GHAdoxy

Generated by Doxygen 1.9.4

1 Hierarchical Index 1

1.1 Class Hierarchy . . . 1

2 Class Index 3

2.1 Class List . . . 3

3 File Index 5

3.1 File List . . . 5

4 Class Documentation 7

4.1 Cocktail Class Reference . . . 7

4.1.1 Detailed Description . . . 7

4.1.2 Constructor & Destructor Documentation . . . 8

4.1.2.1 Cocktail() [1/2] . . . 8

4.1.2.2 Cocktail() [2/2] . . . 8

4.1.3 Member Function Documentation . . . 8

4.1.3.1 print() . . . 8

4.2 Cocktailbook Class Reference . . . 9

4.2.1 Detailed Description . . . 9

4.2.2 Constructor & Destructor Documentation . . . 9

4.2.2.1 Cocktailbook() . . . 9

4.2.3 Member Function Documentation . . . 10

4.2.3.1 atCopy() . . . 10

4.2.3.2 list() [1/2] . . . 10

4.2.3.3 list() [2/2] . . . 10

4.2.3.4 operator+() . . . 11

4.2.3.5 operator-() . . . 11

4.2.3.6 operator[]() . . . 11

4.2.3.7 pushback() . . . 12

4.3 Drink Class Reference . . . 12

4.3.1 Detailed Description . . . 13

4.3.2 Constructor & Destructor Documentation . . . 13

4.3.2.1 Drink() [1/3] . . . 13

4.3.2.2 Drink() [2/3] . . . 14

4.3.2.3 Drink() [3/3] . . . 14

4.3.3 Member Function Documentation . . . 14

4.3.3.1 print() . . . 14

4.3.4 Friends And Related Function Documentation . . . 14

4.3.4.1 operator<< . . . 14

4.3.5 Member Data Documentation . . . 15

4.3.5.1 flavour . . . 15

4.3.5.2 prep . . . 15

4.3.5.3 served_hot . . . 15

4.3.5.4 texture . . . 15
4.4 Tea Class Reference . . . 15
4.4.1 Detailed Description . . . 16
4.4.2 Constructor & Destructor Documentation . . . 16
4.4.2.1 Tea() [1/2] . . . 16
4.4.2.2 Tea() [2/2] . . . 16
4.4.3 Member Function Documentation . . . 17
4.4.3.1 print() . . . 17

5 File Documentation **19**
5.1 Cocktailbuch.cpp File Reference . . . 19
5.1.1 Detailed Description . . . 19
5.2 Cocktailbuch.hpp File Reference . . . 19
5.2.1 Detailed Description . . . 20
5.3 Cocktailbuch.hpp . . . 20
5.4 main.cpp File Reference . . . 21
5.4.1 Detailed Description . . . 21
5.5 Rezept.cpp File Reference . . . 21
5.5.1 Detailed Description . . . 21
5.5.2 Function Documentation . . . 22
5.5.2.1 operator<<() . . . 22
5.6 Rezept.hpp File Reference . . . 22
5.6.1 Detailed Description . . . 22
5.7 Rezept.hpp . . . 23

Index **25**

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Cocktailbook 9
Drink 12
Cocktail 7
Tea 15

Chapter 2

Class Index

2.1 Class List

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

Cocktail
: Speichert Zutaten in einem Vector 7

Cocktailbook
: Diese Klasse ist eine Containerklasse fuer solche Objekte, die die Basisklasse 'class Drink' haben 9

Drink
: Basisklasse fuer Getraenke, erweiterbar mit spezifischen Typen durch Vererbung 12

Tea
: Subklasse von class Drink, mit der zusaetzlichen Membervariable string origin 15

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Cocktailbuch.cpp
: Enthaltet manche Funktiondefinitionen die zu der Containerklasse gehoeren 19

Cocktailbuch.hpp
: Enthaltet die Containerklasse 19

main.cpp
: Demonstration der entwickelten Funktionalitaeten in Rezept.hpp und Cocktailbuch.hpp 21

Rezept.cpp
: Enthaltet manche Funktiondefinitionen, die zu den entwickelten Datentypen gehören 21

Rezept.hpp
: Enthaltet die entwickelten Datentypen: class Drink, Tea, Cocktail 22

Chapter 4

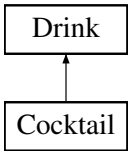
Class Documentation

4.1 Cocktail Class Reference

Speichert Zutaten in einem Vector.

