SOFTWARE REQUIREMENTS

# SPECIFICATION

fur Codegenerator GetOptGen¨

Programmentwurf C/C++ 2022

# TIK/TIM/TIS/TIT21

Version 1.1

Version 1.1 approved

## Erstellt durch Th. Staudacher und A. Maus

## Inhaltsverzeichnis

1. Einfuhrung¨ 4
   1. Gultigkeit . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .¨ 4
   2. Gruppengroße . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .¨ 4
   3. Abgabe der Prufungsleistung¨ . . . . . . . . . . . . . . . . . . . . . . . . . 4
   4. Aufgabenstellung . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
2. Allgemeine Beschreibung 6
   1. Funktionale Anforderungen . . . . . . . . . . . . . . . . . . . . . . . . . . 6
      1. Genereller Ablauf . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
      2. XML-Format . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
      3. Optionsparameter . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
      4. Optionen . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
      5. Codegenerierung . . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
      6. Allgemeine Dateigenerierungs Regeln . . . . . . . . . . . . . . . . . 13
      7. Implementierung in eine Zielapplikation . . . . . . . . . . . . . . . 13
      8. Demonstration durch ein Beispielprojekt . . . . . . . . . . . . . . . 15
   2. Nichtfunktionale Anforderungen . . . . . . . . . . . . . . . . . . . . . . . 16
      1. Tools und Build Umgebungen . . . . . . . . . . . . . . . . . . . . . 16
   3. Optionale Anforderungen . . . . . . . . . . . . . . . . . . . . . . . . . . . 19
      1. Tools und Build Umgebungen . . . . . . . . . . . . . . . . . . . . . 20
   4. Programmierung . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
   5. Programmbeispiele . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
      1. Text Formatierung unter C++ . . . . . . . . . . . . . . . . . . . . 21
      2. Text Formatierung unter C++ analog printf . . . . . . . . . . . . . 21
      3. String in Teilstrings zerlegen . . . . . . . . . . . . . . . . . . . . . 22
      4. Text in Zahlen wandeln . . . . . . . . . . . . . . . . . . . . . . . . 22
      5. Optionale Shortopts . . . . . . . . . . . . . . . . . . . . . . . . . . 23
      6. Text in Linien fester Breite wandeln . . . . . . . . . . . . . . . . . 24
      7. Unittest . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
      8. SAX mit Xerces-C . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
      9. XML, utf-16, utf-8 (ASCII) . . . . . . . . . . . . . . . . . . . . . . 27
      10. Muster XML . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28

## Revision History

Version 0.9: Diese Version ist noch nicht final. Sie ist noch nicht mit den Kursen abgeglichen und einzelne Requirements konnen sich noch¨ andern, bzw hinzugef¨ ugt¨ werden.

Version 1.0a: Diese Version ist die Alphaversion der ersten verbindlichen Version. Sie ist noch nicht mit den Kursen abgeglichen und einzelne Requirements konnen¨ sich noch andern.¨ Anderungen zu Version 0.9 gab es viele. Gr¨ oßere¨ Anderungen¨ in Kapitel 1.3, 1.4, 2.1.2, 2.1.3, 2.1.4, 2.1.5. Folgende Kapitel sind komplett neu: 2.1.7, Identifikation der Gruppe.

Version 1.1: Diese Version ist final. Außer Xerces-C konne auch selbstgeschriebene¨ Parser eingesetzt werden. Anpassungen im Kapitel optionale Requirements. Mehr Programmbeispiele.

## 1 Einfuhrung¨

### 1.1 Gultigkeit¨

Diese Version ist Version 1.1 . Sie ist die Alphaversion der ersten verbindlichen Version. Sie ist noch nicht mit den Kursen abgeglichen und einzelne Requirements konnen sich¨ noch andern.¨

### 1.2 Gruppengroße¨

Die empfohlene Gruppengroße f¨ ur die Bearbeitung dieses Projekts sind 4 Teilnehmer.¨ Die maximale Gruppengroße liegt bei 5 Teilnehmern (der Dozent kann Ausnahmen fest-¨ legen). Der Prufungsumfang¨ andert sich nicht durch kleinere Gruppengr¨ oßen.¨

### 1.3 Abgabe der Prufungsleistung¨

Jeder Student und Studentin gibt eigenstandig sein Programmentwurf bei Moodle ab,¨ spatestens bis zum vorgesehenen Abgabezeitpunkt der unter Moodle vermerkt ist.¨

