# JPCRE2

10.25.01

Generated by Doxygen 1.8.11

# Contents

# 1 JPCRE2

C++ wrapper of PCRE2 library

PCRE2 is the name used for a revised API for the PCRE library, which is a set of functions, written in C, that implement regular expression pattern matching using the same syntax and semantics as Perl, with just a few differences. Some features that appeared in Python and the original PCRE before they appeared in Perl are also available using the Python syntax.

This provides some C++ wrapper functions to provide some useful utilities like regex match and regex replace.

## 1.1 Dependency

1. PCRE2 library (`version >=10.21`).

If the required PCRE2 version is not available in the official channel, download `my fork of the library`.

## 1.2 Install or Include

The `jpcre2.hpp` header should be included in the source file that uses JPCRE2 functionalities.

### 1.2.1 Use with sources

After including the header you can compile your source either by installing and linking with JPCRE2 library or providing the following sources to your compiler:

1. **jpcre2.hpp**
2. **jpcre2.cpp**

An example compile/build command with GCC would be:

```
1 g++ mycpp.cpp jpcre2.cpp jpcre2.hpp -lpcre2-8
```

If your PCRE2 library is not in the standard library path, then add the path:

```
1 g++ mycpp.cpp ... -L/path/to/your/pcre2/library -lpcre2-8
```

**Note that** it requires the PCRE2 library installed in your system. If it is not already installed and linked in your compiler, you will need to link it with appropriate path and options.

#### 1.2.2 Use as a library

To install it as a library in a Unix based system, run:

```
1 ./configure
2 make
3 make install # or sudo make install
```

Now `#include <jpcre2.hpp>` in your code and build/compile by linking with both JPCRE2 and PCRE2 library.

An example command for GCC would be:

```
1 g++  mycpp.cpp -ljpcre2-8 -lpcre2-8 #sequence is important
```

If you are in a non-Unix system (e.g Windows), build a library from the JPCRE2 sources with your favorite IDE or use it as it is.

**Notes:**

1. `PCRE2_CODE_UNIT_WIDTH` other than 8 is not supported in this version.
2. To use the `PCRE2 POSIX` compatible library, add the `-lpcre2-posix` along with the others.

### 1.3 How to code with JPCRE2

Performing a match or replacement against regex pattern involves two steps:

1. Compiling the pattern
2. Performing the match or replacement operation

#### 1.3.1 Compile a pattern

**First create a `jpcre2::Regex` object**

(You can use temporary object too, see `short examples`).

This object will hold the pattern, modifiers, compiled pattern, error and warning codes.

```
jpcre2::Regex re;   //Create object, it's not supposed to throw exception
```

Each object for each regex pattern.

**Compile the pattern** and catch any error exception:

```
try{re.setPattern("(?:(?<word>[?.#@:]+)|(?<word>\\w+))\\s*(?<digit>\\d+)")  //set pattern
      .setModifier("nJ")                                                    //set modifier
      .addJpcre2Option(jpcre2::VALIDATE_MODIFIER
               //modifier goes through validation check
                       | jpcre2::JIT_COMPILE                                //perform
       JIT compile
                       | jpcre2::ERROR_ALL)                                 //treat
       warnings as errors
      .addPcre2Option(0)                                                    //add pcre2
       option
      .compile();                                                           //Finally compile it.

    //Another way is to use constructor to initialize and compile at the same time:
    jpcre2::Regex re2("pattern2","mSi");  //S is an optimization mod.
    jpcre2::Regex re3("pattern3", PCRE2_ANCHORED);
    jpcre2::Regex re4("pattern4", PCRE2_ANCHORED,
      jpcre2::JIT_COMPILE);
}
catch(int e){
    /*Handle error*/
    std::cerr<<re.getErrorMessage(e)<<std::endl;
}
```

Now you can perform match or replace against the pattern. Use the `match()` member function to preform regex match and the `replace()` member function to perform regex replace.

### 1.3.2 Match

The `jpcre2::Regex::match(const String& s)` member function can take two arguments (subject & modifier) and returns the number of matches found against the compiled pattern.

To get the match result (captured groups) however, you need to call the `jpcre2::RegexMatch::match()` function. Point be noted that, you can not call this function directly or create any object of the class `jpcre2::RegexMatch`. To call this function, first invoke the `jpcre2::Regex::initMatch()` function. It will give you a temporary `jpcre2::RegexMatch` object. Now you can chain function calls of `jpcre2::RegexMatch::setNumberedSubstringVector(VecNum* vec_num)` and such functions from `jpcre2::RegexMatch` class to pass various parameters. After you are done passing all the parameter that you need, the `jpcre2::RegexMatch::match()` function should be called to perform the actual match and return the match count. The match results will be stored in vectors (vectors of maps) whose pointers were passed as parameters.

*You should catch any error exception that may be thrown in case error occurs.*

#### 1.3.2.1 Get match count

```
//If you want to match all and get the match count, use the action modifier 'g':
size_t count = jpcre2::Regex("(\\d)|(\\w)","m").match("I am the subject","g");
```

#### 1.3.2.2 Get match result

To get the match results, you need to pass appropriate vector pointers. This is an example of how you can get the numbered substrings/captured groups from a match:

```
jpcre2::VecNum vec_num;
try{
    size_t count=re.initMatch()                                 //prepare for match() call
                   .setSubject(subject)                         //set subject string
                   .setModifier(ac_mod)                         //set modifier string
                   .setNumberedSubstringVector(&vec_num)        //pass VecNum
       vector to store maps of numbered substrings
                   .addJpcre2Option(jpcre2::VALIDATE_MODIFIER)  //
      add jpcre2 option
                   .match();                                    //Finally perform the match.
    //vec_num will be populated with maps of numbered substrings.
    //count is the total number of matches found
}
catch(int e){
    /*Handle error*/
    std::cerr<<re.getErrorMessage(e)<<std::endl;
}
```

#### 1.3.2.3 Access a substring

You can access a substring/captured group by specifying their index (position):

```
std::cout<<vec_num[0][0]; // group 0 in first match
std::cout<<vec_num[0][1]; // group 1 in first match
std::cout<<vec_num[1][0]; // group 0 in second match
```

#### 1.3.2.4 Get named substrings

To get named substring and/or name to number mapping, pass pointer to the appropriate vectors with `jpcre2::RegexMatch::setNamedSubstringVector()` and/or `jpcre2::RegexMatch::setNameToNumberMapVector()` before doing the match.

```
jpcre2::VecNum vec_num;    ///Vector to store numbured substring Map.
jpcre2::VecNas vec_nas;    ///Vector to store named substring Map.
jpcre2::VecNtN vec_ntn;    ///Vector to store Named substring to Number Map.
std::string ac_mod="g";    // g is for global match. Equivalent to using setFindAll() or FIND_ALL in
       addJpcre2Options()
try{
    re.initMatch()
      .setSubject(subject)                          //set subject string
      .setModifier(ac_mod)                          //set modifier string
      .setNumberedSubstringVector(&vec_num)         //pass pointer to vector of
       numbered substring maps
      .setNamedSubstringVector(&vec_nas)            //pass pointer to vector of named
       substring maps
      .setNameToNumberMapVector(&vec_ntn)           //pass pointer to vector of name
       to number maps
      .addJpcre2Option(jpcre2::VALIDATE_MODIFIER)   //add jpcre2
       option
      .addPcre2Option(PCRE2_ANCHORED)               //add pcre2 option
      .match();                                     //Finally perform the match()
}
catch(int e){
    /*Handle error*/
    std::cerr<<re.getErrorMessage(e)<<std::endl;
}
```

#### 1.3.2.5 Accesing a substring by name

```
std::cout<<vec_nas[0]["name"]; // captured group by name in first match
std::cout<<vec_nas[1]["name"]; // captured group by name in second match
```

#### 1.3.2.6 Get the position of a capture group name

If you need this information, you should have passed a `jpcre2::VecNtN` pointer to `jpcre2::RegexMatch::setNameToNumberMapVector()` function before doing the match (see above).

```
std::cout<<vec_ntn[0]["name"]; // position of captured group 'name' in first match
```

#### 1.3.2.7 Iterate through match result

You can iterate through the matches and their substrings like this:

```
for(size_t i=0;i<vec_num.size();++i){
    //i=0 is the first match found, i=1 is the second and so forth
    for(jpcre2::MapNum::iterator ent=vec_num[i].begin();ent!=vec_num[i].end();++ent){
        //ent.first is the number/position of substring found
        //ent.second is the substring itself
        //when ent->first is 0, ent->second is the total match.
        std::cout<<"\n\t"<<ent->first<<": "<<ent->second<<"\n";
    }
}
```

If you are using >=C++11, you can make the loop a lot simpler:

```
for(size_t i=0;i<vec_num.size();++i){
    for(auto const& ent : vec_num[i]){
        std::cout<<"\n\t"<<ent.first<<": "<<ent.second<<"\n";
    }
}
```

*The process of iterating through the vectors and associated maps are the same for all three. The size of those vectors are the same and can be accessed in the same way.*

#### 1.3.3 Replace or Substitute

The `jpcre2::Regex::replace(const String& s, const String& r)` member function can take up-to three arguments (subject, replacement string, modifier) and returns the resultant replaced string.

If you want to pass more options or prefer a named parameter idiom, you will have to use the `jpcre2::RegexReplace::replace()` function instead. Point be noted that, all constructors of the `jpcre2::RegexReplace` class are private and thus you can't create any object of this class or call the mentioned function directly. In this case you need to call `jpcre2::Regex::initReplace()` function which will give you a temporary object that you can use to chain method calls to pass various options to be used by `jpcre2::Regex::Replace::replace()` before calling it.

*You should catch any error exception that may be thrown in case error occurs.*

##### 1.3.3.1 Simple replacement

```
//Using a temporary regex object
std::cout<<jpcre2::Regex("\\d+").replace("I am digits 1234","5678", "g");
//'g' modifier is for global replacement
```

##### 1.3.3.2 Using named parameter idiom

```
try{
    std::cout<<
    re.initReplace()       //Prepare to call jpcre2::RegexReplace::replace()
      .setSubject(s)       //Set various parameters
      .setReplaceWith(s2)  //...
      .setModifier("gE")   //...
      .addJpcre2Option(0)  //...
      .addPcre2Option(0)   //...
      .replace();          //Finally do the replacement.
    //gE is the modifier passed (global and unknown-unset-empty).
    //Access substrings/captured groups with ${1234},$1234 (for numbered substrings)
    // or ${name} (for named substrings) in the replacement part i.e in setReplaceWith()
}
catch(int e){
    /*Handle error*/
    std::cerr<<re.getErrorMessage(e)<<std::endl;
}
```

If you pass the size of the resultant string with `jpcre2::RegexReplace::setBufferSize()` function, make sure it will be enough to store the whole resultant replaced string; otherwise the internal replace function (`pcre2_substitute()`) will be called *twice* to adjust the size of the buffer to hold the whole resultant string in order to avoid `PCRE2_ERROR_NOMEMORY` error.

