [2/2]

```
ProcessIndicator::ProcessIndicator (
    char process_indicator,
    int position,
    int group ) [inline]
```

Constructor that will provide creation and validation methods.

Parameters

<i>process_indicator</i>	Character uppercase alphabetic
<i>position</i>	Character's position in the pool
<i>group</i>	Character1's group in the message, can be negative.

A process indicator is valid when it is a capital alphabetic letter.

Definition at line 114 of file [Importations.hpp](#).

4.4.3 Member Function Documentation

4.4.3.1 get_char()

```
const char ProcessIndicator::get_char ( ) const [inline]
```

Definition at line 134 of file [Importations.hpp](#).

4.4.4 Member Data Documentation

4.4.4.1 group

```
int ProcessIndicator::group [private]
```

Definition at line 101 of file [Importations.hpp](#).

4.4.4.2 position

```
int ProcessIndicator::position [private]
```

Definition at line 100 of file [Importations.hpp](#).

4.4.4.3 process_indicator

```
char ProcessIndicator::process_indicator [private]
```

Definition at line 99 of file [Importations.hpp](#).

The documentation for this class was generated from the following file:

- [src/Importations.hpp](#)

4.5 String Class Reference

Abstraction of our string's characters. Default is that '\0' isn't a char in the string.

```
#include <Importations.hpp>
```

Collaboration diagram for String:

String
- char * array
- int size
- int capacity
+ String(const int &size _to_be_reserve=1)
+ String(const char *conj_de_caracteres)
+ ~String()
+ const int & get_size () const
+ const char * get_string () const
+ void incrementer(bool to_realloc=True)

Public Member Functions

- [String](#) (const int &size_to_be_reserve=1)
Number of char's slots available
- [String](#) (const char *conj_de_caracteres)
Character pointer-based constructor.
- [~String](#) ()
Default Destructor
- const int & [get_size](#) () const
Getter the size.
- const char * [get_string](#) () const
Getter the pointer character.
- void [incrementer](#) (bool to_realloc=[True](#))
Increments the size of the character array by one and reallocates to have one more unit available

Private Attributes

- char * [array](#) = nullptr
- int [size](#) = 0
Pointer to characters
- int [capacity](#) = 0
Number of characters valids

4.5.1 Detailed Description

Abstraction of our string's characters. Default is that '\0' isn't a char in the string.

It was preferred to use a proprietary class to avoid using 'a toolbox too big for something small'

Definition at line [16](#) of file [Importations.hpp](#).

4.5.2 Constructor & Destructor Documentation**4.5.2.1 String() [1/2]**

```
String::String (
    const int & size_to_be_reserve = 1 )    [inline]
```

Number of char's slots available

Default Constructor

Parameters

--	--

Definition at line [29](#) of file [Importations.hpp](#).

4.5.2.2 String() [2/2]

```
String::String (
    const char * conj_de_caracteres )    [inline]
```

Character pointer-based constructor.

Parameters

<i>conj_de_caracteres</i>	Const pointer for characters
---------------------------	------------------------------

Definition at line [41](#) of file [Importations.hpp](#).

4.5.2.3 ~String()

```
String::~~String ( ) [inline]
```

Default Destructor

Definition at line 63 of file [Importations.hpp](#).

4.5.3 Member Function Documentation

4.5.3.1 get_size()

```
const int & String::get_size ( ) const [inline]
```

Getter the size.

Definition at line 72 of file [Importations.hpp](#).

4.5.3.2 get_string()

```
const char * String::get_string ( ) const [inline]
```

Getter the pointer character.

Definition at line 79 of file [Importations.hpp](#).

4.5.3.3 incrementer()

```
void String::incrementer (
    bool to_realloc = True ) [inline]
```

Increments the size of the character array by one and reallocates to have one more unit available

Parameters

<i>to_realloc</i>	If you need to relocate, default is True.
-------------------	---

Definition at line 87 of file [Importations.hpp](#).

4.5.4 Member Data Documentation

4.5.4.1 array

```
char* String::array = nullptr [private]
```

Definition at line 19 of file [Importations.hpp](#).

4.5.4.2 capacity

```
int String::capacity = 0 [private]
```

Number of characters valids

Definition at line 21 of file [Importations.hpp](#).

4.5.4.3 size

```
int String::size = 0 [private]
```

Pointer to characters

Definition at line 20 of file [Importations.hpp](#).

The documentation for this class was generated from the following file:

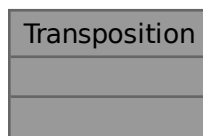
- [src/Importations.hpp](#)

4.6 Transposition Class Reference

Grouping Transposition-based Encryption Methods.

```
#include <Transposition.hpp>
```

Collaboration diagram for Transposition:



Classes

- class [Griding](#)
Responsible for grouping the functions of the grids method.
- class [Inversion](#)
Responsible for grouping the functions of the inversion method.