Anmerkung: [Studien- und Prufungsordnung¨](https://www.ravensburg.dhbw.de/fileadmin/user_upload/Dokumente/Amtliche_Bekanntmachungen/2021/34_2021_Bekanntmachung_StuPrO_Technik_inkl._Fuenfte_AEnderungssatzung.pdf) [fur die Bachelorstudien¨](https://www.ravensburg.dhbw.de/fileadmin/user_upload/Dokumente/Amtliche_Bekanntmachungen/2021/34_2021_Bekanntmachung_StuPrO_Technik_inkl._Fuenfte_AEnderungssatzung.pdf) [gange im Studien¨](https://www.ravensburg.dhbw.de/fileadmin/user_upload/Dokumente/Amtliche_Bekanntmachungen/2021/34_2021_Bekanntmachung_StuPrO_Technik_inkl._Fuenfte_AEnderungssatzung.pdf) [bereich Technik der Dualen Hochschule Baden-Wurttemberg (DHBW) (Studien- un¨](https://www.ravensburg.dhbw.de/fileadmin/user_upload/Dokumente/Amtliche_Bekanntmachungen/2021/34_2021_Bekanntmachung_StuPrO_Technik_inkl._Fuenfte_AEnderungssatzung.pdf) [d Prufungsordnung DHBW Technik – StuPrO DHBW Technik¨](https://www.ravensburg.dhbw.de/fileadmin/user_upload/Dokumente/Amtliche_Bekanntmachungen/2021/34_2021_Bekanntmachung_StuPrO_Technik_inkl._Fuenfte_AEnderungssatzung.pdf) [)](https://www.ravensburg.dhbw.de/fileadmin/user_upload/Dokumente/Amtliche_Bekanntmachungen/2021/34_2021_Bekanntmachung_StuPrO_Technik_inkl._Fuenfte_AEnderungssatzung.pdf) § 11 Versaumnis, R¨ ucktritt,¨

Tauschung, Ordnungsverstoß / 1¨

### 1.4 Aufgabenstellung

Erstellen Sie einen Codegenerator der eine Klasse zum Auswerten von Ubergabeparametern¨ erzeugt. Die Optionen sollen hierbei durch eine XML-Datei (XML-Konfigurationsdatei) an das Programm ubergeben werden. Des weiteren sind in der XML-Datei auch An-¨ gaben zum Autor, Beispiele, Beschreibungen und Ablageorte hinterlegt. Je nach den Vorgaben aus der XML-Datei soll die mit dem Codegenerator erzeugte Klasse entweder direkt instanziierbar oder abstrakt sein. Die Formatierung der erzeugten Header- und Source Datei ist nicht Umfang des Programmentwurfs, diese kann unter Zuhilfenahme des Kommandozeilenprogramms astyle erfolgen. Der Aufruf von astyle findet ebenfalls nicht im Programmentwurf statt. Die Main Funktion ist ebenfalls nicht Bestandteil der Generierung. Um das Auswerten der Optionen durchzufuhren muss eine Instanz der ge-¨ nerierten Klasse in der Applikation erzeugt werden und eine Funktion zum Auslesen der Optionen aufgerufen werden. Falls sich ein abstraktes Element in der generierten Klasse befindet muss diese dann nochmals in der Zielapplikation abgeleitet werden und die abstrakte Methode implementiert werden.

Nahere Information zur Umsetzung sind in den Requirements der folgenden Kapitel¨ detailliert aufgefuhrt.¨

## 2 Allgemeine Beschreibung

### 2.1 Funktionale Anforderungen

Funktionale Anforderungen beschreiben was das Programm ausfuhren soll. Im Gegensatz¨ zu nichtfunktionalen Anforderungen sind sie direkt in der Software zu finden, d.h. es wird kein Tooling beschrieben sondern was im Programmablauf zu beachten ist.

#### 2.1.1 Genereller Ablauf

Dem Codegenerator soll per Kommandozeilenparameter eine XML-Konfigurationsdatei ubergeben werden. Der Codegenerator soll diese XML-Konfigurationsdatei einlesen und¨ basierend auf den darin enthaltenen Einstellungen Code-Dateien generieren. Die auf diese Weise generierten Dateien sollen das Grundgerust f¨ ur eine neue Applikation bilden.¨

#### 2.1.2 XML-Format

Das Format der XML-Konfigurationsdatei ist wie folgt vorgegeben (siehe Example.xml). Zur Reduktion der Komplexitat wird es sich anbieten, Sonderzeichen im XML-Text¨

bereits fertig fur C/C++ zu¨ escapen“. ”

GetOptSetup

Das Attribut SignPerLine stellt die Anzahl der Buchstaben pro Zeile ein. Typisch fur¨ die Kommandozeile ist der Defaultwert 79.

Author

Die Attribute Name, Phone und Mail beinhalten Informationen uber den Autor, die¨ beim Hilfetext mit ausgegeben werden sollen.

HeaderFileName

HeaderFileName hat keine Attribute. Zwischen Start- und End-Tag steht der geplante Dateiname der anzulegenden Header-Datei.

SourceFileName

SourceFileName hat keine Attribute. Zwischen Start- und End-Tag steht der geplante Dateiname der anzulegenden Quellcode-Datei.

