Combat Crytography

Generated by Doxygen 1.9.8

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

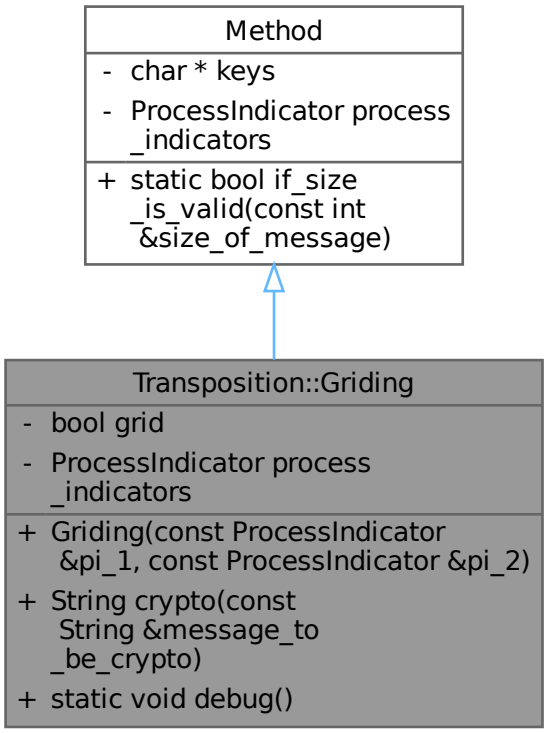
Class Documentation

4.1 Transposition::Griding Class Reference

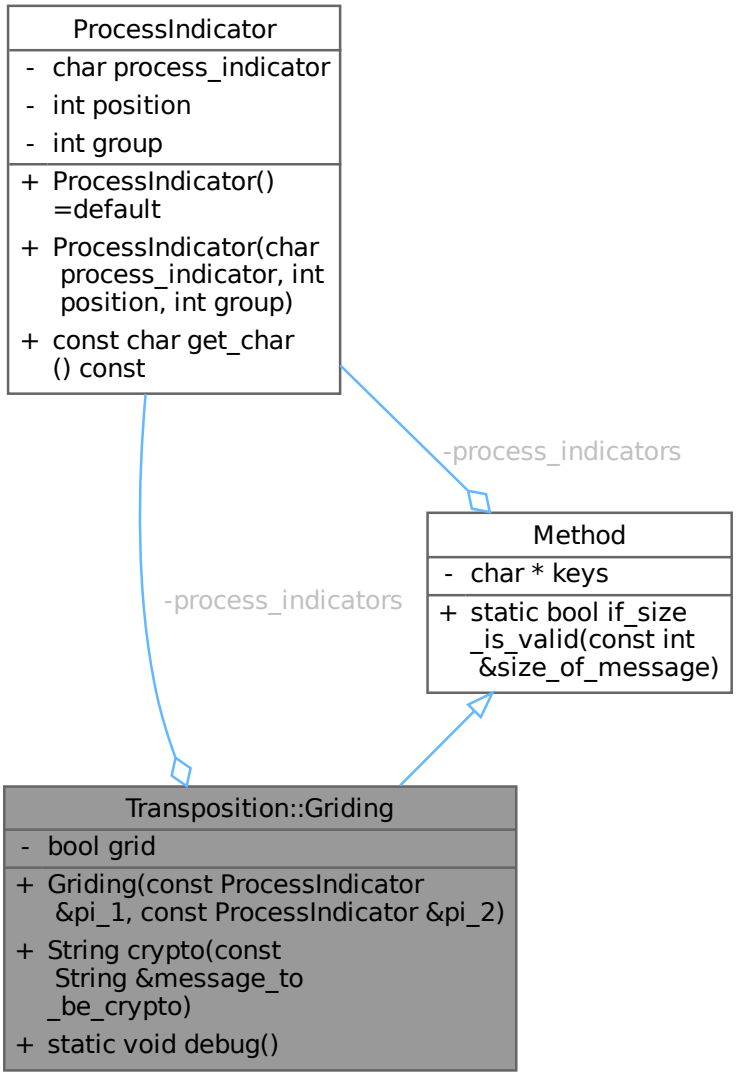
Responsible for grouping the functions of the grids method.