### 1.4 Modifiers

**JPCRE2** uses modifiers to control various options, type, behavior of the regex and its' interactions with different functions that uses it. Two types of modifiers are available: *compile modifiers* and *action modifiers*:

#### 1.4.1 Compile modifiers

These modifiers define the behavior of a regex pattern. They have more or less the same meaning as the `PHP regex modifiers` except for `e, j and n` (marked with *).

| Modifier | Details |
|---|---|
| e* | Unset back-references in the pattern will match to empty strings. Equivalent to PCRE2_MATCH_↩ UNSET_BACKREF. |
| i | Case-insensitive. Equivalent to PCRE2_CASELESS option. |
| j* | \u \U \x and unset back-references will act as JavaScript standard.<br>• matches an upper case "U" character (by default it causes a compile time error if this option is not set).<br>• matches a lower case "u" character unless it is followed by four hexadecimal digits, in which case the hexadecimal number defines the code point to match (by default it causes a compile time error if this option is not set).<br>• matches a lower case "x" character unless it is followed by two hexadecimal digits, in which case the hexadecimal number defines the code point to match (By default, as in Perl, a hexadecimal number is always expected after , but it may have zero, one, or two digits (so, for example,  matches a binary zero character followed by z) ).<br>• Unset back-references in the pattern will match to empty strings. |
| m | Multi-line regex. Equivalent to PCRE2_MULTILINE option. |
| n* | Enable Unicode support for \w \d etc... in pattern. Equivalent to PCRE2_UTF \| PCRE2_UCP. |
| s | If this modifier is set, a dot meta-character in the pattern matches all characters, including newlines. Equivalent to PCRE2_DOTALL option. |
| u | Enable UTF support.Treat pattern and subjects as UTF strings. It is equivalent to PCRE2_UTF option. |
| x | Whitespace data characters in the pattern are totally ignored except when escaped or inside a character class, enables commentary in pattern. Equivalent to PCRE2_EXTENDED option. |
| A | Match only at the first position. It is equivalent to PCRE2_ANCHORED option. |
| D | A dollar meta-character in the pattern matches only at the end of the subject string. Without this modifier, a dollar also matches immediately before the final character if it is a newline (but not before any other newlines). This modifier is ignored if m modifier is set. Equivalent to PCRE2_DOLLAR↩ _ENDONLY option. |
| J | Allow duplicate names for sub-patterns. Equivalent to PCRE2_DUPNAMES option. |
| S | When a pattern is going to be used several times, it is worth spending more time analyzing it in order to speed up the time taken for matching/replacing. It may also be beneficial for a very long subject string or pattern. Equivalent to an extra compilation with JIT_COMPILER with the option PCRE2_JIT_COMPLETE. |
| U | This modifier inverts the "greediness" of the quantifiers so that they are not greedy by default, but become greedy if followed by ?. Equivalent to PCRE2_UNGREEDY option. |

#### 1.4.2 Action modifiers

These modifiers are not compiled in the regex itself, rather they are used per call of each match or replace function.

| Modifier | Details |
|---|---|
| A | Match at start. Equivalent to PCRE2_ANCHORED. Can be used in match operation. Setting this option only at match time (i.e regex was not compiled with this option) will disable optimization during match time. |
| e | Replaces unset group with empty string. Equivalent to PCRE2_SUBSTITUTE_UNSET_EMPTY. Can be used in replace operation. |
| E | Extension of e modifier. Sets even unknown groups to empty string. Equivalent to PCRE2_SUBS↩ TITUTE_UNSET_EMPTY \| PCRE2_SUBSTITUTE_UNKNOWN_UNSET. |
| g | Global. Will perform global matching or replacement if passed. |

| Modifier | Details |
|---|---|
| x | Extended replacement operation. It enables some Bash like features:<br>${<n>:-<string>}<br>${<n>:+<string1>:<string2>}<br><n> may be a group number or a name. The first form specifies a default value. If group <n> is set, its value is inserted; if not, <string> is expanded and the result is inserted. The second form specifies strings that are expanded and inserted when group <n> is set or unset, respectively. The first form is just a convenient shorthand for ${<n>:+${<n>}:<string>}. |

## 1.5 Options

JPCRE2 allows both PCRE2 and native JPCRE2 options to be passed. PCRE2 options are recognized by the PCPRE2 library itself.

### 1.5.1 JPCRE2 options

These options are meaningful only for the **JPCRE2** library itself not the original **PCRE2** library. We use the addJpcre2Options() function to pass these options.

| Option | Details |
|---|---|
| jpcre2::NONE | This is the default option. Equivalent to 0 (zero). |
| jpcre2::VALIDATE_MODIFIER | If this option is passed, modifiers will be subject to validation check. If any of them is invalid, a jpcre2::ERROR::INVALID_MODIFIER error exception will be thrown. You can get the error message with jpcre2::Regex::getErrorMessage(error_code) member function. |
| jpcre2::FIND_ALL | This option will do a global matching if passed during matching. The same can be achieved by passing the 'g' modifier with jpcre2::RegexMatch::setModifier() function. |
| jpcre2::ERROR_ALL | Treat warnings as errors and throw exception. |
| jpcre2::JIT_COMPILE | This is same as passing the S modifier during pattern compilation. |

### 1.5.2 PCRE2 options

While having its own way of doing things, JPCRE2 also supports the traditional PCRE2 options to be passed. We use the addPcre2Option() functions to pass the PCRE2 options. These options are the same as the PCRE2 library and have the same meaning. For example instead of passing the 'g' modifier to the replacement operation we can also pass its PCRE2 equivalent PCRE2_SUBSTITUTE_GLOBAL to have the same effect.

## 1.6 Short examples

```
size_t count;
///Check if string matches the pattern
/**
 * The following uses a temporary Regex object.
 * */
if(jpcre2::Regex("(\\d)|(\\w)").match("I am the subject"))
    std::cout<<"\nmatched";
else
    std::cout<<"\nno match";
/**
 * The above is a good example of using temporary objects to perform match (or replace)
 *
```

```
 * Using the modifier S (i.e jpcre2::JIT_COMPILE) with temporary object may or may not give you
 * any performance boost (depends on the complexity of the pattern). The more complex
 * the pattern gets, the more sense the S modifier makes.
 * */

///If you want to match all and get the match count, use the action modifier 'g':
std::cout<<"\n"<<
    jpcre2::Regex("(\\d)|(\\w)","m").match("I am the subject","g");

/**
 * Modifiers passed to the Regex constructor or with compile() function are compile modifiers
 * Modifiers passed with the match() or replace() functions are action modifiers
 * */

/// Substrings/Captured groups:

/**
 * *** Getting captured groups/substring ***
 *
 * captured groups or substrings are stored in maps for each match,
 * and each match is stored in a vector.
 * Thus captured groups are in a vector of maps.
 *
 * PCRE2 provides two types of substrings:
 *  1. numbered (index) substring
 *  2. named substring
 *
 * For the above two, we have two vectors respectively:
 *  1. jpcre2::VecNum (Corresponding map: jpcre2::MapNum)
 *  2. jpcre2::VecNas (Corresponding map: jpcre2::MapNas)
 *
 * Another additional vector is available to get the substring position/number
 * for a particular captured group by name. It's a vector of name to number maps
 *  * jpcre2::VecNtN (Corresponding map: jpcre2:MapNtN)
 * */

/// ***** Get numbered substring ***** ///
jpcre2::VecNum vec_num;
count =
jpcre2::Regex("(\\w+)\\s*(\\d+)","m")
        .initMatch()
        .setSubject("I am 23, I am digits 10")
        .setModifier("g")
        .setNumberedSubstringVector(&vec_num)
        .match();
/**
* count (the return value) is guaranteed to give you the correct number of matches,
* while vec_num.size() may give you wrong result if any match result
* was failed to be inserted in the vector. This should not happen
* i.e count and vec_num.size() should always be equal.
* */
std::cout<<"\nNumber of matches: "<<count/* or vec_num.size()*/;

///Now vec_num is populated with numbered substrings for each match
///The size of vec_num is the total match count
///vec_num[0] is the first match
///The type of vec_num[0] is jpcre2::MapNum
std::cout<<"\nTotal match of first match: "<<vec_num[0][0];      ///Total match (group 0) from first match
std::cout<<"\nCaptrued group 1 of frist match: "<<vec_num[0][1]; ///captured group 1 from first match
std::cout<<"\nCaptrued group 2 of frist match: "<<vec_num[0][2]; ///captured group 2 from first match
std::cout<<"\nCaptrued group 3 of frist match: "<<vec_num[0][3]; ///captured group 3 doesn't exist, it will
       give you empty string
///Using the [] operator with jpcre2::MapNum will create new element if it doesn't exist
/// i.e vec_num[0][3] were created in the above example.
///This should be ok, if existence of a particular substring is not important

///If the existence of a substring is important, use the std::map::find() or std::map::at() (>=C++11)
       function to access map elements
/* //>=C++11
try{
    ///This will throw exception, because substring 4 doesn't exist
    std::cout<<"\nCaptrued group 4 of frist match: "<<vec_num[0].at(4);
} catch (std::logic_error e){
    std::cout<<"\nCaptrued group 4 doesn't exist";
}*/

///There were two matches found (vec_num.size() == 2) in the above example
std::cout<<"\nTotal match of second match: "<<vec_num[1][0];      ///Total match (group 0) from second
       match
std::cout<<"\nCaptrued group 1 of second match: "<<vec_num[1][1]; ///captured group 1 from second match
std::cout<<"\nCaptrued group 2 of second match: "<<vec_num[1][2]; ///captured group 2 from second match


/// ***** Get named substring ***** ///

jpcre2::VecNas vec_nas;
jpcre2::VecNtN vec_ntn; /// We will get name to number map vector too
```

```
count =
jpcre2::Regex("(?<word>\\w+)\\s*(?<digit>\\d+)","m")
        .initMatch()
        .setSubject("I am 23, I am digits 10")
        .setModifier("g")
        ///.setNumberedSubstringVector(vec_num) /// We don't need it in this example
        .setNamedSubstringVector(&vec_nas)
        .setNameToNumberMapVector(&vec_ntn) /// Additional (name to number maps)
        .match();
std::cout<<"\nNumber of matches: "<<vec_nas.size()/* or count */;
///Now vec_nas is populated with named substrings for each match
///The size of vec_nas is the total match count
///vec_nas[0] is the first match
///The type of vec_nas[0] is jpcre2::MapNas
std::cout<<"\nCaptured group (word) of first match: "<<vec_nas[0]["word"];
std::cout<<"\nCaptured group (digit) of first match: "<<vec_nas[0]["digit"];

///If the existence of a substring is important, use the std::map::find() or std::map::at() (>=C++11)
       function to access map elements
/* //>=C++11
try{
    ///This will throw exception becasue the substring name 'name' doesn't exist
    std::cout<<"\nCaptured group (name) of first match: "<<vec_nas[0].at("name");
} catch(std::logic_error e){
    std::cout<<"\nCaptured group (name) doesn't exist";
}*/

///There were two matches found (vec_nas.size() == 2) in the above example
std::cout<<"\nCaptured group (word) of second match: "<<vec_nas[1]["word"];
std::cout<<"\nCaptured group (digit) of second match: "<<vec_nas[1]["digit"];

///Get the position (number) of a captured group name (that was found in match)
std::cout<<"\nPosition of captured group (word) in first match: "<<vec_ntn[0]["word"];
std::cout<<"\nPosition of captured group (digit) in first match: "<<vec_ntn[0]["digit"];

/**
 * Replacement Examples
 * Replace pattern in a string with a replacement string
 *
 * The initReplace() function can take a subject and replacement string as argument.
 * You can also pass the subject with setSubject() function in method chain,
 * replacement string with setReplaceWith() function in method chain, etc ...
 *
 * A call to replace() will return the resultant string
 * */

std::cout<<"\n"<<
///replace first occurrence of a digit with @
jpcre2::Regex("\\d").replace("I am the subject string 44", "@");

std::cout<<"\n"<<
///replace all occrrences of a digit with @
jpcre2::Regex("\\d").replace("I am the subject string 44", "@", "g");

///swap two parts of a string
std::cout<<"\n"<<
jpcre2::Regex("^([^\t]+)\t([^\t]+)$")
        .replace("I am the subject\tTo be swapped according to tab", "$2 $1");
```

# 2 Namespace Documentation

## 2.1 jpcre2 Namespace Reference

Top level namespace of JPCRE2.

### Namespaces

- ERROR

  *Namespace for error codes.*

- utils

  *Namespace for some utility functions.*

### Classes

- class Regex

  *Implements public overloaded and copy constructors, provides functions to set/unset various options and perform regex match and replace against a compiled pattern.*
- class RegexMatch

  *Performs regex matching.*
- class RegexReplace

  *Performs regex replace on a string.*

### Typedefs

- typedef std::size_t SIZE_T

  *Used for match count and vector size.*
- typedef uint32_t Uint

  *Used for options (bitwise operation)*
- typedef std::string String

  *Used as std::string.*
- typedef std::map< String, String > MapNas

  *Map for Named substrings.*
- typedef std::map< SIZE_T, String > MapNum

  *Map for Numbered substrings.*
- typedef std::map< String, SIZE_T > MapNtN

  *Substring name to Substring number map.*
- typedef MapNtN MapNtn

  *Allow spelling mistake of MapNtN as MapNtn.*
- typedef std::vector< MapNas > VecNas

  *Vector of matches with named substrings.*
- typedef std::vector< MapNtN > VecNtN

  *Vector of substring name to Substring number map.*
- typedef VecNtN VecNtn

  *Allow spelling mistake of VecNtN as VecNtn.*
- typedef std::vector< MapNum > VecNum

  *Vector of matches with numbered substrings.*

### Enumerations

### Variables

- const SIZE_T SUBSTITUTE_RESULT_INIT_SIZE = std::numeric_limits<int>::max()

  *Used by default to provide big enough initial buffer for replaced string.*
- const String LOCALE_NONE = "JPCRE2_NONE"

  *Don't do anything about locale if it is set to LOCALE_NONE.*
- const String LOCALE_DEFAULT = LOCALE_NONE

  *Default locale.*
- const String JIT_ERROR_MESSAGE_PREFIX = "JIT compilation failed! "

  *Prefix to be added to JIT error message.*

#### 2.1.1 Detailed Description

Top level namespace of JPCRE2.

All functions, classes, constants, enums that are provided by JPCRE2 belong to this namespace while **PCRE2** functions, constants remain outside of its scope.