NameSpace

NameSpace hat keine Attribute. Zwischen Start- und End-Tag steht der geplante Name des anzulegenden Namespace in der sich die generierte Klasse befindet.

ClassName

ClassName hat keine Attribute. Zwischen Start- und End-Tag steht der geplante Name der anzulegenden Klasse.

OverAllDescription

OverAllDescription hat keine Attribute. Zwischen Start- und End-Tag stehen 1 oder mehrere Block-Tags. Block hat keine Attribute. Zwischen Start- und End-Tag steht ein Textblock, der das Programm beschreibt.

SampleUsage

SampleUsage hat keine Attribute. Zwischen Start- und End-Tag stehen 1 oder mehrere Sample-Tags. Sample hat keine Attribute. Zwischen Start- und End-Tag steht ein moglicher Programmaufruf.¨

Options

Options hat keine Attribute. Zwischen Start- und End-Tag stehen 1 oder mehrere OptionTags. Option hat die Attribute Ref (Optional), ShortOpt, LongOpt (minimum 1 von beiden muss belegt sein), Exclusion, Interface, ConvertTo, DefaultValue, ConnectToInternalMethod, ConnectToExternalMethod, HasArguments und Description.

Ref ist ein Integerwert und dient dazu die einzelnen Attribute referenzieren zu konnen, zum Beispiel wenn sie sich gegenseitig ausschließen sollen.¨

ShortOpt definiert die Parameter die nur aus einem einzelnen Buchstaben bestehen konnen.¨

LongOpt definiert die Parameter die aus ganzen Worten bestehen konnen.¨

Interface ist die Schnittstelle wo nach Finden der Option der Wert hinterlegt wird

Exclusion listet die Referenzen der Optionen mit der diese Option nicht gemeinsam aufgerufen werden darf

ConvertTo gibt eine Umwandlung vor da ein Command-Line Parameter initial immer Text ist.

DefaultValue gibt bei Optionalen Parametern den Default-Wert vor, der angenommen werden soll, wenn der Parameter nicht gesetzt wird.

ConnectToInternalMethod definiert die Methode die bei der Generierung dieser Klasse angelegt werden muss und beim Auswerten dieses Optionsparameters aufgerufen wird (z.B. den Hilfetext).

ConnectToExternalMethod definiert die Methode die beim Auswerten dieses Optionsparameters aufgerufen wird. Diese ist abstrakt und die Klasse muss einmal abgeleitet werden.

HasArguments kann die Werte optional und required annehmen. Wenn die Option keine Argumente nutzt kann HasArguments weggelasssen werden.

Description ist der eigentliche Hilfetext der zu der Option angezeigt werden soll.

#### 2.1.3 Optionsparameter

Darstellung im Hilfe Menu¨

Um kein aufwendiges sortieren von Hand in der XML Konfiguration durchfuhren zu¨ mussen soll der Parser die Optionen nach Alphabet sortieren. Prior sind die Buchstaben¨ der Kurzoptionen, dann deren der Langoptionen (ohne fuhrende -/– zu beachten).¨

{ReqFunc1} Nach Parsen aller Optionen sollen die Optionen nach Alphabet aufsteigend sortiert werden [A-Z] todo: Elias

Es soll eine Hilfetextausgabe realisiert werden. Diese soll als Methode in der Klasse hinterlegt werden. Der Hilfetext soll aus den Optionen generiert werden. Dabei ist die Kurz-, die Langoption, die Beschreibung sowie die Angabe zum Autor und Benutzung unterzubringen. Die Textbreite soll durch die XML-Einstellungen vorgegeben werden.

|  |  |
| --- | --- |
| {ReqFunc2} | Es ist eine virtuelle protected Methode printHelp zu erstellen die den Hilfetext auf der Konsole ausgibt. todo: Yannic |
| {ReqFunc3} | Der Hilfetext soll die Inhalte Blockformatieren. todo:Yannic |
| {ReqFunc4} | Der Hilfetext ist so einzufassen, dass er die Angabe aus dem Attribute SignPerLine nicht uberschreitet (Defaultwert 79 Zeichen/Linie).¨todo:Yannic |
| {ReqFunc5} | Der Hilfetext soll am Ende eine Zeile mit den Angaben aus dem XML Tag Author enthalten. Todo:Yannic |
| {ReqFunc6} | Der Hilfetext soll die Beschreibungen aus dem XML Tag OverAllDescription enthalten. |
| {ReqFunc7} | Der Hilfetext soll die Beispielbenutzung aus dem XML Tag SampleUsage enthalten. |

#### Todo steve

#### 1.1.4 Optionen

Generelle nicht konfigurierbare Funktionalit¨at

Optionen die nicht geparst werden konnen, sollen mit einer Fehlermeldung ausgegeben¨ werden und um dann das Programm zu beenden. Jedoch kann durch Ableitung der generierten Klasse die ausfuhrende Methode¨ uberschrieben werden. Auf der abgeleiteten¨ Methode kann ein Dateiname oder eine komplexe Auswertung durchgefuhrt werden die¨ in der Konfiguration nicht konfigurierbar ist.