`#include <Rezept.hpp>`

Inheritance diagram for Cocktail:



Public Member Functions

- Cocktail (string n, string f, string t, bool x, const vector< string > &v, string p)
- Cocktail (string filename)
- void **printIngredients** ()

  Eine solche print() Funktion fuer Cocktail Datentypen, die nur die Zutaten des Cocktails auf die Standardausgabe darstellt.
- void print ()

  Gibt alle Membervariablen auf die Standardausgabe aus.

Friends

- ostream & **operator**<< (ostream &out, const Cocktail &c)

  Ueberladener kaskadierbarer Ausgabeoperator fuer Cocktail Datentypen.

Additional Inherited Members

4.1.1 Detailed Description

Speichert Zutaten in einem Vector.

Bei Cocktails wird typisch die Methode der Herstellung ein bisschen langer als bei anderen Typen, deshalb ist der Konstruktor Drink::Drink(string filename) so entwickelt, dass string prep zeichenweise eingelest wird bis der Ende der gegebenen .txt Datei.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Cocktail() [1/2]

```
Cocktail::Cocktail (
            string n,
            string f,
            string t,
            bool x,
            const vector< string > & v,
            string p )  [inline]
```

Eigentlich die Menge der Zutaten, gespeichert in einem Vektor.

4.1.2.2 Cocktail() [2/2]

```
Cocktail::Cocktail (
            string filename )
```

Der empfehlenswerte Konstruktor, indem die Zutaten mit getline() Funktion nacheinander eingelesen werden, bis "Preparation:" Zeile erreicht ist.

Parameters

filename	Die gegebene .txt Datei.

4.1.3 Member Function Documentation

4.1.3.1 print()

```
void Cocktail::print ( )  [inline], [virtual]
```

Gibt alle Membervariablen auf die Standardausgabe aus.

Reimplemented from Drink.

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

- Rezept.hpp
- Rezept.cpp

4.2 Cocktailbook Class Reference

Diese Klasse ist eine Containerklasse fuer solche Objekte, die die Basisklasse 'class Drink' haben.

`#include <Cocktailbuch.hpp>`

Public Member Functions

- Cocktailbook (unsigned u=0)

  Allokiert Speicherplatz fuer Membervariable 'arr'.
- **Cocktailbook** (const Cocktailbook &other)
- void list ()
- void list (unsigned endPos)

  Vom Anfang bis ein gegebenen Position alle Elemente von 'arr' durch std::cout auf dem Bildschirm schriebet.
- Drink atCopy (unsigned pos)
- void pushback (Drink *d)

  Memberfunktion, womit man die heterogene Sammlung mit einem zusaetzlichen Element erweitern kann.
- Cocktailbook & operator+ (Drink *d)

  Kaskadierbarer Operator, der ein Element zur heterogener Sammlung zufugt.
- Cocktailbook & operator- (Drink *d)
- Drink ** operator[ ] (unsigned pos)

  Unaerer Operator, der die Addresse an dem gegebenen Position zurueckgibt.
- ~**Cocktailbook** ()

  Destruktor, der die dynamische heterogene Sammlung freigibt.

4.2.1 Detailed Description

Diese Klasse ist eine Containerklasse fuer solche Objekte, die die Basisklasse 'class Drink' haben.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Cocktailbook()

```
Cocktailbook::Cocktailbook (
            unsigned u = 0 )  [inline]
```

Allokiert Speicherplatz fuer Membervariable 'arr'.

Aktuelle Groesse von 'Drink** arr'.

Wenn u > 0, dann allokiert dieser Konstruktor leere Stellen, die mit Hilfe des [] Operators einzufuellen sind.

See also

operator[]

Parameters

u	Erwuenschte (mininale) Groesse von 'arr'.

4.2.3 Member Function Documentation

4.2.3.1 atCopy()

```
Drink Cocktailbook::atCopy (
            unsigned pos )
```

Gibt eine Kopie von demjenigen Element zurueck, welches an der angegebenen Stelle gespeichert ist.

Parameters

pos	die angegebene Stelle

Returns

Kopie des Getraenkes, der auf der gefragten Stelle gespeichert ist.