If you want to use any PCRE2 functions or constants, remember that they are in the global scope and should be used as such.

#### 2.1.2 Enumeration Type Documentation

##### 2.1.2.1 anonymous enum

These constants provide JPCRE2 options.

**Enumerator**

***NONE*** Option 0 (zero)

***VALIDATE_MODIFIER*** Perform validation check on modifiers and throw #INVALID_MODIFIER if any wrong modifier is passed.

***FIND_ALL*** Find all during match (global match)

***JIT_COMPILE*** Perform JIT compilation for optimization.

***ERROR_ALL*** Treat warnings as error and throw exception (warnings don't throw exception)

#### 2.1.3 Variable Documentation

##### 2.1.3.1 const jpcre2::String jpcre2::LOCALE_DEFAULT = LOCALE_NONE

Default locale.

Default local to be used.

Referenced by jpcre2::Regex::init_vars().

##### 2.1.3.2 const jpcre2::String jpcre2::LOCALE_NONE = "JPCRE2_NONE"

Don't do anything about locale if it is set to LOCALE_NONE.

Nothing to be done on locale.

Referenced by jpcre2::Regex::compile().

##### 2.1.3.3 const jpcre2::SIZE_T jpcre2::SUBSTITUTE_RESULT_INIT_SIZE = std::numeric_limits<int>::max()

Used by default to provide big enough initial buffer for replaced string.

Use max of int as the initial size of replaced string.

**Author**

Md Jahidul Hamid

Referenced by jpcre2::RegexReplace::init_vars().

## 2.2 jpcre2::ERROR Namespace Reference

Namespace for error codes.

### Enumerations

#### 2.2.1 Detailed Description

Namespace for error codes.

#### 2.2.2 Enumeration Type Documentation

##### 2.2.2.1 anonymous enum

ERROR codes that are thrown in case error occurs.

JPCRE2 error codes are positive integers while PCRE2 error codes are negative integers.

**Enumerator**

***INVALID_MODIFIER*** Error to be thrown when invalid modifier detected.

***JIT_COMPILE_FAILED*** Error to be thrown when JIT compile fails.

## 2.3 jpcre2::utils Namespace Reference

Namespace for some utility functions.

### Functions

- String toString (int a)

  *Converts an integer to String.*
- String toString (char a)

  *Converts a char to String.*
- String toString (const char ∗a)

  *Converts const char∗ to String.*
- String toString (PCRE2_UCHAR ∗a)

  *Converts a PCRE2_UCHAR∗ to String.*
- String getPcre2ErrorMessage (int err_num)

  *Get PCRE2 error message for an error number.*

#### 2.3.1 Detailed Description

Namespace for some utility functions.

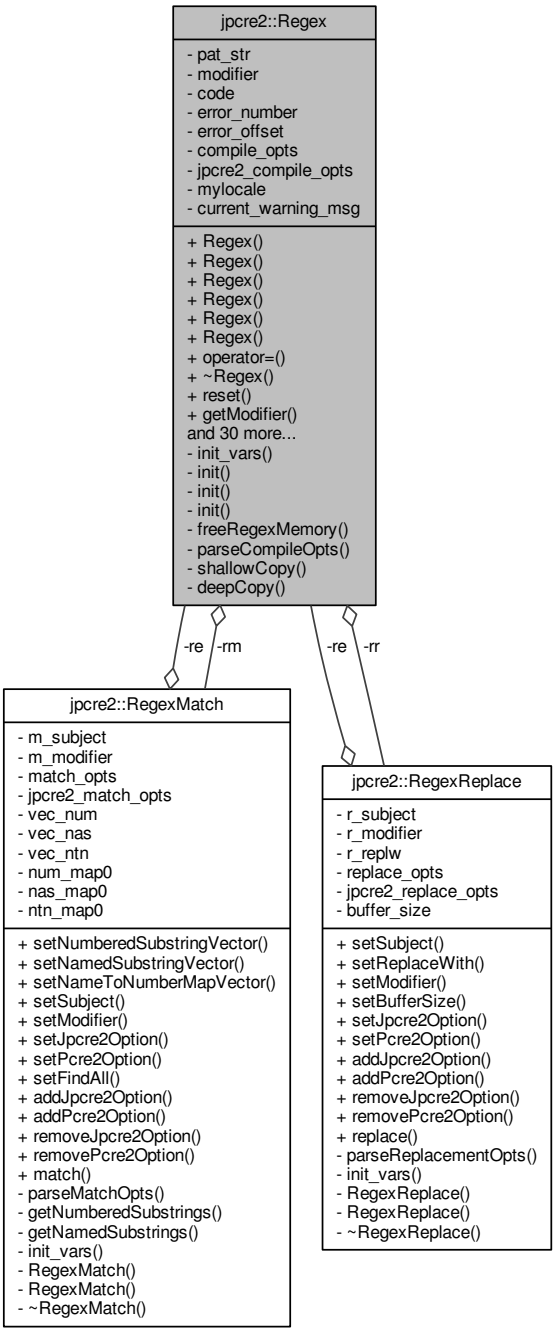
# 3 Class Documentation

## 3.1 jpcre2::Regex Class Reference

Implements public overloaded and copy constructors, provides functions to set/unset various options and perform regex match and replace against a compiled pattern.

`#include <jpcre2.hpp>`

Collaboration diagram for jpcre2::Regex:



### Public Member Functions

- Regex ()

  *Default Constructor.*
- Regex (const String &re)

  *Compiles pattern with initialization.*
- Regex (const String &re, const String &mod)

  *This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.*
- Regex (const String &re, Uint pcre2_opts)

  *This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.*
- Regex (const String &re, Uint pcre2_opts, Uint opt_bits)

  *This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.*
- Regex (const Regex &r)

  *This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Copy constructor.*
- Regex & operator= (const Regex &r)

  *Overloaded assignment operator.*
- ~Regex ()

  *Destructor Deletes memory used by rm an rr.*
- Regex & reset ()

  *Reset all class variables to its default (initial) state.*
- String getModifier ()

  *Get modifier string.*
- String getPattern ()

  *Get pattern string.*
- String getLocale ()

  *Get locale as a string.*
- Uint getPcre2Option ()

  *Get PCRE2 option.*
- Uint getJpcre2Option ()

  *Get JPCRE2 option.*
- String getErrorMessage (int err_num, PCRE2_SIZE err_off)

  *Get error message by error number and error offset.*
- String getErrorMessage (int err_num)

  *This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use class variable error_offest as error offset.*
- String getErrorMessage ()

  *This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use class variable error_number as error number and error_offest as error offset.*
- String getWarningMessage ()

  *Get current warning message.*
- int getErrorNumber ()

  *Get error number return error_number.*
- PCRE2_SIZE getErrorOffset ()

  *Get error offset return error_offset.*
- Regex & setPattern (const String &re)

  *Set the Pattern string pat_str.*
- Regex & setModifier (const String &x)

  *Set the modifier modifier (overwrite existing JPCRE2 and PCRE2 option).*

- Regex & setLocale (const String &x)

  *Set the locale mylocale.*
- Regex & setJpcre2Option (Uint x)

  *Set JPCRE2 option jpcre2_compile_opts (overwrites existing option)*
- Regex & setPcre2Option (Uint x)

  *Set PCRE2 option compile_opts (overwrites existing option)*
- Regex & addJpcre2Option (Uint x)

  *Add option to existing JPCRE2 options jpcre2_compile_opts.*
- Regex & addPcre2Option (Uint x)

  *Add option to existing PCRE2 options compile_opts.*
- Regex & removeJpcre2Option (Uint x)

  *Remove option from existing JPCRE2 option jpcre2_compile_opts.*
- Regex & removePcre2Option (Uint x)

  *Remove option from existing PCRE2 option compile_opts.*
- void compile (void)

  *Compile the regex pattern from class variable pat_str.*
- void compile (const String &re, Uint po, Uint jo)

  *This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.*
- void compile (const String &re, Uint po)

  *This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.*
- void compile (const String &re, const String &mod)

  *This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.*
- void compile (const String &re)

  *This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.*
- SIZE_T match (const String &s, const String &mod)

  *Perform regex match.*
- SIZE_T match (const String &s)

  *This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.*
- RegexMatch & initMatch ()

  *Prepare to call RegexMatch::match().*
- String replace (const String &mains, const String &repl, const String &mod)

  *Perform regex replace.*
- String replace (const String &mains, const String &repl)

  *This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.*
- RegexReplace & initReplace ()

  *Prepare to call RegexReplace::replace().*

#### Private Member Functions

- void init_vars ()

  *Initialize class variables.*
- void init ()

  *Call Regex::init_vars() and initialize class variables.*
- void init (const String &re, const String &mod)

*This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.*

- void init (const String &re, Uint po, Uint jo)

  *This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.*

- void freeRegexMemory (void)

  *Free code if it's non-NULL.*

- void parseCompileOpts (void)

  *Parse modifier and set equivalent PCRE2 and JPCRE2 options.*

- void shallowCopy (const Regex &r)

  *Do a shallow copy of class variables.*

- void deepCopy (const Regex &r)

  *Do a deep copy of rm, rr and code.*

#### Private Attributes

- RegexMatch * rm

  *Pointer to RegexMatch object.*

- RegexReplace * rr

  *Pointer to RegexReplace object.*

- String pat_str

  *Pattern string.*

- String modifier

  *Modifier string.*

- pcre2_code * code

  *Pointer to compiled pattern.*

- int error_number

  *Error number.*

- PCRE2_SIZE error_offset

  *Error offset.*

- Uint compile_opts

  *Compile options for PCRE2 (used by PCRE2 internal function pcre2_compile())*

- Uint jpcre2_compile_opts

  *Compile options specific to JPCRE2.*

- String mylocale

  *Locale as a string.*

- String current_warning_msg

  *current warning message*

#### Friends

- class RegexMatch

  *Define RegexMatch as friends. It needs to access the compiled pattern which is a private property of this class.*

- class RegexReplace

  *Define RegexReplace as friends. It needs to access the compiled pattern which is a private property of this class.*

#### 3.1.1 Detailed Description

Implements public overloaded and copy constructors, provides functions to set/unset various options and perform regex match and replace against a compiled pattern.

Each regex pattern needs an object of this class.

A pattern must be compiled either by explicitly calling the compile function or using one of the parameterized constructors.

#### 3.1.2 Constructor & Destructor Documentation

##### 3.1.2.1 jpcre2::Regex::Regex ( ) [inline]

Default Constructor.

Initializes all class variables to defaults. Does not perform any compilation.

##### 3.1.2.2 jpcre2::Regex::Regex ( const String & *re* ) [inline]

Compiles pattern with initialization.

**Parameters**

| | |
|---|---|
| *re* | Pattern string |

##### 3.1.2.3 jpcre2::Regex::Regex ( const String & *re,* const String & *mod* ) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Compiles pattern.

**Parameters**

| | |
|---|---|
| *re* | Pattern string |
| *mod* | Modifier string |

##### 3.1.2.4 jpcre2::Regex::Regex ( const String & *re,* Uint *pcre2_opts* ) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Compiles pattern.

**Parameters**

| | |
|---|---|
| *re* | Pattern string |
| *pcre2_opts* | PCRE2 option value |

#### 3.1.2.5 jpcre2::Regex::Regex ( const String & *re*, Uint *pcre2_opts*, Uint *opt_bits* ) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Compiles pattern.

**Parameters**

| | |
|---|---|
| *re* | Pattern string |
| *pcre2_opts* | PCRE2 option value |
| *opt_bits* | JPCRE2 option value |

#### 3.1.2.6 jpcre2::Regex::Regex ( const Regex & *r* ) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Copy constructor.

Compiles pattern and Performs a deep copy.

**Parameters**

| | |
|---|---|
| *r* | const Regex& |

### 3.1.3 Member Function Documentation

#### 3.1.3.1 Regex& jpcre2::Regex::addJpcre2Option ( Uint *x* ) [inline]

Add option to existing JPCRE2 options jpcre2_compile_opts.

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

∗this

#### 3.1.3.2 Regex& jpcre2::Regex::addPcre2Option ( Uint *x* ) [inline]

Add option to existing PCRE2 options compile_opts.

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

*this

#### 3.1.3.3 void jpcre2::Regex::compile ( void )

Compile the regex pattern from class variable pat_str.

Use options from class variables.

Prefer using one of its variants when compiling pattern for an already declared Regex object. An use of

```
re = Regex("pattern");
```

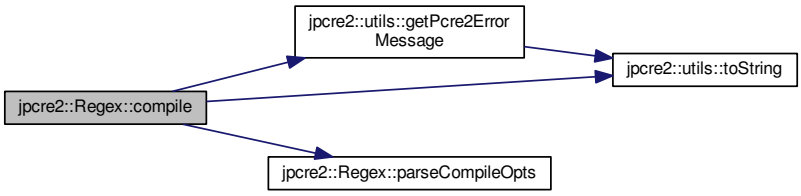
(or such) is discouraged. see `Regex::operator=(const Regex& r)` for details.