|  |  |
| --- | --- |
| {ReqFunc8} | Fur Option die nicht vorhanden sind soll eine Routine implementiert werden mit¨ dem Methodenrumpf **virtual void unknownOption(const string& unknown option)** die die Fehlerabarbeitung fur diese Situation¨ ubernimmt¨ |
| {ReqFunc9} | Ist eine Option fehlerhaft ausgefuhrt (z.B. option required und nicht¨ ubergeben¨ worden) und der Methodenrumpf nicht uberschrieben, dann soll das Programm¨ eine Fehlermeldung mit Angabe welche Kombination in diesem Fall nicht erlaubt war auf stderr ausgeben werden |
| {ReqFunc10} | Wenn eine Option nicht vorhanden und der Methodenrumpf nicht uberschrieben¨ ist, dann soll das Programm abgebrochen werden |

Referenzen und Exclusion

Da nicht alle Optionsparameter sinnvoll gleichzeitig verwendet werden konnen, soll es¨ Einschrankungen geben. Diese Einschr¨ ankungen sollen durch Referenzen in den Ein-¨ schrankungen eingetragen werden k¨ onnen. Tritt nun so eine Kombination auf, dann soll¨ das Parsen der Optionen im generierten Programm beendet werden und auf dem Bildschirm eine Fehlermeldung ausgegeben werden.

|  |  |
| --- | --- |
| {ReqFunc11} | Referenzen durfen Integer Werte von 1 bis 63 aufnehmen¨ |
| {ReqFunc12} | Referenz 0 wird nicht in den Ausschluss mit ein bezogen |
| {ReqFunc13} | Referenzen werden in Exclusion komma-separiert eingetragen |
| {ReqFunc14} | Wenn eine Option in Kombination mit einer Exclusion vorkommt, dann soll das Programm eine Fehlermeldung mit Angabe welche Kombination in diesem Fall nicht erlaubt war auf der stderr ausgeben |
| {ReqFunc15} | Wenn eine Option in Kombination mit einer Exclusion vorkommt, dann soll das  Programm abgebrochen werden |

Interfaces

In der generierten Klasse werden, fur die Optionen wo ein Interface in der Konfiguration¨ angegeben wurde, entsprechende Getter Methoden in der generierten Klasse angelegt. Wird diese Option spater dann gefunden so ist einzutragen dass sie aufgerufen wurde,¨ ggf. sind auch Werte nach aussen zu ubertragen.¨

|  |  |
| --- | --- |
| {ReqFunc16} | Ist die Option Interface angewahlt, dann soll eine Getter Funktion in der gene-¨ rierten Datei angelegt werden damit ausgelesen werden kann ob die Option vorgekommen ist. Die Schnittstellensyntax ist bool isSet+Optionsname (bevorzugt  LongOptName, ggf. ShortOptname) |
| {ReqFunc17} | Ist die Option Interface angewahlt und es gibt einen optionalen oder required¨ Ubergabeparameter, dann soll eine Getter Funktion zum Auslesen der¨ Ubergabe¨ angelegt werden. Die Schnittstellensyntax ist T getValueOf+Optionsname (bevorzugt LongOptName, ggf. ShortOptname, T kann std:string, bool oder int sein) |
| {ReqFunc18} | Ist die Option Interface angewahlt und es gibt einen optionalen¨ Ubergabeparameter,¨ dann soll eine Getter Funktion den Defaultwert ubernehmen falls kein¨ Ubergabewert¨ bei dem Aufruf der Option ubergeben wurde¨ |
| {ReqFunc19} | Ist die Option Interface angewahlt und es gibt einen optionalen oder required¨ Ubergabeparameter, dann soll eine Getter Funktion den Wert in das Zielformat¨ wandeln. Vorgesehen sind hier Grundformat string, Wandlungsformat bool und integer |
| {ReqFunc20} | Ist keine der drei Option Interface, ConnectToInternalMethod, ConnectToExternalMethod ausgewahlt, dann soll eine Getter Funktion in der generierten Datei¨ angelegt werden damit ausgelesen werden kann ob die Option vorgekommen ist. Die Schnittstellensyntax ist bool isSet+Optionsname (bevorzugt LongOptName, ggf. ShortOptname) |

ConnectToInternalMethod zum Aufruf klasseninterner Methoden

Wird ConnectToInternalMethod mit einem Namen belegt so soll die interne Methode aufgerufen werden. Dieses kann z.B. die vordefinierte Hilfefunktion helpText sein.