4.6.1 Detailed Description

Grouping Transposition-based Encryption Methods.

Definition at line 9 of file [Transposition.hpp](#).

The documentation for this class was generated from the following file:

- [src/Transposition.hpp](#)

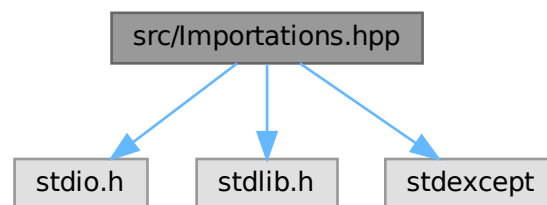
Chapter 5

File Documentation

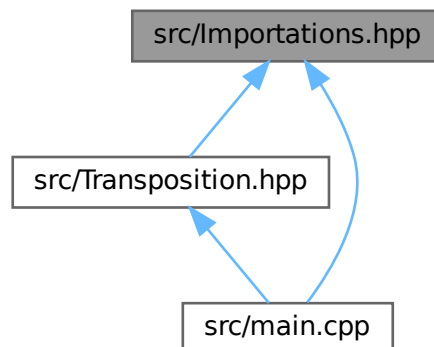
5.1 src/Importations.hpp File Reference

```
#include <stdio.h>  
#include <stdlib.h>  
#include <stdexcept>
```

Include dependency graph for Importations.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [String](#)
Abstraction of our string's characters. Default is that '\0' isnt a char in the string.
- class [ProcessIndicator](#)
Responsible for represents the process indicators.
- class [Method](#)
Abstract class to standardize method development.

Macros

- `#define True true`
- `#define False false`

5.1.1 Macro Definition Documentation

5.1.1.1 False

```
#define False false
```

Definition at line 5 of file [Importations.hpp](#).

5.1.1.2 True

```
#define True true
```

Definition at line 4 of file [Importations.hpp](#).

5.2 Importations.hpp

[Go to the documentation of this file.](#)

```

00001 #ifndef IMPORTATIONS_H
00002 #define IMPORTATIONS_H
00003
00004 #define True true
00005 #define False false
00006
00007 #include <stdio.h>
00008 #include <stdlib.h>
00009 #include <stdexcept>
00010
00016 class String {
00017 private:
00018     char* array = nullptr;
00019     int size = 0;
00020     int capacity = 0;
00021
00022 public:
00023     String(
00024         const int& size_to_be_reserve = 1
00025     ){
00026         this->array = (char*)malloc(size_to_be_reserve * sizeof(char));
00027         this->size = 0;
00028     }
00029
00030     String(
00031         const char* conj_de_caracteres
00032     ){
00033         this->array = (char*)malloc(sizeof(char));
00034
00035         while(
00036             conj_de_caracteres[this->size] != '\0'
00037         ){
00038             array[this->size] = conj_de_caracteres[this->size];
00039             this->size++;
00040
00041             array = (char*)realloc(array, (this->size + 1) * sizeof(char));
00042             this->capacity++;
00043         }
00044         array[this->size] = '\0';
00045     }
00046
00047 ~String(){ if(array){ free(array); } }
00048
00049 inline
00050 const int&
00051 get_size() const { return this->size; }
00052
00053 inline
00054 const char*
00055 get_string() const { return this->array; }
00056
00057 inline
00058 void
00059 incrementer(bool to_realloc = True){
00060     this->size++;
00061     if(to_realloc){ this->array = (char*)realloc(this->array, (this->size + 1) * sizeof(char));
00062         this->capacity++;}
00063 }
00064 };
00065
00066 class ProcessIndicator {
00067 private:
00068     char process_indicator;
00069     int position;
00070     int group;
00071
00072 public:
00073     ProcessIndicator() = default;
00074
00075     ProcessIndicator(
00076         char process_indicator,
00077         int position,
00078         int group
00079     ){
00080

```

```

00120         // Verificamos se é válido
00121         if (
00122             process_indicator < 'A' || process_indicator > 'Z' || position < 0 || position > 4
00123         ){
00124
00125             throw std::invalid_argument("Indicador de Processo inválido!");
00126         }
00127
00128         this->process_indicator = process_indicator;
00129         this->position = position;
00130         this->group = group;
00131     }
00132
00133     const char
00134     get_char() const { return this->process_indicator; }
00135 };
00136
00149 class Method {
00150 private:
00151
00152     char* keys[2] = {nullptr, nullptr};
00153     ProcessIndicator process_indicators[2];
00154
00155 public:
00156
00160     inline
00161     static
00162     bool
00163     if_size_is_valid(
00164         const int& size_of_message
00165     ){
00166
00167         return !( (size_of_message - 2) % 5 );
00168     }
00169
00170 };
00171
00172 #endif // IMPORTATIONS_H