**See also**

void compile(const String& re, Uint po, Uint jo)
void compile(const String& re, Uint po)
void compile(const String& re, const String& mod)
void compile(const String& re)

References code, compile_opts, current_warning_msg, jpcre2::ERROR_ALL, error_number, error_offset, jpcre2::utils::getPcre2ErrorMessage(), jpcre2::JIT_COMPILE, jpcre2::ERROR::JIT_COMPILE_FAILED, jpcre2_↩ compile_opts, jpcre2::LOCALE_NONE, mylocale, parseCompileOpts(), pat_str, and jpcre2::utils::toString().

Here is the call graph for this function:




#### 3.1.3.4 void jpcre2::Regex::compile ( const String & *re,* Uint *po,* Uint *jo* ) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Set the specified parameters, then compile the pattern using information from class variables.

**Parameters**

| | |
|---|---|
| *re* | Pattern string |
| *po* | PCRE2 option |
| *jo* | JPCRE2 option |

#### 3.1.3.5 void jpcre2::Regex::compile ( const String & *re,* Uint *po* ) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Set the specified parameters, then compile the pattern using options from class variables.

**Parameters**

| | |
|---|---|
| *re* | Pattern string |
| *po* | PCRE2 option |

#### 3.1.3.6 void jpcre2::Regex::compile ( const String & *re,* const String & *mod* ) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Set the specified parameters, then compile the pattern using options from class variables.

**Parameters**

| | |
|---|---|
| *re* | Pattern string |
| *mod* | Modifier string |

#### 3.1.3.7 void jpcre2::Regex::compile ( const String & *re* ) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Set the specified parameters, then compile the pattern using options from class variables.

**Parameters**

| | |
|---|---|
| *re* | Pattern string |

#### 3.1.3.8 jpcre2::String jpcre2::Regex::getErrorMessage ( int *err_num,* PCRE2_SIZE *err_off* )

Get error message by error number and error offset.

**Parameters**

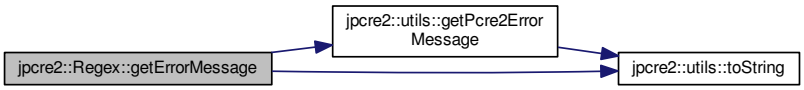
| | |
|---|---|
| *err_num* | Error number |
| *err_off* | Error offset |

**Returns**

Error message as a string

References jpcre2::utils::getPcre2ErrorMessage(), jpcre2::ERROR::INVALID_MODIFIER, jpcre2::ERROR::JIT_↩ COMPILE_FAILED, jpcre2::JIT_ERROR_MESSAGE_PREFIX, and jpcre2::utils::toString().

Here is the call graph for this function:




#### 3.1.3.9 String jpcre2::Regex::getErrorMessage ( int *err_num* ) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use class variable error_offest as error offset.

**Parameters**

| | |
|---|---|
| *err_num* | |

**Returns**

Error message as a string

#### 3.1.3.10 String jpcre2::Regex::getErrorMessage ( ) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts. Use class variable error_number as error number and error_offest as error offset.

**Returns**

Error message as a string (empty if there is no error)

#### 3.1.3.11 Uint jpcre2::Regex::getJpcre2Option ( ) [inline]

Get JPCRE2 option.

**Returns**

jpcre2_compile_opts

#### 3.1.3.12 String jpcre2::Regex::getLocale ( ) [inline]

Get locale as a string.

**Returns**

mylocale

#### 3.1.3.13 String jpcre2::Regex::getModifier ( ) [inline]

Get modifier string.

**Returns**

modifier

#### 3.1.3.14 String jpcre2::Regex::getPattern ( ) [inline]

Get pattern string.

**Returns**

pat_str

#### 3.1.3.15 Uint jpcre2::Regex::getPcre2Option ( ) [inline]

Get PCRE2 option.

**Returns**

compile_opts

#### 3.1.3.16 String jpcre2::Regex::getWarningMessage ( ) [inline]

Get current warning message.

**Returns**

current_warning_msg

#### 3.1.3.17 void jpcre2::Regex::init ( ) [inline], [private]

Call Regex::init_vars() and initialize class variables.

This function should not be attempted to call after creating object. To re-initialize class variables at a later stage after creating object, use the Regex::reset() function. This function is private and should remain as such.

#### 3.1.3.18 void jpcre2::Regex::init ( const String & *re,* const String & *mod* ) [inline], [private]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**Parameters**

| | |
|---|---|
| *re* | Regex pattern |
| *mod* | Modifier string |

#### 3.1.3.19 void jpcre2::Regex::init ( const String & *re,* Uint *po,* Uint *jo* ) [inline], [private]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**Parameters**

| | |
|---|---|
| *re* | Regex pattern |
| *po* | PCRE2 options |
| *jo* | JPCRE2 options |

#### 3.1.3.20 RegexMatch& jpcre2::Regex::initMatch ( ) [inline]

Prepare to call RegexMatch::match().

Other options can be set with the setter functions of RegexMatch class in-between the Regex::initMatch() and RegexMatch::match() call.

**Returns**

RegexMatch object

**See also**

RegexMatch::match()
RegexMatch::setSubject(const String& s)
RegexMatch::setModifier(const String& mod)
RegexMatch::setNumberedSubstringVector(VecNum* vec_num)
RegexMatch::setNamedSubstringVector(VecNas* vec_nas)
RegexMatch::setNameToNumberMapVector(VecNtN* vec_ntn)

References jpcre2::RegexMatch::re.

#### 3.1.3.21 RegexReplace& jpcre2::Regex::initReplace ( ) [inline]

Prepare to call RegexReplace::replace().

Other options can be set with the setter functions of RegexReplace class in-between the Regex::initReplace() and RegexReplace::replace() call.

**Returns**

Resultant string after regex replace

**See also**

RegexReplace::replace()
RegexReplace::setSubject(const String& s)
RegexReplace::setModifier(const String& mod)
RegexReplace::setReplaceWith(const String& s)
RegexReplace::setBufferSize(PCRE2_SIZE x)

References jpcre2::RegexReplace::re.

#### 3.1.3.22 SIZE_T jpcre2::Regex::match ( const String & *s,* const String & *mod* ) [inline]

Perform regex match.

This function takes the parameters, then sets the parameters to RegexMatch class and calls RegexMatch::match() which returns the result

**Parameters**

| | |
|---|---|
| *s* | Subject string |
| *mod* | Modifier string |

**Returns**

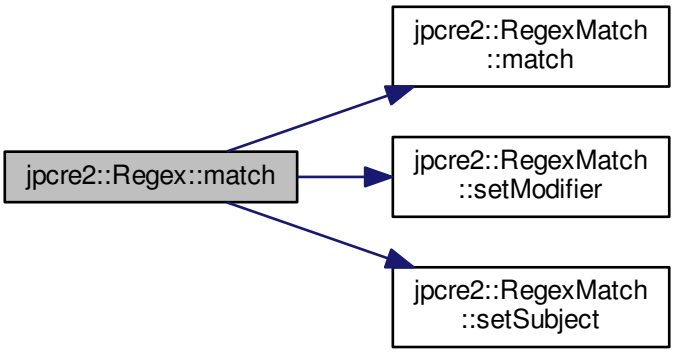
Match count

**See also**

RegexMatch::match()

References jpcre2::RegexMatch::match(), jpcre2::RegexMatch::re, jpcre2::RegexMatch::setModifier(), and jpcre2::RegexMatch::setSubject().

Here is the call graph for this function:




#### 3.1.3.23 SIZE_T jpcre2::Regex::match ( const String & *s* ) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**Parameters**

| | |
|---|---|
| *s* | Subject string |

**Returns**

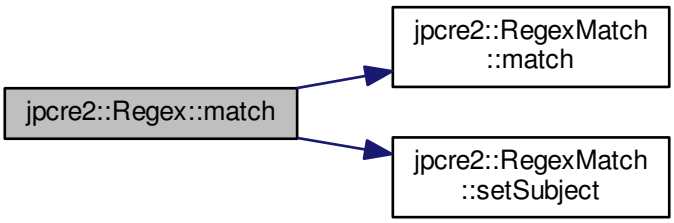
Match count

**See also**

RegexMatch::match(const String& s)

References jpcre2::RegexMatch::match(), jpcre2::RegexMatch::re, and jpcre2::RegexMatch::setSubject().

Here is the call graph for this function:




#### 3.1.3.24 Regex& jpcre2::Regex::operator= ( const Regex & *r* ) [inline]

Overloaded assignment operator.

Performs a deep copy.

Allows assigning objects like this:

```
Regex re;
re = Regex("new pattern");
```

However, use of this method is discouraged (Use Regex::compile() instead), because a call to this function requires an additional call to PCRE2 internal function pcre2_code_copy(). If the pattern was JIT compiled, it requires another additional JIT compilation because JIT memory was not copied by pcre2_code_copy().

**Memory management:** Old JIT memory will be released along with the old compiled code.

**Parameters**

| | |
|---|---|
| *r* | const Regex& |

**Returns**

*this

#### 3.1.3.25 void jpcre2::Regex::parseCompileOpts ( void ) [private]

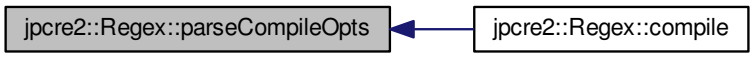
Parse modifier and set equivalent PCRE2 and JPCRE2 options.

After a call to this function compile_opts and jpcre2_compile_opts will be properly set.

References compile_opts, error_number, error_offset, jpcre2::ERROR::INVALID_MODIFIER, jpcre2::JIT_COMP↩ILE, jpcre2_compile_opts, modifier, and jpcre2::VALIDATE_MODIFIER.

Referenced by compile().

Here is the caller graph for this function:




#### 3.1.3.26 Regex& jpcre2::Regex::removeJpcre2Option ( Uint *x* ) [inline]

Remove option from existing JPCRE2 option jpcre2_compile_opts.

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

∗this

#### 3.1.3.27 Regex& jpcre2::Regex::removePcre2Option ( Uint *x* ) [inline]

Remove option from existing PCRE2 option compile_opts.

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

∗this

#### 3.1.3.28 String jpcre2::Regex::replace ( const String & *mains*, const String & *repl*, const String & *mod* ) [inline]

Perform regex replace.

This function takes the parameters, then sets the parameters to RegexReplace class and calls RegexReplace↩::replace() which returns the result.

**Parameters**

| | |
|---|---|
| *mains* | Subject string |
| *repl* | String to replace with |
| *mod* | Modifier string |

**Returns**

Resultant string after regex replace

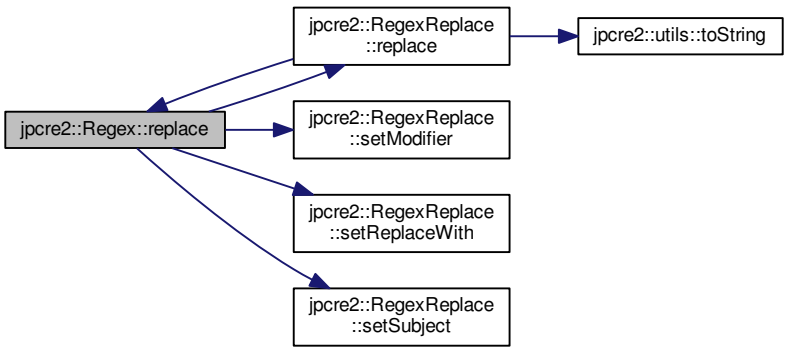
**See also**

RegexReplace::replace()

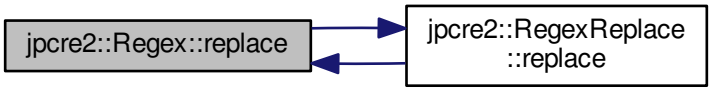
References jpcre2::RegexReplace::re, jpcre2::RegexReplace::replace(), jpcre2::RegexReplace::setModifier(), jpcre2::RegexReplace::setReplaceWith(), and jpcre2::RegexReplace::setSubject().

Referenced by jpcre2::RegexReplace::replace().