4.2.3.2 list() [1/2]

```
void Cocktailbook::list ( )
```

Vom Anfang bis Ende alle Elemente von 'arr' durch std::cout auf dem Bildschirm schriebet.

Weitere erklaerung in Cocktailbuch.cpp.

Listet alle Elemenente, die in einem Cocktailbuch sind, auf die Standardausgabe.

Da die Containerklasse eigentlich eine heterogene Sammlung (mit weiteren Funktionalitaeten) ist, ist es moglich, fur jedes Element waehrend Durchiterierung eines Buches die print() Funktion des aktuellen Objektes anzurufen.

4.2.3.3 list() [2/2]

```
void Cocktailbook::list (
            unsigned endPos )
```

Vom Anfang bis ein gegebenen Position alle Elemente von 'arr' durch std::cout auf dem Bildschirm schriebet.

Nutzhaft zum Beispiel, wenn der Benutzer mehr Speicherplatz fuer 'arr' als er anwendet allokiert hat.

Parameters

endPos	Die gegebene Endposition.

4.2.3.4 operator+()

```
Cocktailbook & Cocktailbook::operator+ (
            Drink * d )  [inline]
```

Kaskadierbarer Operator, der ein Element zur heterogener Sammlung zufugt.

Parameters

d	Pointer auf das Element, das zur Sammlung zugefuegt wird.

Returns

Referenz auf das Buch selbst, sodass der Operation kaskadierbar sei.

4.2.3.5 operator-()

```
Cocktailbook & Cocktailbook::operator- (
            Drink * d )
```

Sucht nach eine Addresse in der heterogene Sammlung, die mit Eingangsparameter 'Drink∗ d' uebereinstimmt, und entfernt dieses Element von der Sammlung, wenn es ein solches Element gibt.

Parameters

d	Addresse eines Getraenkes

Returns

Referenz auf das Buch selbst, sodass der Operation kaskadierbar sei.

4.2.3.6 operator[]()

```
Drink ** Cocktailbook::operator[] (
            unsigned pos )
```

Unaerer Operator, der die Addresse an dem gegebenen Position zurueckgibt.

Wenn die gegebene Position kleiner als die hoechste Position der heterogenen Sammlung ist, dann terminiert das Programm.

Attention

Rueckgabewerttyp ist "Drink**", also muss der Benutzer den Rueckgabewert DEREFERENZIEREN!

Parameters

pos	Die abgefragte Position.

Returns

Pointer auf demjenigen Pointer, der auf der gegebenen Stelle gespeichert wurde.

4.2.3.7 pushback()

```
void Cocktailbook::pushback (
            Drink * d )
```

Memberfunktion, womit man die heterogene Sammlung mit einem zusaetzlichen Element erweitern kann.

Nach der Anruf dieser Funktion wird die Groeße von Membervariable 'arr' inkrementiert. Wenn die Membervariable 'size' und daimt die Groesse von 'arr' uberflussig waere (d.h. groesser als 'const static unsigned Cocktailbook↩ ::max_size'), dann terminiert dass Programm.

Parameters

d	das Element, welches am Ende der Liste zugefuegt wird

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

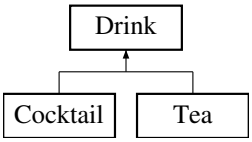
- Cocktailbuch.hpp
- Cocktailbuch.cpp

4.3 Drink Class Reference

Basisklasse fuer Getraenke, erweiterbar mit spezifischen Typen durch Vererbung.

```
#include <Rezept.hpp>
```

Inheritance diagram for Drink:



Public Member Functions

- Drink ()
- Drink (string s, string f, string t, bool x, string p)
- Drink (string filename)
- string **getName** ()
- string **getFlavour** ()
- string **getTexture** ()
- bool **servedHot** ()
- virtual void print ()

Protected Attributes

- string **name**
- string flavour
- string texture
- bool served_hot
- string prep

Friends

- bool **operator==** (const Drink &d1, const Drink &d2)
- ostream & operator<< (ostream &out, const Drink &d)

4.3.1 Detailed Description

Basisklasse fuer Getraenke, erweiterbar mit spezifischen Typen durch Vererbung.