`#include <Transposition.hpp>`

Inheritance diagram for Transposition::Griding:



Collaboration diagram for Transposition::Griding:



Public Member Functions

- Griding (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::Griding::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::Griding::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::Griding::process_indicators[2]  [private]
```

Definition at line 90 of file Transposition.hpp.

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

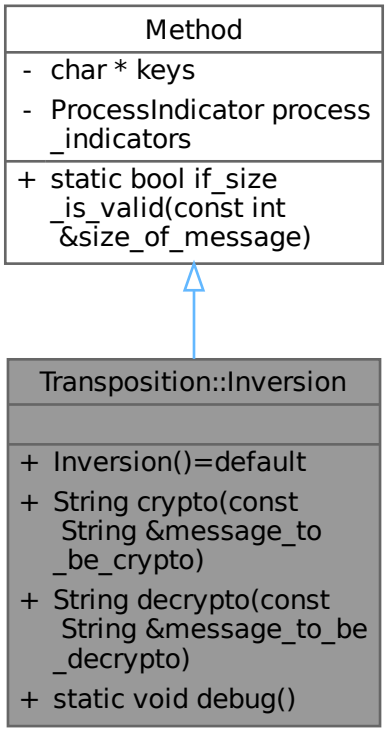
- src/Transposition.hpp

4.2 Transposition::Inversion Class Reference

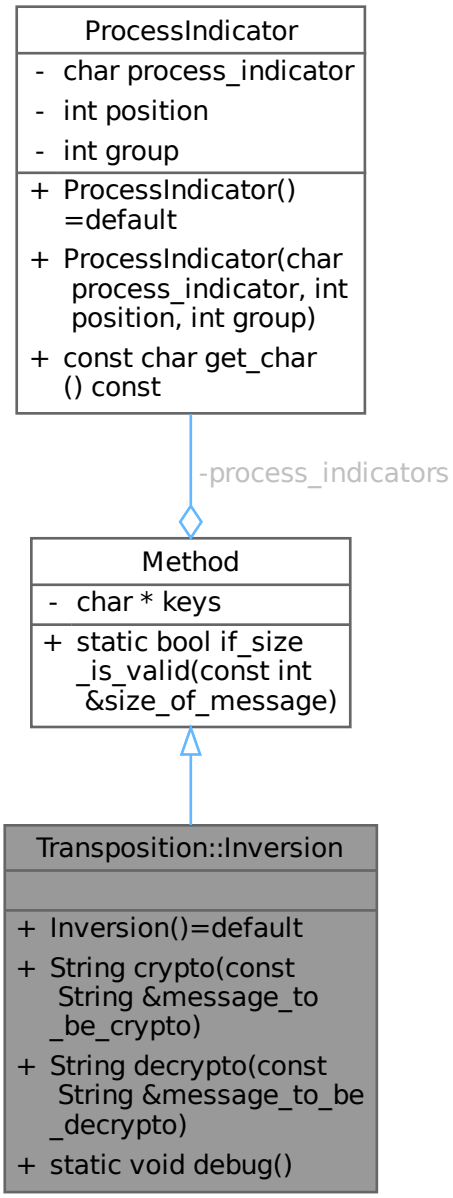
Responsible for grouping the functions of the inversion method.

```
#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

message_to_be_crypto	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

message_to_be_decrypto	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:

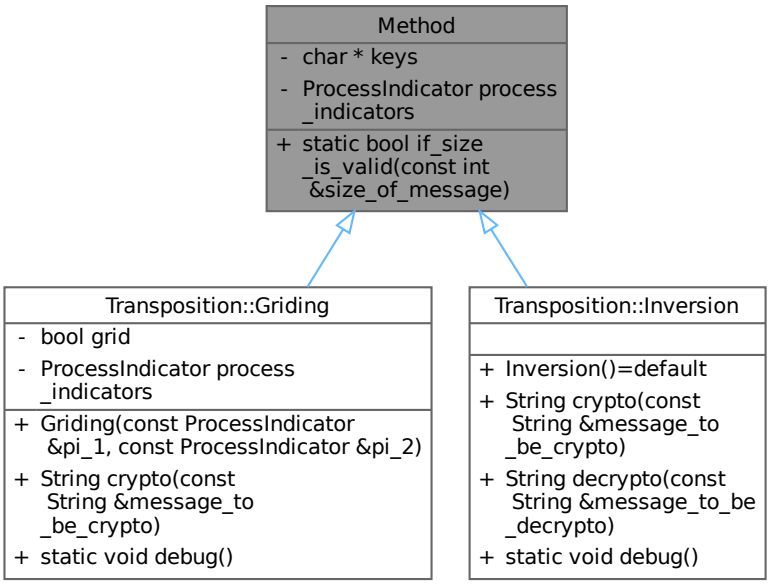
- src/Transposition.hpp

4.3 Method Class Reference

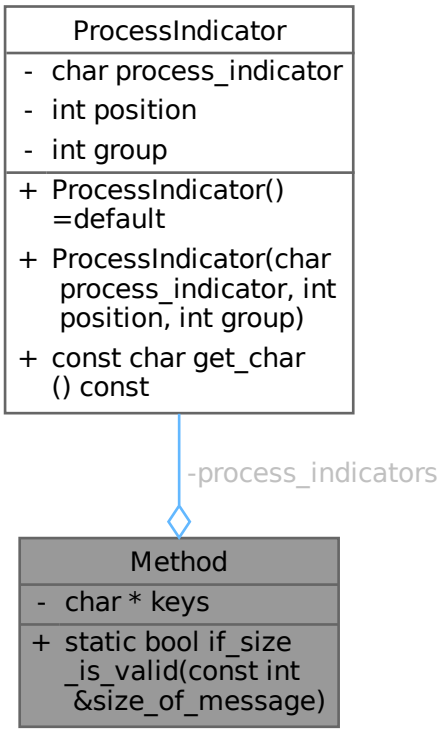
Abstract class to standardize method development.

```
#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:

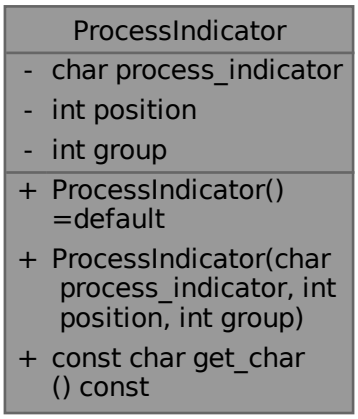
- src/Importations.hpp

4.4 ProcessIndicator Class Reference

Responsible for represents the process indicators.

`#include <Importations.hpp>`

Collaboration diagram for ProcessIndicator:



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

process_indicator	Character uppercase alphabetic
position	Character's position in the pool
group	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:

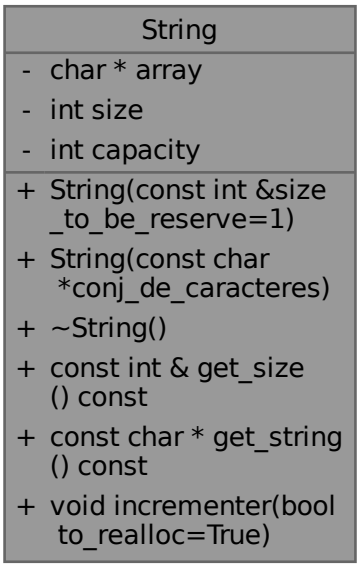
- src/Importations.hpp

4.5 String Class Reference

Abstration of our string's characters. Default is that '\0' isnt a char in the string.

`#include <Importations.hpp>`

Collaboration diagram for String:



Public Member Functions

- String (const int &size_to_be_reserve=1)

  Number of char's slots avaiable
- 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

Abstration of our string's characters. Default is that '\0' isnt 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 avaiable

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

conj_de_caracteres	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

to_realloc	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:

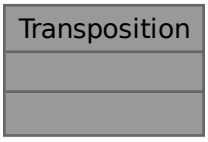
- src/Importations.hpp

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

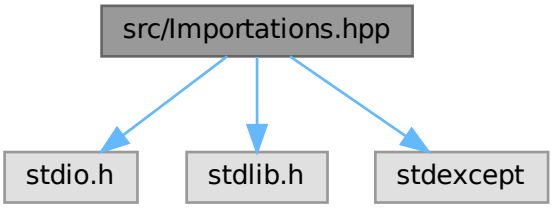
Chapter 5

File Documentation

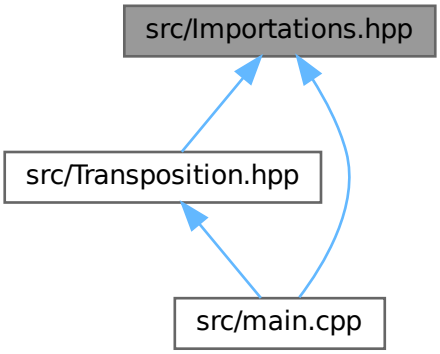
5.1 src/Importations.hpp File Reference