|  |  |
| --- | --- |
| {ReqFunc21} | Ist ConnectToInternalMethod fur eine Option ausgew¨ ahlt, dann soll diese Methode¨ aufgerufen werden sobald die Option erkannt wird |
| {ReqFunc22} | Ist ConnectToInternalMethod und ein required Ubergabeparameter f¨ ur eine Op-¨ tion ausgewahlt, dann soll diese Methode mit dem passenden¨ Ubergabeparameter¨ aufgerufen werden sobald die Option erkannt wird |
| {ReqFunc23} | Ist ConnectToInternalMethod fur eine Option ausgew¨ ahlt und es gibt einen op-¨ tionalen Ubergabeparameter, dann soll die Methode so generiert werden, dass¨ der Defaultwert gesetzt wird falls kein Ubergabewert bei dem Aufruf der Opti-¨ on ubergeben wurde¨ |
| {ReqFunc24} | Ist ConnectToInternalMethod fur eine Option ausgew¨ ahlt und es gibt einen op-¨ tionalen oder required Ubergabeparameter und eine Formatwandlung ConvertTo¨ ist mit konfiguriert, dann soll die Ubergabe im Zielformat stattfinden. Vorgesehen¨ sind hier Grundformat string, Wandlungsformat bool und integer |

ConnectToExternalMethod zum Aufruf klassenexterner Methoden

Wird ConnectToExternalMethod mit einem Namen belegt so soll die Methode der Ableitung dieser Klasse aufgerufen werden. Dazu ist eine abstrakter Methodenrumpf zu generieren.

|  |  |
| --- | --- |
| {ReqFunc25} | Ist ConnectToExternalMethod fur eine Option ausgew¨ ahlt, dann soll ein abstrak-¨ ter Methodenrumpf unter Verwendung des Namens in ConnectToExternalMethod erstellt werden |
| {ReqFunc26} | Ist ConnectToExternalMethod fur eine Option ausgew¨ ahlt, dann soll diese aufge-¨ rufen werden sobald die Option erkannt wird |
| {ReqFunc27} | Ist ConnectToExternalMethod und ConvertTo fur eine Option ausgew¨ ahlt, dann¨ soll diese mit dem passenden Ubergabeparameter aufgerufen werden sobald die¨  Option erkannt wird |
| {ReqFunc28} | Ist ConnectToExternalMethod fur eine Option ausgew¨ ahlt und es gibt einen op-¨ tionalen Ubergabeparameter, dann soll der Defaultwert¨ ubernehmen falls kein ¨Ubergabewert bei dem Aufruf der Option¨ ubergeben wurde¨ |
| {ReqFunc29} | Ist ConnectToExternalMethod fur eine Option ausgew¨ ahlt und es gibt einen op-¨ tionalen oder required Ubergabeparameter und eine Formatwandlung ConvertTo¨ ist mitkonfiguriert, dann soll die Ubergabe im Zielformat stattfinden. Vorgesehen¨ sind hier Grundformat string, Wandlungsformat bool und integer |

HasArguments zur Steuerung fur¨ Ubergabeparameter¨

**Hinweis:** optional und required bei HasArguments bezieht sich tatsachlich auf die Argu-¨ mente der Option und nicht auf die Option selbst. Die Option selbst ist immer optional. Als Beispiel fur einen optionales Argument sei hier die Option -O f¨ ur den Optimizer bei¨ g++ genannt. Sie kann nur als -O gesetzt sein, aber auch mit dem Optimierungslevel oder der Optimierungsart zusammenstehen: z.B. -O2 oder -Os (zur Optimierung auf Große). Als Beispiel f¨ ur ein required Argument sei die Option -o f¨ ur g++ genannt. Ohne¨ die Angabe der Ausgabedatei ergibt sie wenig Sinn, allerdings ist auch hier ein Default fur die Ausgabedatei vorgesehen (a.exe), wenn die Option -o weggelassen wird.¨

{ReqFunc30} Ist in HasArguments Required fur den¨ Ubergabeparameter eingetragen und der¨ Nutzer ubergibt keinen¨ Ubergabeparameter, dann soll dann unter Ausgabe einer¨ Fehlermeldung auf stderr abgebrochen werden

ConvertTo zur Steuerung der Datenwandlung fur¨ Ubergabeparameter¨

Soll ein Ubergabeparameter in ein anderes Format gewandelt werden so ist die M¨ oglichkeit¨ gleich das Zielformat mit anzugeben. Default werden alle Texte in std::string utf-8

Ubertragen. Weitere M¨ oglichkeiten sind bool und int.¨

|  |  |
| --- | --- |
| {ReqFunc31} | Ist in ConvertTo Integer fur den¨ Ubergabeparameter eingetragen, dann soll der¨ Ubergabeparameter zur Weiterverarbeitung in das Zielformat int¨ uberf¨ uhrt wer-¨ den. |
| {ReqFunc32} | Ist in ConvertTo Bool fur den¨ Ubergabeparameter eingetragen, dann soll der¨ Ubergabeparameter zur Weiterverarbeitung in das Zielformat bool¨ uberf¨ uhrt wer-¨ den. Erlaubt sind 0, 1, true, false in der Ubergabe¨ |
| {ReqFunc33} | Ist die Wandlung mit ConvertTo auf Grund der Werteart und Bereiche die da ubergeben werden nicht m¨ oglich, dann soll unter Ausgabe einer Fehlermeldung¨ auf stderr abgebrochen werden todo ali |