```

5.3 src/inversion.hpp File Reference

5.4 inversion.hpp

[Go to the documentation of this file.](#)

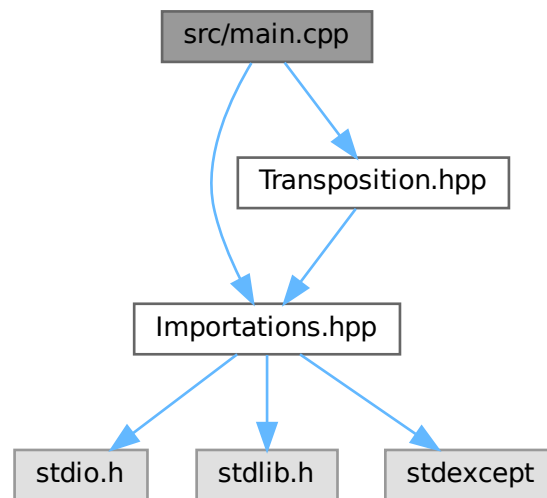
5.5 src/main.cpp File Reference

```

#include "Importations.hpp"
#include "Transposition.hpp"

```


Include dependency graph for main.cpp:



Functions

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

5.5.1 Function Documentation

5.5.1.1 main()

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

Definition at line 5 of file [main.cpp](#).

5.6 main.cpp

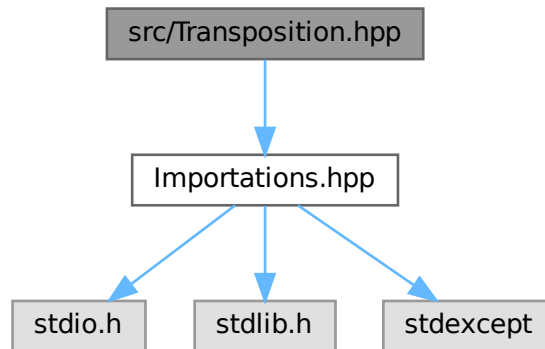
[Go to the documentation of this file.](#)

```
00001 #include "Importations.hpp"
00002 #include "Transposition.hpp"
00003
00004 int
00005 main(
00006     int argc,
00007     char* argv[]
00008 ){
00009     Transposition::Gridding::debug();
00010
00011
00012
00013
00014     return 0;
00015 }
```

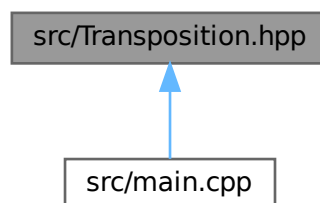
5.7 src/Transposition.hpp File Reference

```
#include "Importations.hpp"
```

Include dependency graph for Transposition.hpp:



This graph shows which files directly or indirectly include this file:



Classes

- class [Transposition](#)
Grouping Transposition-based Encryption Methods.
- class [Transposition::Inversion](#)
Responsible for grouping the functions of the inversion method.
- class [Transposition::Gridding](#)
Responsible for grouping the functions of the grids method.

Functions

- [String](#) * [apply_inversion](#) ([String](#) *str)
Apply the method of inversion on the string.

5.7.1 Function Documentation

5.7.1.1 apply_inversion()

```
String * apply_inversion (
    String * str )
```

Apply the method of inversion on the string.

Parameters

<i>str</i>	String to be cript or decrypt
------------	-------------------------------

Remember that this method is isomorphic.

Definition at line 166 of file [Transposition.hpp](#).