```
#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

  Abstration 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
00019     char* array = nullptr;
00020     int size    =       0;
00021     int capacity = 0;
00022
00023 public:
00024
00029     String(
00030         const int& size_to_be_reserve = 1
00031     ){
00032
00033         this->array = (char*)malloc(size_to_be_reserve * sizeof(char));
00034         this->size  = 0;
00035     }
00036
00041     String(
00042         const char* conj_de_caracteres
00043     ){
00044
00045         this->array = (char*)malloc(sizeof(char));
00046
00047         while(
00048             conj_de_caracteres[this->size] != '\0'
00049         ){
00050
00051             array[this->size] = conj_de_caracteres[this->size];
00052             this->size++;
00053
00054             array = (char*)realloc(array, (this->size + 1) * sizeof(char));
00055             this->capacity++;
00056         }
00057         array[this->size] = '\0';
00058     }
00059
00063     ~String(){ if(array){ free(array); } }
00064
00066
00070     inline
00071     const int&
00072     get_size() const { return this->size; }
00073
00077     inline
00078     const char*
00079     get_string() const { return this->array; }
00080
00085     inline
00086     void
00087     incrementer(bool to_realloc = True){
00088         this->size++;
00089         if(to_realloc){ this->array = (char*)realloc(this->array, (this->size + 1) * sizeof(char));
      this->capacity++;}
00090     }
00091 };
00092
00096 class ProcessIndicator {
00097 private:
00098
00099     char process_indicator;
00100     int position;
00101     int group;
00102 public:
00103
00104     ProcessIndicator() = default;
00105
00114     ProcessIndicator(
00115         char process_indicator,
00116         int  position,
00117         int  group
00118     ){
00119
```

```
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

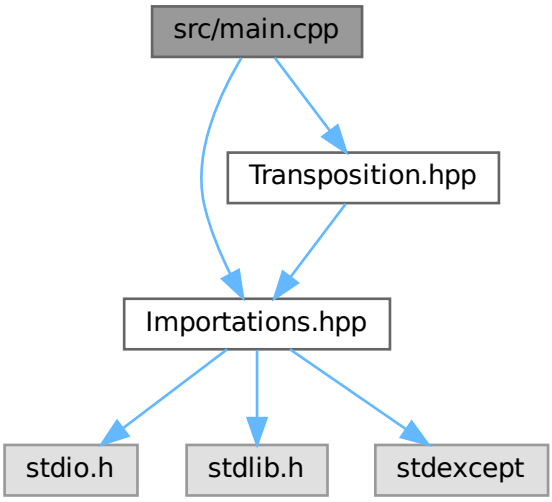
5.4 inversion.hpp

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

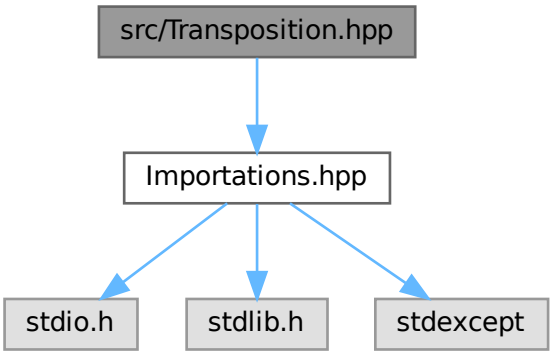
Go to the documentation of this file.

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

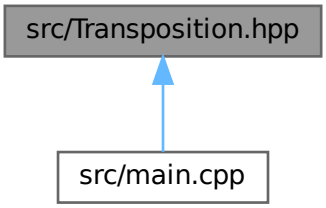
5.7 src/Transposition.hpp File Reference

```
#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::Griding

  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

str	String to be cript or decript

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
00040                 result.array[result.get_size()] =
      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
00096     public:
00097
00102         Griding(
00103             const ProcessIndicator& pi_1,
00104             const ProcessIndicator& pi_2
00105         ){
00106
00107             process_indicators[0] = std::move(pi_1);
00108             process_indicators[1] = std::move(pi_2);
00109         }
00110
00115         String
00116         crypto(
00117             const String& message_to_be_crypto
00118         ){
00119
00120             if( !Method::if_size_is_valid(message_to_be_crypto.get_size()) ){ throw
      std::invalid_argument("Mensagem não tem tamanho válido"); }
00121
00122             // ------------------
00123             // Iniciamos aplicação do algoritmo
00124             String temp;
00125
00126
00127             return temp;
00128         }
00129
00130
00131
00132
00136         static
00137         void
00138         debug(){
00139
00140             Griding exemplo(
00141                 {'A', 1, 2},
00142                 {'B', 2, 1}
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