#### 2.1.5 Codegenerierung

Dateinamen

Die Dateinamen der Header- und der Source Datei sollen aus der XML-Konfigurationsdatei entnommen werden. Es gibt dafur keine Vorgaben. Sind diese Tags leer so kann die je-¨ weilige Datei nicht generiert werden. Auf eine komplette Pfadextraktion wird verzichtet da unter Windows die Pfadauflosung anders geregelt ist wie unter Unix. Es handelt¨ sich hierbei nur um reine Dateinamen. Der Namen der Header- und Source-Datei ist separat aufgefuhrt, da falls dieses Applikation einmal produktiv eingesetzt wird sich die¨ Endungen der Header- und Source Datei unterscheiden wird.

UNIX-V .h/.cc

Visual Studio .hpp/.cpp Linux .h/.cpp macOS .h/.cpp

{ReqFunc34} Ist in der XML-Datei der XML Tag HeaderFileName befullt, dann soll eine Header-¨ Datei mit diesem Namen erstellt werden.

{ReqFunc35} Ist in der XML-Datei der XML Tag SourceFileName befullt, dann soll eine Source-¨ Datei mit diesem Namen erstellt werden.

Header Datei

Um die Header Datei mehrfach includieren zu konnen ist eine Mehrfacheinbindung vorzu-¨ sehen. Diese soll in die erzeugte Datei geschrieben werden dass sie sowohl unter Windows MinGW wie auch unter UNIX GNU-C Compiler funktioniert. Das Kommando #pragma once ist nicht erwunscht.¨

{ReqFunc36} Die erzeugte Header Datei soll Mehrfacheinbindung unterstutzen.¨

{ReqFunc37} Die Mehrfacheinbindung der erzeugten Header Datei soll kompatibel zum Compiler Windows MinGW wie auch unter UNIX zu GNU-C sein.

#### 2.1.6 Allgemeine Dateigenerierungs Regeln

Namespaces

Ist in der XML-Datei ein Namespace benannt, dann soll dieser mit in die Erzeugung der Zieldateien geschrieben werden.

{ReqFunc38} Ist in der XML-Datei der XML Tag NameSpace befullt, dann soll in der Header-¨ Datei dieser Namespace eingefugt werden.¨ …

{ReqFunc39} Ist in der XML-Datei der XML Tag NameSpace befullt, dann soll in der Source-¨ Datei dieser Namespace eingefugt werden.¨

#### 2.1.7 Implementierung in eine Zielapplikation

Die Main-Funktion ist nicht Bestandteil der Generierung. Um das Auswerten der Optionen durchfuhren zu k¨ onnen muss eine Instanz der generierten Klasse in der Applikation¨ erzeugt werden und eine Funktion zum Auslesen der Optionen aufgerufen werden.

Aufruf der Auswertung der Optionen

{ReqFunc40} In der Generierung soll eine Methode implementiert werden mit der die Optionen geparst werden. Die Parameter dieser Methode sind **void parseOptions(int argc, char \*\*argv)**

Anwendungsbeispiel

Interface Bei einem Interface (Attribut Interface) liegen die Daten im Objekt und konnen durch Getter Funktionen abgerufen werden.¨

|  |
| --- |
| //Fusst auf der im Anhang beigefugten¨ Muster XML=Konfiguration . #include *<*options .h*>* // . . .  //Aufruf programm ==astyle=path=/usr/bin int main( int argc , char argv ) {  GenGetCreator : : COptionParser options ; options . parseOptions ( argc , argv ) ;  i f ( option . isSetAstylePath () ) {  //std : : string . . . = option . getValueOfAstylePath () ;  }  // . . . } |

2

4

6

8

10

12

14

16

Interne Methode Bei einer internen Methode (Attribut ConnectToInternalMethod) wird die im internen Objekt befindliche Methode aufgerufen.

|  |
| --- |
| //Fusst auf der im Anhang beigefugten¨ Muster XML=Konfiguration . #include *<*options .h*>* // . . .  //Aufruf programm =h int main( int argc , char \*\*argv ) {  GenGetCreator : : COptionParser options ; options . parseOptions ( argc , argv ) ;  //Durch Aufruf von Help wird der Hilfetext ausgegeben  //und das Programm beendet ( geschieht bereits in Parse )  } |

2

4

6

8

10

12

14

Externe Methode Bei einer externen Methode (Attribut ConnectToExternalMethod) wird die im internen Objekt befindliche Methode aufgerufen.