Here is the call graph for this function:




Here is the caller graph for this function:




#### 3.1.3.29 String jpcre2::Regex::replace ( const String & *mains*, const String & *repl* ) [inline]

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

**Parameters**

| | |
|---|---|
| *mains* | Subject string |
| *repl* | String to replace with |

**Returns**

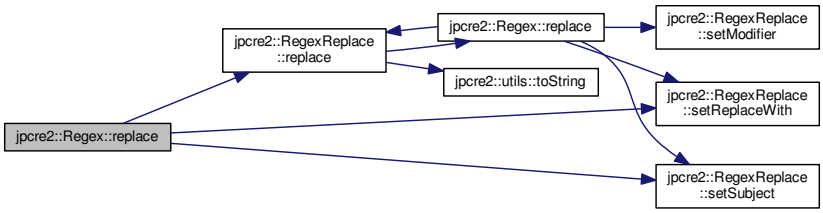
Resultant string after regex replace

**See also**

RegexReplace::replace()

References jpcre2::RegexReplace::re, jpcre2::RegexReplace::replace(), jpcre2::RegexReplace::setReplaceWith(), and jpcre2::RegexReplace::setSubject().

Here is the call graph for this function:




#### 3.1.3.30 Regex& jpcre2::Regex::reset ( ) [inline]

Reset all class variables to its default (initial) state.

**Returns**

∗this

#### 3.1.3.31 Regex& jpcre2::Regex::setJpcre2Option ( Uint *x* ) [inline]

Set JPCRE2 option jpcre2_compile_opts (overwrites existing option)

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

∗this

#### 3.1.3.32 Regex& jpcre2::Regex::setLocale ( const String & *x* ) [inline]

Set the locale mylocale.

**Parameters**

| | |
|---|---|
| *x* | Locale string |

**Returns**

∗this

#### 3.1.3.33 Regex& jpcre2::Regex::setModifier ( const String & *x* ) [inline]

Set the modifier modifier (overwrite existing JPCRE2 and PCRE2 option).

Re-initializes the option bits for PCRE2 and JPCRE2 options, then sets the modifier.

**Parameters**

| | |
|---|---|
| *x* | Modifier string |

**Returns**

∗this

#### 3.1.3.34 Regex& jpcre2::Regex::setPattern ( const String & *re* ) [inline]

Set the Pattern string pat_str.

**Parameters**

| | |
|---|---|
| *re* | Pattern string |

**Returns**

∗this

#### 3.1.3.35 Regex& jpcre2::Regex::setPcre2Option ( Uint *x* ) [inline]

Set PCRE2 option compile_opts (overwrites existing option)

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

*this

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

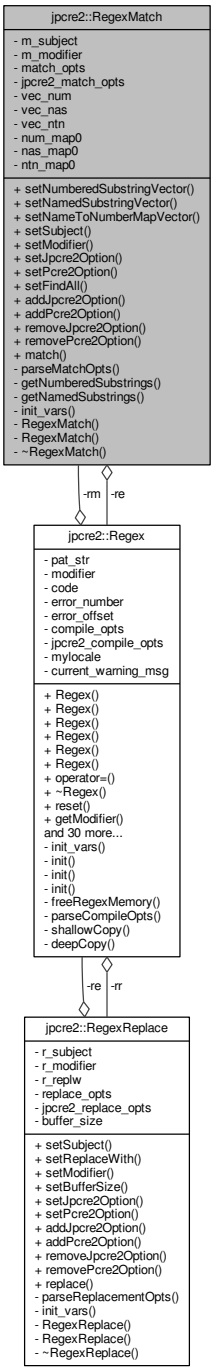
- jpcre2.hpp
- jpcre2.cpp

## 3.2 jpcre2::RegexMatch Class Reference

Performs regex matching.

```
#include <jpcre2.hpp>
```

Collaboration diagram for jpcre2::RegexMatch:




### Public Member Functions

- RegexMatch & setNumberedSubstringVector (VecNum ∗v)

  *Set a pointer to the numbered substring vector of type jpcre2::VecNum.*
- RegexMatch & setNamedSubstringVector (VecNas ∗v)

  *Set a pointer to the named substring vector of type jpcre2::VecNas.*
- RegexMatch & setNameToNumberMapVector (VecNtN ∗v)

*Set a pointer to the name to number map vector of type jpcre2::VecNtN.*

- RegexMatch & setSubject (const String &s)

  *Set the subject string m_subject.*
- RegexMatch & setModifier (const String &s)

  *Set the modifier m_modifier (overwrites existing JPCRE2 and PCRE2 option).*
- RegexMatch & setJpcre2Option (Uint x)

  *Set JPCRE2 option jpcre2_match_opts (overwrite existing option)*
- RegexMatch & setPcre2Option (Uint x)

  *Set PCRE2 option match_opts (overwrite existing option)*
- RegexMatch & setFindAll (bool x=true)

  *Set whether to perform global match.*
- RegexMatch & addJpcre2Option (Uint x)

  *Add option to existing JPCRE2 options jpcre2_match_opts.*
- RegexMatch & addPcre2Option (Uint x)

  *Add option to existing PCRE2 options match_opts.*
- RegexMatch & removeJpcre2Option (Uint x)

  *Remove option from existing JPCRE2 option jpcre2_match_opts.*
- RegexMatch & removePcre2Option (Uint x)

  *Remove option from existing PCRE2 option match_opts.*
- SIZE_T match (void)

  *Return the number of matches, store the match results in the specified vectors (vec_num, vec_nas, vec_ntn)*

#### Private Member Functions

- void parseMatchOpts (void)

  *Parse m_modifier and set equivalent PCRE2 and JPCRE2 options.*
- void getNumberedSubstrings (int rc, pcre2_match_data *match_data)

  *Populate num_map0 with numbered substrings.*
- void getNamedSubstrings (int namecount, int name_entry_size, PCRE2_SPTR tabptr, pcre2_match_data *match_data)

  *Populate nas_map0 and/or ntn_map0 with named substring and/or name to number mapping.*
- void init_vars ()

  *Initialize class variables.*
- RegexMatch ()

  *Default constructor.*
- RegexMatch (const RegexMatch &)

  *This is a copy constructor which is only used to prevent public object creation.*
- ~RegexMatch ()

  *Destructor.*

#### Private Attributes

- Regex * re

  *This is used to access private members in Regex.*
- String m_subject

  *Subject string for match.*
- String m_modifier

  *Pattern for match.*
- Uint match_opts

  *PCRE2 options for pcre2_match() (PCRE2 internal function)*

- Uint jpcre2_match_opts
  *JPCRE2 options for match.*
- VecNum * vec_num
  *Pointer to vector that will store the numbered substring maps.*
- VecNas * vec_nas
  *Pointer to vector that will store the named substring maps.*
- VecNtN * vec_ntn
  *Pointer to vector that will store the name to number maps.*
- MapNum * num_map0
  *Pointer to map that will store numbered substrings temporarily.*
- MapNas * nas_map0
  *Pointer to map that will store named substrings temporarily.*
- MapNtN * ntn_map0
  *Pointer to map that will store name to number mapping temporarily.*

**Friends**

- class Regex
  *Define class Regex as friend and thus allow Regex to create object of this class.*

### 3.2.1 Detailed Description

Performs regex matching.

Provides chained methods to set various options.

All constructors of this class are private.

### 3.2.2 Constructor & Destructor Documentation

#### 3.2.2.1 jpcre2::RegexMatch::RegexMatch ( ) [inline], [private]

Default constructor.

Initialize class variables.

#### 3.2.2.2 jpcre2::RegexMatch::RegexMatch ( const RegexMatch & ) [inline], [private]

This is a copy constructor which is only used to prevent public object creation.

No need to implement it completely

#### 3.2.2.3 jpcre2::RegexMatch::~RegexMatch ( ) [inline], [private]

Destructor.

Deletes the temporary maps that were created to store substrings

### 3.2.3 Member Function Documentation

#### 3.2.3.1 RegexMatch& jpcre2::RegexMatch::addJpcre2Option ( Uint *x* ) [inline]

Add option to existing JPCRE2 options jpcre2_match_opts.

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

∗this

#### 3.2.3.2 RegexMatch& jpcre2::RegexMatch::addPcre2Option ( Uint *x* ) [inline]

Add option to existing PCRE2 options match_opts.

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

∗this

#### 3.2.3.3 jpcre2::SIZE_T jpcre2::RegexMatch::match ( void )

Return the number of matches, store the match results in the specified vectors (vec_num, vec_nas, vec_ntn)

**Returns**

Number of matches found

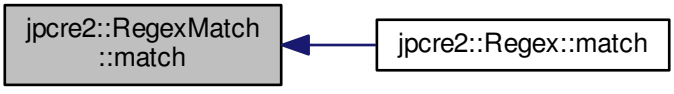
**See also**

SIZE_T match(const String& s)
SIZE_T match(const String& s, const String& mod)

References jpcre2::FIND_ALL.

Referenced by jpcre2::Regex::match().

Here is the caller graph for this function:



#### 3.2.3.4 void jpcre2::RegexMatch::parseMatchOpts ( void ) [private]

Parse m_modifier and set equivalent PCRE2 and JPCRE2 options.

After a call to this function match_opts and jpcre2_match_opts will be properly set.

References jpcre2::FIND_ALL, jpcre2::ERROR::INVALID_MODIFIER, and jpcre2::VALIDATE_MODIFIER.

#### 3.2.3.5 RegexMatch& jpcre2::RegexMatch::removeJpcre2Option ( Uint *x* ) [inline]

Remove option from existing JPCRE2 option jpcre2_match_opts.

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

∗this

#### 3.2.3.6 RegexMatch& jpcre2::RegexMatch::removePcre2Option ( Uint *x* ) [inline]

Remove option from existing PCRE2 option match_opts.

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

∗this

#### 3.2.3.7 RegexMatch& jpcre2::RegexMatch::setFindAll ( bool *x* = true ) [inline]

Set whether to perform global match.

**Parameters**

| | |
|---|---|
| *x* | True or False |

**Returns**

∗this

References jpcre2::FIND_ALL.

#### 3.2.3.8 RegexMatch& jpcre2::RegexMatch::setJpcre2Option ( Uint *x* ) [inline]

Set JPCRE2 option jpcre2_match_opts (overwrite existing option)

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

*this

#### 3.2.3.9 RegexMatch& jpcre2::RegexMatch::setModifier ( const String & *s* ) [inline]

Set the modifier m_modifier (overwrites existing JPCRE2 and PCRE2 option).

Re-initializes the option bits for PCRE2 and JPCRE2 options, then sets the modifier.

**Parameters**

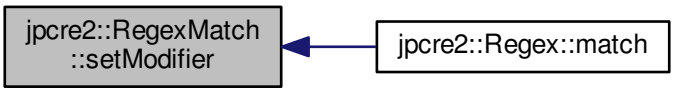
| | |
|---|---|
| *s* | Modifier string |

**Returns**

*this

Referenced by jpcre2::Regex::match().

Here is the caller graph for this function:




#### 3.2.3.10 RegexMatch& jpcre2::RegexMatch::setNamedSubstringVector ( VecNas * *v* ) [inline]

Set a pointer to the named substring vector of type jpcre2::VecNas.

**Parameters**

| | |
|---|---|
| *v* | vec_nas |

**Returns**

*this

#### 3.2.3.11 RegexMatch& jpcre2::RegexMatch::setNameToNumberMapVector ( VecNtN * v ) [inline]

Set a pointer to the name to number map vector of type jpcre2::VecNtN.

**Parameters**

| | |
|---|---|
| *v* | vec_ntn |

**Returns**

*this

#### 3.2.3.12 RegexMatch& jpcre2::RegexMatch::setNumberedSubstringVector ( VecNum * v ) [inline]

Set a pointer to the numbered substring vector of type jpcre2::VecNum.

**Parameters**

| | |
|---|---|
| *v* | vec_num |

**Returns**

*this

#### 3.2.3.13 RegexMatch& jpcre2::RegexMatch::setPcre2Option ( Uint x ) [inline]

Set PCRE2 option match_opts (overwrite existing option)

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

*this

#### 3.2.3.14 RegexMatch& jpcre2::RegexMatch::setSubject ( const String & s ) [inline]

Set the subject string m_subject.

**Parameters**

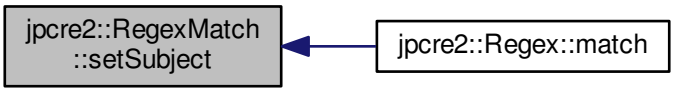
| | |
|---|---|
| *s* | Subject string |

**Returns**

*this

Referenced by jpcre2::Regex::match().