Diese Klasse speichert gerade eine Menge von Informationen (Name, Geschmack, Substanz, ob das Getraenke heiss oder kalt zu andienen ist, Methode der Preparation), weil sie als eine allgemeine Klasse fuer Getraenke behandelt ist.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Drink() [1/3]

```
Drink::Drink ( )  [inline]
```

Methode der Herstellung in einigen Saetzen Default Konstruktor fuer class Drink.

Um das Programm angenehm mit zusaetzlichen Klassen erweiterbar zu machen, ist es noetig fuer die Basisklasse, ein parameterloser Defaultkonstruktor zu haben. Diejenige Subklasse-Konstruktorren, die die Informationen von einem File einlesen, werden mittels dieses Konstruktors das Basis-Subobjekt initialisieren.

See also

Tea::Tea(string filename)

Cocktail::Cocktail(string filename)

4.3.2.2 Drink() [2/3]

```
Drink::Drink (
            string s,
            string f,
            string t,
            bool x,
            string p )  [inline]
```

Konstruktor, indem alle einzige Parameter koennen angegeben werden

4.3.2.3 Drink() [3/3]

```
Drink::Drink (
            string filename )
```

Diese Konstruktor ist empfehlenswert anzuwenden.

Parameters

filename	.txt File, von welchem alle Membervariablen einlesbar sind.

4.3.3 Member Function Documentation

4.3.3.1 print()

```
virtual void Drink::print ( )  [inline], [virtual]
```

Schreibt den Namen, den Geschmack, den Substanz, und die Methode der Preparation auf die Standardausgabe aus.

Reimplemented in Tea, and Cocktail.

4.3.4 Friends And Related Function Documentation

4.3.4.1 operator<<

```
ostream & operator<< (
            ostream & out,
            const Drink & d )  [friend]
```

Ahnlich wie Drink::print(), aber kaskadierbar und wirkt fur allgemeine Streams.

4.3.5 Member Data Documentation

4.3.5.1 flavour

```
string Drink::flavour  [protected]
```

Name des Getraenkes

4.3.5.2 prep

```
string Drink::prep  [protected]
```

Eigenschaft des Getraenkes, die sagt uns, ob es kalt oder heiss am besten zu geniessen ist

4.3.5.3 served_hot

```
bool Drink::served_hot  [protected]
```

Substanz des Getraenkes

4.3.5.4 texture

```
string Drink::texture  [protected]
```

Geschmack des Getraenkes

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

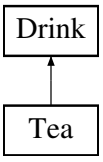
- Rezept.hpp
- Rezept.cpp

4.4 Tea Class Reference

Subklasse von class Drink, mit der zusaetzlichen Membervariable string origin.

```
#include <Rezept.hpp>
```

Inheritance diagram for Tea:



Public Member Functions

- Tea (string s, string f, string t, bool x, string o, string p)
- Tea (string filename)

  Konstruktion von einer .txt Datei, erweitern mit dem Einlesen von 'string origin'.
- void print ()

  Drueckt sowohl die Eigenschaften als auch den Ursprung des Tees.
- void **printOrigin** ()

  Ausgabefunktion fuer den Ursprung des Tees.

Friends

- ostream & **operator**<< (ostream &out, const Tea &t)

  Ausgabeoperator, erweitert mit string origin Membervariable.

Additional Inherited Members

4.4.1 Detailed Description

Subklasse von class Drink, mit der zusaetzlichen Membervariable string origin.

Diese Subklasse, wie alle geplante Subklassen, werden am einfachsten von einem vorher geschriebenen File konstruiert werden. Eine Neuigkeit ist aber, dass die Datei muss schon auch enthalten, woher die Teeart kommt.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 Tea() [1/2]

```
Tea::Tea (
            string s,
            string f,
            string t,
            bool x,
            string o,
            string p )  [inline]
```

Woher das Tee kommt

4.4.2.2 Tea() [2/2]

```
Tea::Tea (
            string filename )
```

Konstruktion von einer .txt Datei, erweitern mit dem Einlesen von 'string origin'.

Initialisiert das Subobjekt von class Drink mit Hilfe von Drink::Drink() parameterloser Konstruktor.

Funktioniert genau wie Drink::Drink(string filename), aber muss noch string origin Membervariable einlesen. Diese Information kann höchstens eine Zeile lang sein in der .txt Datei.