|  |
| --- |
| //Fusst auf der im Anhang beigefugten¨ Muster XML=Konfiguration . #include *<*options .h*>* // . . .  class MyOptionParser : public GenGetCreator : : COptionParser {  public :  //Anlegen von Ctor und Destruktor protected :  virtual void Version () {  //Befu¨ llen mit dem Versionstext exit (EXIT SUCCESS) ;  }  }  //Aufruf programm =v int main( int argc , char \*\*argv ) { MyOptionParser options ; options . parseOptions ( argc , argv ) ;  //Durch Aufruf von Version wird der Versionstext ausgegeben //und das Programm beendet ( geschieht bereits in Parse )  } |

2

4

6

8

10

12

14

16

18

20

22

24

#### 2.1.8 Demonstration durch ein Beispielprojekt

Inhalte

Es soll eine Beispielapplikation erstellt werden zur der eine XML-Konfiguration erstellt wird. In der XML-Konfiguration soll die gesammte Funktionalitat vorgef¨ uhrt werden.¨ Was das Programm macht ist zweitrangig, die Testimplementierung der erzeugten Klasse steht im Vordergrund.

|  |  |
| --- | --- |
| {ReqFunc41} | In der Musterimplementierung soll eine Autorenangabe der Gruppenteilnehmer enthalten sein. |
| {ReqFunc42} | In der Autorenangabe der Gruppenteilnehmer muss keine Telefonnummer enthalten sein. |
| {ReqFunc43} | In der Autorenangabe der Gruppenteilnehmer muss mindestens eine Kontakt-Mail  Adresse enthalten sein um Ruckfrage seitens Dozent stellen zu k¨ onnen.¨ |
| {ReqFunc44} | In der Musterimplementierung soll ein Namespace enthalten sein. |
| {ReqFunc45} | In der Musterimplementierung soll im Hilfetext die Beschreibung des Programms (OverAllDescription) enthalten sein. |
| {ReqFunc46} | In der Musterimplementierung soll im Hilfetext die Musterbenutzung (Sample Usage) enthalten sein. Todo steve tot he eve |
| {ReqFunc47} | Die erstellte Klasse soll eine beliebige Zahl von Argumenten parsen konnen die nur¨ von getopt(...) und getopt long(...) limitiert sind. |
| {ReqFunc48} | Die erzeugte Parser Klasse soll Character vom Type ASCII verarbeiten konnen.¨ |
| {ReqFunc49} | Die generierte Klasse soll Ihren Speicher selbstandig verwalten. Nach Entfernen¨ der Instanz soll der ganze Speicher wieder freigegeben werden. |
| {ReqFunc50} | Das Generierungsprogramm soll den Speicher selbstandig verwalten und auch wie-¨ der freigeben. |
| {ReqFunc51} | In der Musterimplementierung soll die Option Hilfe mit dem ShortOpt -h und LongOpt –help enthalten und zur internen Methode printHelp (ConnectToInternalMethod) verbunden werden. |
| {ReqFunc52} | In der Musterimplementierung soll die Option Version mit dem ShortOpt -v und LongOpt –version enthalten (Inhalte der Version bitte selbst bestimmen) und zu einer externen Routine verbunden werden (Interface). |
| {ReqFunc53} | In der Musterimplementierung soll mindestens ein Ausschluss (Ref/Exclusion) enthalten sein. |
| {ReqFunc54} | In der Musterimplementierung soll mindestens eine Option mit einem required Parameter mit Typwandlung bool enthalten sein. Todo yannic |

{ReqFunc55} In der Musterimplementierung soll mindestens eine Option mit einem optional Parameter mit Typwandlung Integer enthalten sein, wobei der Integerwert aus DefaultValue ausgelesen wird.

Erstellungsprozess der Musterimplementierung

Es soll eine CMake Datei erstellt werden die die Musterimplementierung sowohl mit MinGw (Windows) als auch mit GNU-C/C++ (UNIX) erstellen kann.

{ReqFunc56} Es soll eine CMake Datei erstellt werden die die Musterimplementierung sowohl under MinGw (Windows) und GNU-C/C++ (UNIX) erstellen kann. Todo steve&ali

Dokumentation der Musterimplementierung

Die Musterimplementierung soll dokumentiert werden. Zusammen mit der erzeugten Header- und Source-Datei konnen die Dozenten die Funktion der einzelen Teilfunktionen¨ aus der Dokumentation verstehen.

{ReqFunc57} Die Dokumentation im Beispielprojekt soll im doxygenFormat sein.

{ReqFunc58} Die Kommentare der Methoden sollen doxgen-konform sein.

{ReqFunc59} Eine erzeugte Doxygen-Dokumentation soll der Abgabe beiliegen.