Here is the caller graph for this function:




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

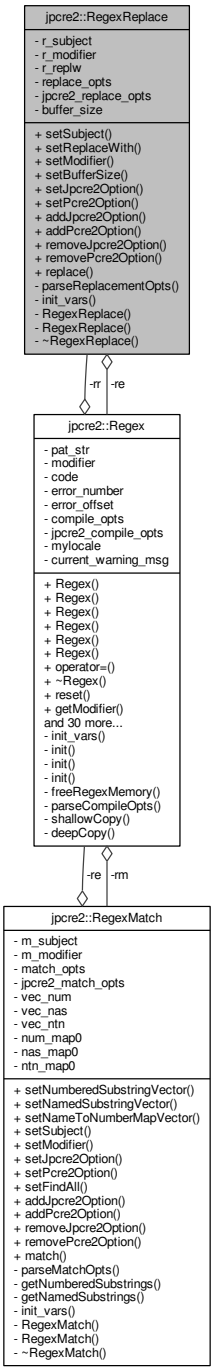
- jpcre2.hpp
- jpcre2.cpp

### 3.3 jpcre2::RegexReplace Class Reference

Performs regex replace on a string.

```
#include <jpcre2.hpp>
```

Collaboration diagram for jpcre2::RegexReplace:




### Public Member Functions

- RegexReplace & setSubject (const String &s)

  *Set the subject string r_subject.*
- RegexReplace & setReplaceWith (const String &s)

  *Set the replacement string r_replw.*
- RegexReplace & setModifier (const String &s)

*Set the modifier string r_modifier (overwrites existing JPCRE2 and PCRE2 option).*

- RegexReplace & setBufferSize (PCRE2_SIZE x)

  *Set the initial buffer size (buffer_size) to be allocated for replaced string (used by PCRE2)*
- RegexReplace & setJpcre2Option (Uint x)

  *Set JPCRE2 option jpcre2_replace_opts (overwrite existing option)*
- RegexReplace & setPcre2Option (Uint x)

  *Set PCRE2 option replace_opts (overwrite existing option)*
- RegexReplace & addJpcre2Option (Uint x)

  *Add specified JPCRE2 option to existing options jpcre2_replace_opts.*
- RegexReplace & addPcre2Option (Uint x)

  *Add specified PCRE2 option to existing options replace_opts.*
- RegexReplace & removeJpcre2Option (Uint x)

  *Remove JPCRE2 option from existing options jpcre2_replace_opts.*
- RegexReplace & removePcre2Option (Uint x)

  *Remove PCRE2 option from existing options replace_opts.*
- String replace (void)

  *Returns the resultant string after performing regex replace.*

#### Private Member Functions

- void parseReplacementOpts (void)

  *Parse r_modifier and set equivalent PCRE2 and JPCRE2 options.*
- void init_vars ()

  *Initialize class variables.*
- RegexReplace ()

  *Default constructor.*
- RegexReplace (const RegexReplace &)

  *This is a copy constructor which is only used to prevent public object creation.*
- ~RegexReplace ()

  *Destructor.*

#### Private Attributes

- Regex * re

  *This is used to access private members in Regex.*
- String r_subject

  *Subject string for replace.*
- String r_modifier

  *Modifier string for replace.*
- String r_replw

  *Replacement string i.e string to replace with.*
- Uint replace_opts

  *PCRE2 options for pcre2_substitute() (PCRE2 internal function)*
- Uint jpcre2_replace_opts

  *JPCRE2 options.*
- PCRE2_SIZE buffer_size

  *Size of the resultant string after replacement.*

#### Friends

- class Regex

  *Define Regex as a friend so that it can create object of this class.*

### 3.3.1 Detailed Description

Performs regex replace on a string.

Provides chained methods to set various options.

All constructors of this class are private.

### 3.3.2 Constructor & Destructor Documentation

#### 3.3.2.1 jpcre2::RegexReplace::RegexReplace ( ) [inline], [private]

Default constructor.

Initialize class variables

#### 3.3.2.2 jpcre2::RegexReplace::RegexReplace ( const RegexReplace & ) [inline], [private]

This is a copy constructor which is only used to prevent public object creation.

No need to implement it completely

#### 3.3.2.3 jpcre2::RegexReplace::~RegexReplace ( ) [inline], [private]

Destructor.

Nothing to be done here.

### 3.3.3 Member Function Documentation

#### 3.3.3.1 RegexReplace& jpcre2::RegexReplace::addJpcre2Option ( Uint *x* ) [inline]

Add specified JPCRE2 option to existing options jpcre2_replace_opts.

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

*this

#### 3.3.3.2 RegexReplace& jpcre2::RegexReplace::addPcre2Option ( Uint *x* ) [inline]

Add specified PCRE2 option to existing options replace_opts.

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

∗this

#### 3.3.3.3 void jpcre2::RegexReplace::parseReplacementOpts ( void ) [private]

Parse r_modifier and set equivalent PCRE2 and JPCRE2 options.

After a call to this function replace_opts and jpcre2_replace_opts will be properly set.

References jpcre2::ERROR::INVALID_MODIFIER, and jpcre2::VALIDATE_MODIFIER.

#### 3.3.3.4 RegexReplace& jpcre2::RegexReplace::removeJpcre2Option ( Uint *x* ) [inline]

Remove JPCRE2 option from existing options jpcre2_replace_opts.

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

∗this

#### 3.3.3.5 RegexReplace& jpcre2::RegexReplace::removePcre2Option ( Uint *x* ) [inline]

Remove PCRE2 option from existing options replace_opts.

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

∗this

#### 3.3.3.6 jpcre2::String jpcre2::RegexReplace::replace ( void )

Returns the resultant string after performing regex replace.

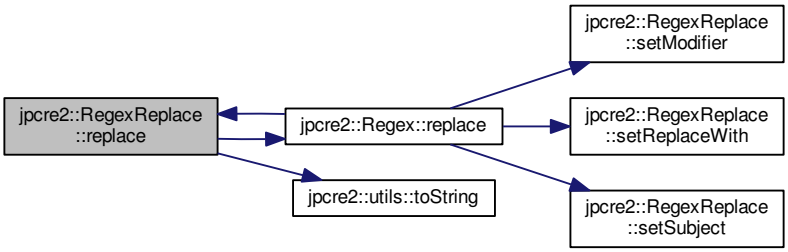
**Returns**

Replaced string

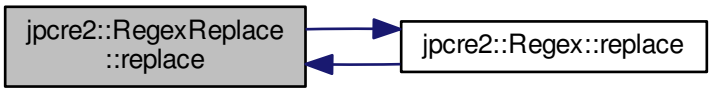
References jpcre2::Regex::replace(), and jpcre2::utils::toString().

Referenced by jpcre2::Regex::replace().

Here is the call graph for this function:




Here is the caller graph for this function:




#### 3.3.3.7 RegexReplace& jpcre2::RegexReplace::setBufferSize ( PCRE2_SIZE *x* ) [inline]

Set the initial buffer size (buffer_size) to be allocated for replaced string (used by PCRE2)

**Parameters**

| | |
|---|---|
| *x* | Buffer size |

**Returns**

∗this

#### 3.3.3.8 RegexReplace& jpcre2::RegexReplace::setJpcre2Option ( Uint *x* ) [inline]

Set JPCRE2 option jpcre2_replace_opts (overwrite existing option)

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

∗this

#### 3.3.3.9 RegexReplace& jpcre2::RegexReplace::setModifier ( const String & *s* ) [inline]

Set the modifier string r_modifier (overwrites existing JPCRE2 and PCRE2 option).

**Parameters**

| | |
|---|---|
| *s* | Modifier string |

**Returns**

∗this

Referenced by jpcre2::Regex::replace().

Here is the caller graph for this function:


#### 3.3.3.10 RegexReplace& jpcre2::RegexReplace::setPcre2Option ( Uint *x* ) [inline]

Set PCRE2 option replace_opts (overwrite existing option)

**Parameters**

| | |
|---|---|
| *x* | Option value |

**Returns**

∗this

#### 3.3.3.11 RegexReplace& jpcre2::RegexReplace::setReplaceWith ( const String & *s* ) [inline]

Set the replacement string r_replw.

**Parameters**

| | |
|---|---|
| *s* | String to replace with |

**Returns**

∗this

Referenced by jpcre2::Regex::replace().

Here is the caller graph for this function:


#### 3.3.3.12 RegexReplace& jpcre2::RegexReplace::setSubject ( const String & *s* ) [inline]

Set the subject string r_subject.

**Parameters**

| | |
|---|---|
| *s* | Subject string |

**Returns**

∗this

Referenced by jpcre2::Regex::replace().

Here is the caller graph for this function:


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

- jpcre2.hpp
- jpcre2.cpp

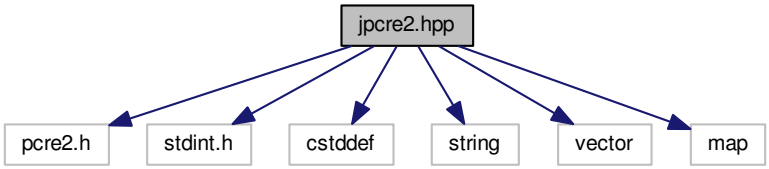
# 4 File Documentation

## 4.1 jpcre2.hpp File Reference

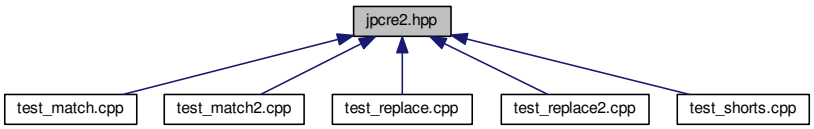
Main header file for JPCRE2 library to be included by programs that uses its functionalities.

```
#include <pcre2.h>
#include <stdint.h>
#include <cstddef>
#include <string>
#include <vector>
#include <map>
```

Include dependency graph for jpcre2.hpp:




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




### Classes

- class jpcre2::RegexMatch

  *Performs regex matching.*
- class jpcre2::RegexReplace

  *Performs regex replace on a string.*
- class jpcre2::Regex

  *Implements public overloaded and copy constructors, provides functions to set/unset various options and perform regex match and replace against a compiled pattern.*

### Namespaces

- jpcre2

  *Top level namespace of JPCRE2.*
- jpcre2::ERROR

  *Namespace for error codes.*
- jpcre2::utils

  *Namespace for some utility functions.*

### Macros

- #define PCRE2_CODE_UNIT_WIDTH 8

  *Code unit width 8 is used by default.*

### Typedefs

- typedef std::size_t jpcre2::SIZE_T

  *Used for match count and vector size.*
- typedef uint32_t jpcre2::Uint

  *Used for options (bitwise operation)*
- typedef std::string jpcre2::String

  *Used as std::string.*
- typedef std::map< String, String > jpcre2::MapNas

  *Map for Named substrings.*
- typedef std::map< SIZE_T, String > jpcre2::MapNum

  *Map for Numbered substrings.*
- typedef std::map< String, SIZE_T > jpcre2::MapNtN

  *Substring name to Substring number map.*
- typedef MapNtN jpcre2::MapNtn

  *Allow spelling mistake of MapNtN as MapNtn.*
- typedef std::vector< MapNas > jpcre2::VecNas

  *Vector of matches with named substrings.*
- typedef std::vector< MapNtN > jpcre2::VecNtN

  *Vector of substring name to Substring number map.*
- typedef VecNtN jpcre2::VecNtn

  *Allow spelling mistake of VecNtN as VecNtn.*
- typedef std::vector< MapNum > jpcre2::VecNum

  *Vector of matches with numbered substrings.*

### Enumerations

### Functions

- String jpcre2::utils::toString (int a)

  *Converts an integer to String.*
- String jpcre2::utils::toString (char a)

  *Converts a char to String.*
- String jpcre2::utils::toString (const char *a)

  *Converts const char* to String.*
- String jpcre2::utils::toString (PCRE2_UCHAR *a)

  *Converts a PCRE2_UCHAR* to String.*
- String jpcre2::utils::getPcre2ErrorMessage (int err_num)

  *Get PCRE2 error message for an error number.*

#### Variables

- const SIZE_T jpcre2::SUBSTITUTE_RESULT_INIT_SIZE = std::numeric_limits<int>::max()

  *Used by default to provide big enough initial buffer for replaced string.*
- const String jpcre2::LOCALE_NONE = "JPCRE2_NONE"

  *Don't do anything about locale if it is set to LOCALE_NONE.*
- const String jpcre2::LOCALE_DEFAULT = LOCALE_NONE

  *Default locale.*
- const String jpcre2::JIT_ERROR_MESSAGE_PREFIX = "JIT compilation failed! "

  *Prefix to be added to JIT error message.*

### 4.1.1 Detailed Description

Main header file for JPCRE2 library to be included by programs that uses its functionalities.

It includes the pcre2.h header, therefore you shouldn't include pcre2.h separately in your program. Make sure to link both jpcre2 and pcre2 library when compiling.