Parameters

filename	Name der .txt Datei woher das Tee eingelesen wird.

4.4.3 Member Function Documentation

4.4.3.1 print()

```
void Tea::print ( )  [inline], [virtual]
```

Drueckt sowohl die Eigenschaften als auch den Ursprung des Tees.

Reimplemented from Drink.

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

- Rezept.hpp
- Rezept.cpp

Chapter 5

File Documentation

5.1 Cocktailbuch.cpp File Reference

Enthaltet manche Funktiondefinitionen die zu der Containerklasse gehoeren.

```
#include "Cocktailbuch.hpp"
```

5.1.1 Detailed Description

Enthaltet manche Funktiondefinitionen die zu der Containerklasse gehoeren.

Author

Karsai Janos

Date

May 2022

5.2 Cocktailbuch.hpp File Reference

Enthaltet die Containerklasse.

```
#include "Rezept.hpp"
#include <cassert>
```

Classes

- class Cocktailbook

  Diese Klasse ist eine Containerklasse fuer solche Objekte, die die Basisklasse 'class Drink' haben.

5.2.1 Detailed Description

Enthaltet die Containerklasse.

Author

Karsai Janos

Date

May 2022

5.3 Cocktailbuch.hpp

Go to the documentation of this file.

```
1 /*****************************************************************/
8 #ifndef COCKTAILBUCH_H
9 #define COCKTAILBUCH_H
10 #include "Rezept.hpp"
11 #include <cassert>
12
18 class Cocktailbook {
19
20     const static unsigned max_size;
22     Drink** arr;
23     unsigned size;
24 public:
32     Cocktailbook(unsigned u = 0) : size(u)
33     {
34         if (size == 0) arr = nullptr;
35         else
36         {
37             assert(size <= max_size);
38             arr = new Drink*[size];
39         }
40     }
41     Cocktailbook(const Cocktailbook& other)
42     {
43         size = other.size;
44         if (size == 0) arr = nullptr;
45         else
46         {
47             arr = new Drink * [size];
48             for (int i = 0; i < size; i++)
49             {
50                 arr[i] = other.arr[i];
51             }
52         }
53     }
60     void list();
61
70     void list(unsigned endPos); // vom Anfang bis ein gegebenen Position
71
72
73     Drink atCopy(unsigned pos);
74
83     void pushback(Drink* d);
84
85
86     // ueberladene Operatoren
87
94     Cocktailbook& operator+(Drink* d)
95     {
96         pushback(d);
97         return *this;
98     }
99
100     Cocktailbook& operator-(Drink* d);
101
110     Drink** operator[](unsigned pos);
111
116     ~Cocktailbook() { delete[] arr; }
117
118 };
119
120
121 #endif
```

5.4 main.cpp File Reference

Demonstration der entwickelten Funktionalitaeten in Rezept.hpp und Cocktailbuch.hpp.

```
#include "Rezept.hpp"
#include "Cocktailbuch.hpp"
```

Functions

- int **main** ()

5.4.1 Detailed Description

Demonstration der entwickelten Funktionalitaeten in Rezept.hpp und Cocktailbuch.hpp.

Dokumentation der main() Funktion befindet sich in der Benutzerdokumentation.

Author

Karsai Janos

Date

May 2022

5.5 Rezept.cpp File Reference

Enthaltet manche Funktiondefinitionen, die zu den entwickelten Datentypen gehören.

```
#include "Rezept.hpp"
```

Functions

- bool **operator==** (const Drink &d1, const Drink &d2)
- ostream & operator<< (ostream &out, const Drink &d)
- ostream & **operator**<< (ostream &out, const Tea &t)
- ostream & **operator**<< (ostream &out, const Cocktail &c)

5.5.1 Detailed Description

Enthaltet manche Funktiondefinitionen, die zu den entwickelten Datentypen gehören.

Author

User

Date

May 2022

5.5.2 Function Documentation

5.5.2.1 operator<<()

```
ostream & operator<< (
            ostream & out,
            const Drink & d )
```

Ahnlich wie Drink::print(), aber kaskadierbar und wirkt fur allgemeine Streams.

5.6 Rezept.hpp File Reference

Enthaltet die entwickelten Datentypen: class Drink, Tea, Cocktail.