5.8 Transposition.hpp

[Go to the documentation of this file.](#)

```
00001 #ifndef TRANSPOSITION_H
00002 #define TRANSPOSITION_H
00003
00004 #include "Importations.hpp"
00005
00009 class Transposition {
00010 public:
00011
00015     class Inversion : public Method {
00016     public:
00017
00021         Inversion() = default;
00022
00028         String
00029         crypto(
00030             const String& message_to_be_crypto
00031         ){
00032
00033             String result;
00034
00035             while (
00036                 result.get_size() < message_to_be_crypto.get_size()
00037             ){
00038
00039                 result.array[result.get_size()] =
00040                 message_to_be_crypto.array[message_to_be_crypto.get_size() - 1 - result.get_size()];
00041                 result.incrementer();
00042             }
00043             result.array[result.get_size()] = '\0';
00044
00045             return result;
00046         }
00047
00055         String
00056         decrypto(
00057             const String& message_to_be_decrypto
00058         ){
00059
00060             return crypto(message_to_be_decrypto);
00061         }
00062
00066         static
00067         void
00068         debug() {
00069
00070             Inversion cypher;
00071             String exemplo("Meu nome eh Matheus");
00072             printf("String de Teste: \t'%s'", exemplo.get_string());
00073             printf("\nApós aplicar crypto: \t'%s'", cypher.crypto(exemplo).get_string());
```

```

00074         printf("\nVoltando:          \t'%s'\n",
cypher.decrypto(cypher.crypto(exemplo)).get_string());
00075     }
00076 };
00077
00081     class Griding : public Method {
00082     private:
00083
00084         bool grid[30] = {
00085             1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
00086             0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
00087             1, 0, 1, 0, 1, 1, 1, 0, 1, 0,
00088         };
00089
00090         ProcessIndicator process_indicators[2];
00091
00095     public:
00096
00097         Griding(
00102             const ProcessIndicator& pi_1,
00103             const ProcessIndicator& pi_2
00104         ){
00105
00106             process_indicators[0] = std::move(pi_1);
00107             process_indicators[1] = std::move(pi_2);
00108         }
00109
00110         String
00115         crypto(
00116             const String& message_to_be_crypto
00117         ){
00118
00119             if( !Method::if_size_is_valid(message_to_be_crypto.get_size()) ){ throw
00120                 std::invalid_argument("Mensagem não tem tamanho válido"); }
00121
00122             // -----
00123             // Iniciamos aplicação do algoritmo
00124             String temp;
00125
00126             return temp;
00127         }
00128     };
00129
00130
00131
00132     static
00136     void
00137     debug(){
00138         Griding exemplo(
00139             {'A', 1, 2},
00140             {'B', 2, 1}
00141         );
00142     };
00143
00144
00145
00146
00147
00148
00149
00150     }
00151
00152
00153
00154
00155     };
00156 };
00157
00165 String*
00166 apply_inversion(
00167     String* str
00168 ){
00169
00170     String* output = (String*)malloc(sizeof(String));
00171     output->array = (char*)malloc(str->size * sizeof(char)); // Already reserve
00172     output->size = 0;
00173
00174     while(
00175         output->size != str->size
00176     ){
00177
00178         output->array[output->size] = str->array[str->size - 1 - output->size];
00179         output->size++;
00180     }
00181
00182     output->array[output->size] = '\0';

```

```
00183
00184     return output;
00185 }
00186
00187 #endif // TRANSPOSITION_H
```


Index

- ~String
 - String, [21](#)
- apply_inversion
 - Transposition.hpp, [31](#)
- array
 - String, [22](#)
- capacity
 - String, [22](#)
- crypto
 - Transposition::Gridding, [9](#)
 - Transposition::Inversion, [13](#)
- debug
 - Transposition::Gridding, [10](#)
 - Transposition::Inversion, [13](#)
- decrypto
 - Transposition::Inversion, [14](#)
- False
 - Importations.hpp, [26](#)
- get_char
 - ProcessIndicator, [19](#)
- get_size
 - String, [22](#)
- get_string
 - String, [22](#)
- grid
 - Transposition::Gridding, [10](#)
- Gridding
 - Transposition::Gridding, [9](#)
- group
 - ProcessIndicator, [19](#)
- if_size_is_valid
 - Method, [17](#)
- Importations.hpp
 - False, [26](#)
 - True, [26](#)
- incrementer
 - String, [22](#)
- Inversion
 - Transposition::Inversion, [13](#)
- keys
 - Method, [17](#)
- main
 - main.cpp, [29](#)
- main.cpp
 - main, [29](#)
- Method, [14](#)
 - if_size_is_valid, [17](#)
 - keys, [17](#)
 - process_indicators, [17](#)
- position
 - ProcessIndicator, [19](#)
- process_indicator
 - ProcessIndicator, [19](#)
- process_indicators
 - Method, [17](#)
 - Transposition::Gridding, [10](#)
- ProcessIndicator, [17](#)
 - get_char, [19](#)
 - group, [19](#)
 - position, [19](#)
 - process_indicator, [19](#)
 - ProcessIndicator, [18](#)
- size
 - String, [23](#)
- src/Importations.hpp, [25](#), [27](#)
- src/inversion.hpp, [28](#)
- src/main.cpp, [28](#), [29](#)
- src/Transposition.hpp, [30](#), [31](#)
- String, [20](#)
 - ~String, [21](#)
 - array, [22](#)
 - capacity, [22](#)
 - get_size, [22](#)
 - get_string, [22](#)
 - incrementer, [22](#)
 - size, [23](#)
 - String, [21](#)
- Transposition, [23](#)
- Transposition.hpp
 - apply_inversion, [31](#)
- Transposition::Gridding, [7](#)
 - crypto, [9](#)
 - debug, [10](#)
 - grid, [10](#)
 - Gridding, [9](#)
 - process_indicators, [10](#)
- Transposition::Inversion, [10](#)
 - crypto, [13](#)
 - debug, [13](#)
 - decrypto, [14](#)

Inversion, [13](#)
True
Importations.hpp, [26](#)