If you are using JPCRE2 with all of its source files, you won't need to link it with JPCRE2 library, but do remember that you still need to link with pcre2 library

#### Author

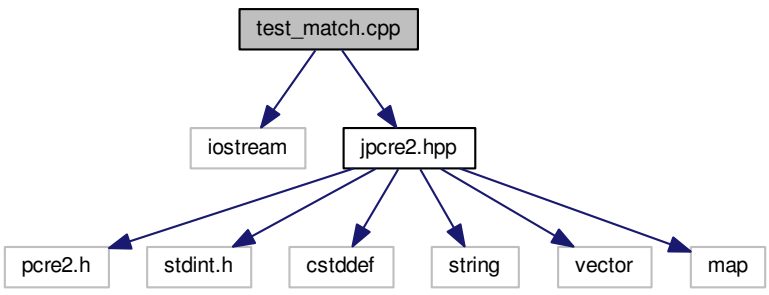
Md Jahidul Hamid

## 4.2 test_match.cpp File Reference

An example of performing regex match against a pattern with JPCRE2 and getting the match count and match results.

```
#include <iostream>
#include "jpcre2.hpp"
```

Include dependency graph for test_match.cpp:



### 4.2.1 Detailed Description

An example of performing regex match against a pattern with JPCRE2 and getting the match count and match results.

Shows how to iterate over the match results to get the captured groups/substrings.

```
/**@file test_match.cpp
 * An example of performing regex match against a pattern with JPCRE2 and getting the
 * match count and match results.
 * Shows how to iterate over the match results to get the captured groups/substrings.
 * @include test_match.cpp
 * @author [Md Jahidul Hamid](https://github.com/neurobin)
 * */

#include <iostream>
#include "jpcre2.hpp"


int main(){

    jpcre2::VecNum vec_num0;    ///Vector to store numbered substring Maps.
    jpcre2::VecNas vec_nas0;    ///Vector to store named substring Maps.
    jpcre2::VecNtN vec_nn0;     ///Vector to store Named substring to Number Maps.

    jpcre2::Regex re;     ///An empty object is not supposed to throw any exception in normal
       cases.

    ///Compile the pattern
    try{re.setPattern("(?:(?<word>[?.#@:]+)|(?<word>\\w+))\\s*(?<digit>\\d+)")  //set pattern
          .setModifier("nJ")                                                    //set modifier
          .addJpcre2Option(jpcre2::VALIDATE_MODIFIER
                   //modifier goes through validation check
                           | jpcre2::JIT_COMPILE                                //
      perform JIT compile
                           | jpcre2::ERROR_ALL)                                 //treat
       warnings as errors
          .addPcre2Option(0)                                                    //add pcre2
       option
          .compile();}                                                          //Finally compile
       it.
    catch(int e){std::cerr<<re.getErrorMessage(e);}

    /*************************************************************************************************
      *********
     * Always use try catch to catch any exception and avoid unexpected termination of the program.
     * All jpcre2 exceptions are of type int (integer)
     * ***********************************************************************************************
      *********/

    ///subject string
    std::string subject = "(I am a string with words and digits 45 and specials chars: ?.#@ 443
        56)";

    size_t count=0;

    try{count = re.initMatch()                                  //Invoke the initMatch() function
                  .setModifier("g")                             //set various parameters
                  .setSubject(subject)                          //...
                  .setNumberedSubstringVector(&vec_num0)        //...
                  .setNamedSubstringVector(&vec_nas0)           //...
                  .setNameToNumberMapVector(&vec_nn0)           //...
                  .addJpcre2Option(jpcre2::VALIDATE_MODIFIER)   //
      ...
                  .addPcre2Option(0)                            //...
                  .match();}                                    //Finaly perform the match
    catch(int e){std::cerr<<re.getErrorMessage(e);}


    /// re.reset(); /// re-initialize re


    std::cout<<"\nTotal number of mathces: "<<count<<std::endl;
    ///Now let's access the matched data

    ///Each of these vectors contains maps.
    ///Each element in the vector specifies a particular match
    ///First match is the vector element 0, second is at index 1 and so forth
    ///A map for a vector element, i.e for a match contains all of its substrings/capture groups
    ///The first element of the map is capture group 0 i.e total match
```

```
    for(size_t i=0;i<vec_num0.size();++i){


        std::cout<< "\n################## Match no: "<<i+1<<" ####################\n";



        ///This vector contains maps with number as the key and the corresponding substring as the value
        std::cout<<"\n-------------------------------------------------------------------------";
        std::cout<< "\n--- Numbered Substrings (number: substring) for match "<<i+1<<" ---\n";
        for(jpcre2::MapNum::iterator ent=vec_num0[i].begin();ent!=vec_num0[i].end();++ent){
            std::cout<<"\n\t"<<ent->first<<": "<<ent->second<<"\n";
        }



        ///This vector contains maps with name as the key and the corresponding substring as the value
        std::cout<<"\n-------------------------------------------------------------------------";
        std::cout<< "\n--- Named Substrings (name: substring) for match "<<i+1<<" ---\n";
        for(jpcre2::MapNas::iterator ent=vec_nas0[i].begin();ent!=vec_nas0[i].end();++ent){
            std::cout<<"\n\t"<<ent->first<<": "<<ent->second<<"\n";
        }



        ///This vector contains maps with name as the key and number as the value
        ///i.e the number (of substring) can be accessed with the name for named substring.
        std::cout<<"\n-------------------------------------------------------------------------";
        std::cout<< "\n--- Name to number mapping (name: number/position) for match "<<i+1<<" ---\n";
        for(jpcre2::MapNtN::iterator ent=vec_nn0[i].begin();ent!=vec_nn0[i].end();++ent){
            std::cout<<"\n\t"<<ent->first<<": "<<ent->second<<"\n";
        }
    }
    return 0;
}
```

**Author**

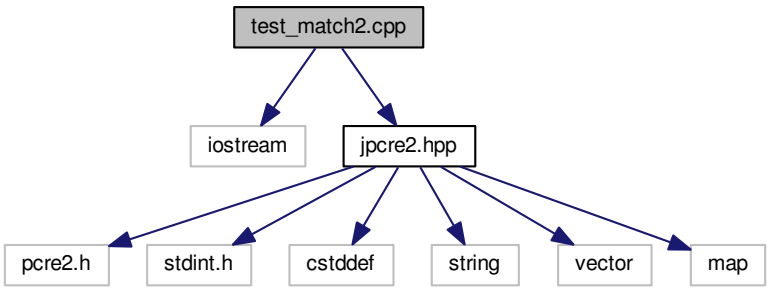
Md Jahidul Hamid

## 4.3 test_match2.cpp File Reference

Contains an example to take subject string, pattern and modifier from user input and perform regex match using JPCRE2.

```
#include <iostream>
#include "jpcre2.hpp"
```

Include dependency graph for test_match2.cpp:



### 4.3.1 Detailed Description

Contains an example to take subject string, pattern and modifier from user input and perform regex match using JPCRE2.

```
/**@file test_match2.cpp
 * Contains an example to take subject string, pattern and modifier
 * from user input and perform regex match using JPCRE2.
 * @include test_match2.cpp
 * @author [Md Jahidul Hamid](https://github.com/neurobin)
 * */

#include <iostream>
#include "jpcre2.hpp"


#define getLine(a) std::getline(std::cin,a,'\n')


int main(){

    jpcre2::VecNum vec_num0;    ///Vector to store numbered substring Map.
    jpcre2::VecNas vec_nas0;    ///Vector to store named substring Map.
    jpcre2::VecNtN vec_nn0;     ///Vector to store Named substring to Number Map.


    std::string pat,mod,subject;

    ///create an object
    jpcre2::Regex re;     /// This should not throw any exception

    std::cout<<"Enter pattern: ";
    getLine(pat);
    cp:
    std::cout<<"Enter compile modifiers (eijmnsuxADJSU): ";
    getLine(mod);

    ///Compile pattern
    try{re.compile(pat,mod);}
    catch(int e){std::cerr<<re.getErrorMessage(e)<<std::endl;goto cp;}

    /**************************************************************************************************
      *********
     * Use try catch block to catch any exception and avoid unexpected termination of the program in case
       of error
     * All jpcre2 exceptions are of type int (integer)
     * ************************************************************************************************
      *********/


    ///subject string
    std::cout<<"\nEnter subject string (enter quit to quit): "<<std::endl;
    getLine(subject);
    std::string ac_mod;

    size_t matched = 0;
    /// Continue loop as long as error occurs
    while(true){
        std::cout<<"\nEnter action (matching) modifier (Ag): "<<std::endl;
        getLine(ac_mod);
        if(subject=="quit")return 0;
        try{matched=re.initMatch()                                  //Invoke the initMatch()
       function
                    .setModifier(ac_mod)                            //Set various options
                    .setNumberedSubstringVector(&vec_num0)          //...
                    .setNamedSubstringVector(&vec_nas0)             //...
                    .setNameToNumberMapVector(&vec_nn0)             //...
                    .addJpcre2Option(jpcre2::VALIDATE_MODIFIER)
      //...
                    .addPcre2Option(0)                              //...
                    .match();                                       //Finally do the match
        }
        catch(int e){std::cerr<<re.getErrorMessage(e);
            if(e==jpcre2::ERROR::INVALID_MODIFIER) continue;
        }
        break;
    }
    ///Now let's access the matched data

    ///Each of these vectors contains maps.
    ///Each element in the vector specifies a particular match
    ///First match is the vector element 0, second is at index 1 and so forth
    ///A map for a vector element, i.e for a match contains all of its substrings/capture groups
```

```
    ///The first element of the map is capture group 0 i.e total match
    std::cout<<"\nTotal number of matches: "<<matched<<std::endl;
    if(matched){
        for(size_t i=0;i<vec_num0.size();++i){


            std::cout<< "\n################### Match no: "<<i+1<<" ####################\n";



            ///This vector contains maps with number as the key and the corresponding substring as the
       value
            std::cout<<"\n--------------------------------------------------------------------------------";
            std::cout<< "\n--- Numbered Substrings (number: substring) for match "<<i+1<<" ---\n";
            for(jpcre2::MapNum::iterator ent=vec_num0[i].begin();ent!=vec_num0[i].end();++ent){
                std::cout<<"\n\t"<<ent->first<<": "<<ent->second<<"\n";
            }



            ///This vector contains maps with name as the key and the corresponding substring as the value
            std::cout<<"\n--------------------------------------------------------------------------------";
            std::cout<< "\n--- Named Substrings (name: substring) for match "<<i+1<<" ---\n";
            for(jpcre2::MapNas::iterator ent=vec_nas0[i].begin();ent!=vec_nas0[i].end();++ent){
                std::cout<<"\n\t"<<ent->first<<": "<<ent->second<<"\n";
            }



            ///This vector contains maps with name as the key and number as the value
            ///i.e the number (of substring) can be accessed with the name for named substring.
            std::cout<<"\n--------------------------------------------------------------------------------";
            std::cout<< "\n--- Name to number mapping (name: number/position) for match "<<i+1<<" ---\n";
            for(jpcre2::MapNtN::iterator ent=vec_nn0[i].begin();ent!=vec_nn0[i].end();++ent){
                std::cout<<"\n\t"<<ent->first<<": "<<ent->second<<"\n";
            }
        }
    }
    else std::cout<<"\nNo match found\n";
    //main();
    return 0;
}
```

**Author**

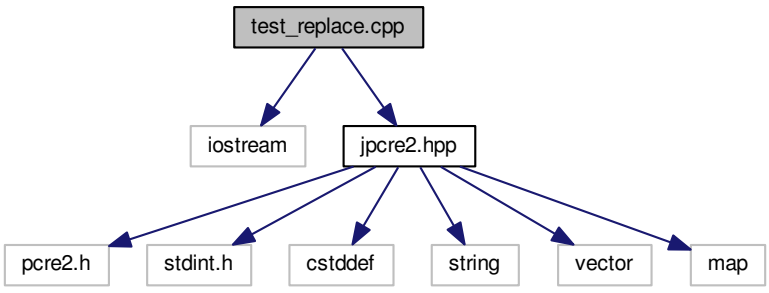
Md Jahidul Hamid

## 4.4 test_replace.cpp File Reference

An example of doing regex replace with JPCRE2.

```
#include <iostream>
#include "jpcre2.hpp"
```

Include dependency graph for test_replace.cpp:



### 4.4.1 Detailed Description

An example of doing regex replace with JPCRE2.

```
/**@file test_replace.cpp
 * An example of doing regex replace with JPCRE2
 * @include test_replace.cpp
 * @author [Md Jahidul Hamid](https://github.com/neurobin)
 * */

#include <iostream>
#include "jpcre2.hpp"


int main(){
    jpcre2::Regex re;     /// This is not supposed to throw any exception.

    ///Compile the pattern
    try{re.setPattern("(?:(?<word>[?.#@:]+)|(?<word>\\w+))\\s*(?<digit>\\d+)")     //Set various
       parameters
          .setModifier("Jin")                                                     //...
          .addJpcre2Option(jpcre2::VALIDATE_MODIFIER)
                     //...
          .addPcre2Option(0)                                                      //...
          .compile();}                                                            //Finally compile
       it.
    catch(int e){std::cerr<<re.getErrorMessage(e);}

    /**************************************************************************************************
      ************
     * Use try catch block to catch any exception and avoid unexpected termination of the program in case
       of error
     * All jpcre2 exceptions are of type int (integer)
     * ************************************************************************************************
      ************/

    //subject string
    std::string s="I am a string with words and digits 45 and specials chars: ?.#@ 443        56";

    try{std::cout<<"\nreplaced string: \n"<<
        re.initReplace()                                              //Invoke the
       initReplace() function
          .setSubject(s)                                              //Set various
       parameters
          .setReplaceWith("(replaced:$1)(replaced:$2)(replaced:${word})")    //...
          .setModifier("xE")                                          //...
          .addJpcre2Option(jpcre2::VALIDATE_MODIFIER)
               //...
          .addPcre2Option(0)                                          //...
          .replace();                                                 //Finally perform the
       replace operation.
    }
    catch(int e){std::cerr<<re.getErrorMessage(e);}

    return 0;
}
```

#### Author

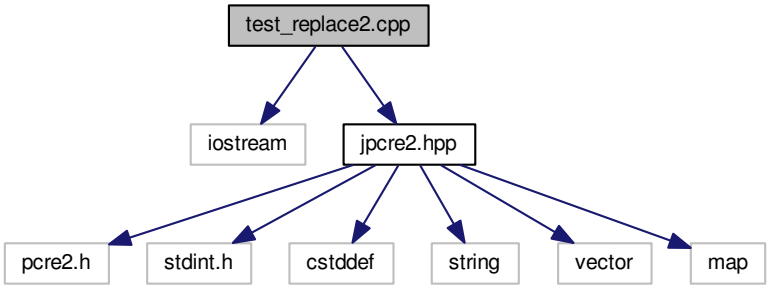
Md Jahidul Hamid

## 4.5 test_replace2.cpp File Reference

Contains an example to take subject string, replacement string, modifier and pattern from user input and perform regex replace with JPCRE2.

```
#include <iostream>
#include "jpcre2.hpp"
```

Include dependency graph for test_replace2.cpp:




### 4.5.1 Detailed Description

Contains an example to take subject string, replacement string, modifier and pattern from user input and perform regex replace with JPCRE2.

```
/**@file test_replace2.cpp
 * Contains an example to take subject string, replacement string, modifier and pattern
 * from user input and perform regex replace with JPCRE2
 * @include test_replace2.cpp
 * @author [Md Jahidul Hamid](https://github.com/neurobin)
 * */


#include <iostream>
#include "jpcre2.hpp"


#define getLine(a) std::getline(std::cin,a,'\n')


int main(){
    std::string pat,mod,subject,repl,repl_mod;

    std::cout<<"\nEnter pattern: ";
    getLine(pat);

    std::cout<<"\nEnter compile modifiers (eijmnsuxADJSU): ";
    getLine(mod);
    jpcre2::Regex re;     /// This is not supposed to throw any exception.


    /// Compile the pattern
    try{re.compile(pat,mod);}
    catch(int e){std::cerr<<re.getErrorMessage(e);}


    /**************************************************************************************************************
      *********
     * Use try catch block to catch any exception and avoid unexpected termination of the program in case
       of error.
     * All jpcre2 exceptions are of type int (integer)
     * ************************************************************************************************************
      *********/


    ///subject string
    std::cout<<"\nEnter subject string (enter quit to quit): "<<std::endl;
    getLine(subject);
    if(subject=="quit")return 0;
     ///replacement string
    std::cout<<"\nEnter replacement string: "<<std::endl;
    getLine(repl);

    /// Continue loop as long as error occurs
```

```
    while(true){
        std::cout<<"\nEnter action (replacement) modifiers (eEgx): ";
        getLine(repl_mod);

        //perform replace

        try{std::cout<<"\nreplaced string: "<<re.initReplace()
                                                .setSubject(subject)
                                                .setReplaceWith(repl)
                                                .setModifier(repl_mod)
                                                .addJpcre2Option(
      jpcre2::VALIDATE_MODIFIER)
                                                .replace();}
        catch(int e){std::cerr<<re.getErrorMessage(e);
            if(e==jpcre2::ERROR::INVALID_MODIFIER) continue;
        }
        break;
    }
    std::cout<<"\n\n----------------------------------------------------\n";
    //main();
    return 0;
}
```

**Author**

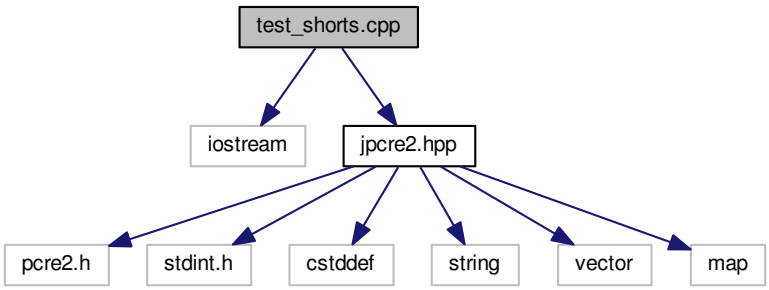
Md Jahidul Hamid

## 4.6 test_shorts.cpp File Reference

Contains some short examples of performing regex match and regex replace with JPCRE2.

```
#include <iostream>
#include "jpcre2.hpp"
```

Include dependency graph for test_shorts.cpp:




### 4.6.1 Detailed Description

Contains some short examples of performing regex match and regex replace with JPCRE2.

```
/** @file test_shorts.cpp
 *  Contains some short examples of performing regex match and regex replace with JPCRE2
 *  @include test_shorts.cpp
 * @author [Md Jahidul Hamid](https://github.com/neurobin)
 *  */

#include <iostream>
#include "jpcre2.hpp"
```

```
int main(){
    size_t count;
    ///Check if string matches the pattern
    /**
     * The following uses a temporary Regex object.
     * */
    if(jpcre2::Regex("(\\d)|(\\w)").match("I am the subject"))
        std::cout<<"\nmatched";
    else
        std::cout<<"\nno match";
    /**
     * Using the modifier S (i.e jpcre2::JIT_COMPILE) with temporary object may or may not give you
     * any performance boost (depends on the complexity of the pattern). The more complex
     * the pattern gets, the more sense the S modifier makes.
     * */

    ///If you want to match all and get the match count, use the action modifier 'g':
    std::cout<<"\n"<<
        jpcre2::Regex("(\\d)|(\\w)","m").match("I am the subject","g");

    /**
     * Modifiers passed to the Regex constructor or with compile() function are compile modifiers
     * Modifiers passed with the match() or replace() functions are action modifiers
     * */

    /// Substrings/Captured groups:

    /**
     * *** Getting captured groups/substring ***
     *
     * captured groups or substrings are stored in maps for each match,
     * and each match is stored in a vector.
     * Thus captured groups are in a vector of maps.
     *
     * PCRE2 provides two types of substrings:
     *  1. numbered (index) substring
     *  2. named substring
     *
     * For the above two, we have two vectors respectively:
     *  1. jpcre2::VecNum (Corresponding map: jpcre2::MapNum)
     *  2. jpcre2::VecNas (Corresponding map: jpcre2::MapNas)
     *
     * Another additional vector is available to get the substring position/number
     * for a particular captured group by name. It's a vector of name to number maps
     *  * jpcre2::VecNtN (Corresponding map: jpcre2:MapNtN)
     * */

    /// ***** Get numbered substring ***** ///
    jpcre2::VecNum vec_num;
    count =
    jpcre2::Regex("(\\w+)\\s*(\\d+)","m")
            .initMatch()
            .setSubject("I am 23, I am digits 10")
            .setModifier("g")
            .setNumberedSubstringVector(&vec_num)
            .match();
    /**
    * count (the return value) is guaranteed to give you the correct number of matches,
    * while vec_num.size() may give you wrong result if any match result
    * was failed to be inserted in the vector. This should not happen
    * i.e count and vec_num.size() should always be equal.
    * */
    std::cout<<"\nNumber of matches: "<<count/* or vec_num.size()*/;

    ///Now vec_num is populated with numbered substrings for each match
    ///The size of vec_num is the total match count
    ///vec_num[0] is the first match
    ///The type of vec_num[0] is jpcre2::MapNum
    std::cout<<"\nTotal match of first match: "<<vec_num[0][0];      ///Total match (group 0) from first
       match
    std::cout<<"\nCaptrued group 1 of frist match: "<<vec_num[0][1]; ///captured group 1 from first match
    std::cout<<"\nCaptrued group 2 of frist match: "<<vec_num[0][2]; ///captured group 2 from first match
    std::cout<<"\nCaptrued group 3 of frist match: "<<vec_num[0][3]; ///captured group 3 doesn't exist, it
       will give you empty string
    ///Using the [] operator with jpcre2::MapNum will create new element if it doesn't exist
    /// i.e vec_num[0][3] were created in the above example.
    ///This should be ok, if existence of a particular substring is not important

    ///If the existence of a substring is important, use the std::map::find() or std::map::at() (>=C++11)
       function to access map elements
    /* //>=C++11
    try{
        ///This will throw exception, because substring 4 doesn't exist
        std::cout<<"\nCaptrued group 4 of frist match: "<<vec_num[0].at(4);
    } catch (std::logic_error e){
```

```
        std::cerr<<"\nCaptrued group 4 doesn't exist";
    }*/

    ///There were two matches found (vec_num.size() == 2) in the above example
    std::cout<<"\nTotal match of second match: "<<vec_num[1][0];      ///Total match (group 0) from second
       match
    std::cout<<"\nCaptrued group 1 of second match: "<<vec_num[1][1]; ///captured group 1 from second match

    std::cout<<"\nCaptrued group 2 of second match: "<<vec_num[1][2]; ///captured group 2 from second match


    /// ***** Get named substring ***** ///

    jpcre2::VecNas vec_nas;
    jpcre2::VecNtN vec_ntn; /// We will get name to number map vector too
    count =
    jpcre2::Regex("(?<word>\\w+)\\s*(?<digit>\\d+)","m")
            .initMatch()
            .setSubject("I am 23, I am digits 10")
            .setModifier("g")
            ///.setNumberedSubstringVector(vec_num) /// We don't need it in this example
            .setNamedSubstringVector(&vec_nas)
            .setNameToNumberMapVector(&vec_ntn) /// Additional (name to number
       maps)
            .match();
    std::cout<<"\nNumber of matches: "<<vec_nas.size()/* or count */;
    ///Now vec_nas is populated with named substrings for each match
    ///The size of vec_nas is the total match count
    ///vec_nas[0] is the first match
    ///The type of vec_nas[0] is jpcre2::MapNas
    std::cout<<"\nCaptured group (word) of first match: "<<vec_nas[0]["word"];
    std::cout<<"\nCaptured group (digit) of first match: "<<vec_nas[0]["digit"];

    ///If the existence of a substring is important, use the std::map::find() or std::map::at() (>=C++11)
       function to access map elements
    /* //>=C++11
    try{
        ///This will throw exception becasue the substring name 'name' doesn't exist
        std::cout<<"\nCaptured group (name) of first match: "<<vec_nas[0].at("name");
    } catch(std::logic_error e){
        std::cerr<<"\nCaptured group (name) doesn't exist";
    }*/

    ///There were two matches found (vec_nas.size() == 2) in the above example
    std::cout<<"\nCaptured group (word) of second match: "<<vec_nas[1]["word"];
    std::cout<<"\nCaptured group (digit) of second match: "<<vec_nas[1]["digit"];

    ///Get the position (number) of a captured group name (that was found in match)
    std::cout<<"\nPosition of captured group (word) in first match: "<<vec_ntn[0]["word"];
    std::cout<<"\nPosition of captured group (digit) in first match: "<<vec_ntn[0]["digit"];

    /**
     * Replacement Examples
     * Replace pattern in a string with a replacement string
     *
     * The initReplace() function can take a subject and replacement string as argument.
     * You can also pass the subject with setSubject() function in method chain,
     * replacement string with setReplaceWith() function in method chain, etc ...
     *
     * A call to replace() will return the resultant string
     * */

    std::cout<<"\n"<<
    ///replace first occurrence of a digit with @
    jpcre2::Regex("\\d").replace("I am the subject string 44", "@");

    std::cout<<"\n"<<
    ///replace all occrrences of a digit with @
    jpcre2::Regex("\\d").replace("I am the subject string 44", "@", "g");

    ///swap two parts of a string
    std::cout<<"\n"<<
    jpcre2::Regex("^([^\t]+)\t([^\t]+)$")
            .replace("I am the subject\tTo be swapped according to tab", "$2 $1");


    return 0;
}
```

## Author

Md Jahidul Hamid