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
#include <vector>
```

Classes

- class Drink

  Basisklasse fuer Getraenke, erweiterbar mit spezifischen Typen durch Vererbung.
- class Tea

  Subklasse von class Drink, mit der zusaetzlichen Membervariable string origin.
- class Cocktail

  Speichert Zutaten in einem Vector.

5.6.1 Detailed Description

Enthaltet die entwickelten Datentypen: class Drink, Tea, Cocktail.

Author

Karsai Janos

Date

May 2022

5.7 Rezept.hpp

Go to the documentation of this file.

```
1 #define _CRT_SECURE_NO_WARNINGS
2 /*****************************************************************/
9 #ifndef REZEPT_H
10
11 #define REZEPT_H
12 #include <iostream>
13 #include <fstream>
14 #include <string>
15 #include <cstring>
16 #include <vector>
17
18 using namespace std;
19
29 class Drink {
30 protected:
31
32     string name;
33     string flavour;
34     string texture;
35     bool served_hot;
36     string prep;
39 public:
50     Drink() : name(), flavour(), texture(), served_hot(false), prep() {}
51
53     Drink(string s, string f, string t, bool x, string p) : name(s), flavour(f), texture(t),
       served_hot(x), prep(p) {}
54
59     Drink(string filename);
60
61     virtual ~Drink() {}
62
63
64     string getName() { return name; }
65     string getFlavour() { return flavour; }
66     string getTexture() { return texture; }
67     bool servedHot() { return served_hot; }
68
69
71     virtual void print()
72     {
73         cout « "Drink name: " « name « "\nFlavour: " « flavour
74             « "\nTexture: " « texture « endl;
75         if (served_hot) cout « "Best served hot" « endl;
76         else cout « "Best served cold" « endl;
77         cout « "Preparation: " « prep « endl;
78     }
79     friend bool operator==(const Drink& d1, const Drink& d2);
80
82     friend ostream& operator«(ostream& out, const Drink& d);
83 };
84
92 class Tea : public Drink{
93 private:
94
95     string origin;
96 public:
97
98     Tea(string s, string f, string t, bool x, string o, string p) : Drink(s, f, t, x, p), origin(o) {}
102     Tea(string filename);
103     ~Tea() {}
104
109     void print()
110     {
111         Drink::print(); printOrigin();
112     }
116     void printOrigin() { cout « "Origin: " « origin « endl; }
120     friend ostream& operator«(ostream& out, const Tea& t);
121 };
122
131 class Cocktail : public Drink
132 {
133 private:
134     vector<string> ingredients;
135 public:
136     Cocktail(string n, string f, string t, bool x, const vector<string>& v, string p) :
137         Drink(n, f, t, x, p), ingredients(v) {}
138
144     Cocktail(string filename);
149     void printIngredients()
150     {
151         cout « "Ingredients: ";
152         for (int i = 0; i < ingredients.size(); i++)
```

```
153         {
154 
155             cout « " > " « ingredients[i] « endl;
156         }
157     }
162     void print()
163     {
164         Drink::print();
165         printIngredients();
166     }
170     friend ostream& operator«(ostream& out, const Cocktail& c);
171 };
172 
173 
174 
175 #endif
```

Index

atCopy
    Cocktailbook, 10

Cocktail, 7
    Cocktail, 8
    print, 8
Cocktailbook, 9
    atCopy, 10
    Cocktailbook, 9
    list, 10
    operator+, 11
    operator-, 11
    operator[], 11
    pushback, 12
Cocktailbuch.cpp, 19
Cocktailbuch.hpp, 19

Drink, 12
    Drink, 13, 14
    flavour, 15
    operator<<, 14
    prep, 15
    print, 14
    served_hot, 15
    texture, 15

flavour
    Drink, 15

list
    Cocktailbook, 10

main.cpp, 21

operator<<
    Drink, 14
    Rezept.cpp, 22
operator+
    Cocktailbook, 11
operator-
    Cocktailbook, 11
operator[]
    Cocktailbook, 11

prep
    Drink, 15
print
    Cocktail, 8
    Drink, 14
    Tea, 17
pushback
    Cocktailbook, 12

Rezept.cpp, 21
    operator<<, 22
Rezept.hpp, 22

served_hot
    Drink, 15

Tea, 15
    print, 17
    Tea, 16
texture
    Drink, 15