### 2.2 Nichtfunktionale Anforderungen

Nichtfunktionale Anforderungen sind umzusetzen. Im Gegensatz zu funktionalen Anforderungen, sind nichtfunktionale Anforderungen nicht direkt an Features des Programms gebunden sondern beschreiben die Umgebung und das Tooling.

#### 2.2.1 Tools und Build Umgebungen

Unterstutze Plattformen und Compiler¨

Um die Software auf den Systemen Windows 7/10/11 und Linux Ubuntu lauffahig zu¨ bekommen ist zwingend erforderlich einen Compiler zu verwenden, der die gleichen Kommandozeilenparameter versteht und frei von Lizenzen ist. Das ist aktuell beim MSVC2017 Compiler nicht gegeben.

{ReqNonFunc1} Die Software soll auf [MinGW64](https://www.mingw-w64.org/) fur Windows complilerbar sein¨

{ReqNonFunc2} Die Software soll mit [GNU-C/C++](https://gcc.gnu.org/) fur UNIX complilerbar sein¨ todo steve&Ali

Erstellung der ausfuhrbaren Datei¨

Um die Software frei von graphischen Entwicklungsumgebungen erstellen zu konnen wird¨ ein Makefile benotigt. Dieses kann dann unter mingw32-make (Windows) oder make¨ (UNIX) erstellt werden. Da der Umfang dieses von Hand zu erstellen den Umfang des Programmentwurfs uberschreiten w¨ urde, wird hier auf CMake als Erstellungstool f¨ ur das¨ Makefile verwiesen.

{ReqNonFunc3} Das Erstellen der Software soll mit [CMake](https://cmake.org/) erfolgen

Einlesen von XML-Dateien

Eine XML-Datei kann mit einer schier unuberschaubaren Anzahl von Libraries in C/C++¨ eingelesen werden. Die bekannteste hier ist XERCES von Apache. Apache ist ein kostenloses Framework zum hosten von XHTML Seiten. Die Library beinhaltet die Varianten SAX/SAX2 und DOM. SAX durchlauft das Dokument linear und DOM baut eine Baum-¨ struktur auf. Beides ist geeignet zum auslesen einer XML-Datei. Mit DOM kann auch eine XML-Datei erstellt werden.

Hinweis: Ob die XML-Datei als SAX oder DOM ausgelesen wird bleibt der Gruppe die die Implementierung erstellt uberlassen.¨

|  |  |
| --- | --- |
| {ReqNonFunc4} | Zum Einlesen der XML Datei soll die Softwarelibrary [XERCES-C](https://xerces.apache.org/xerces-c) verwendet werden oder optional ein selbstgeschriebener Parser (keine Zusatzpunkte furs Selber-¨ schreiben). |
| {ReqNonFunc5} | Der XML-Parser soll, mit Ausnahme der Xerces-DLL, keine weiteren DLLs oder andere Fremdsoftware benotigen.¨ |
| {ReqNonFunc6} | Der XML-Parser soll robust entwickelt sein und bei unerwarteten Formaten oder  Tags nicht absturzen sondern sich sauber mit einer Fehlermeldung beenden.¨ todo elias |

Dokumentation des Programmentwurf

Um den Dozenten Th. Staudacher und A. Maus die Chance zu geben, die Struktur und Funktionalitat des Programmentwurf zu¨ Uberblicken ist es dringend gegeben, dass¨ die Dokumentation des Codes im doxygen-Format erfolgt. Durch gute Dokumentation werden Ruckfragen im Fehlerfall minimiert und es k¨ onnen ggf. auch besser Teilpunkte¨ vergeben werden.

{ReqNonFunc7} Die Dokumentation des Codes muss mit dem Syntax von [doxygen](https://doxygen.nl/) erfolgen

Einlesen der Optionen

Um eine einheitliche Basis zu schaffen und das Programm portabel zu halten wird zum Parsen der Optionen die getopt( long) Implementierung verwendet. Das implementieren anderer Libraries oder eigengeschriebene Anwendungen ist nicht gestattet.

{ReqNonFunc8} Das Parsen der Optionen soll uber die LibC Funktionalit¨ at getopt(...) und getopt¨ long(...) aus getopt.h erfolgen

Dateien und Verzeichnisstruktur

Um eine saubere Trennung zwischen Implementierung und Deklarierung zu bekommen ist eine Auftrennung in zwei Verzeichnisse notig. Unterverzeichnisse in dieser Auftren-¨ nung sind ebenfalls moglich.¨

{ReqNonFunc9} Header- und Source Dateien sollen sich in verschiedenen Unterverzeichnissen befinden

{ReqNonFunc10} Header Dateien sollen relativ und nicht absolut includiert werden

Identifikation der Gruppe

Um die Bewertung der Gruppe durch zu fuhren ist die angef¨ ugte Datei students.xml¨ mit den Teilnehmer der Gruppe aus zu fullen. Die Angabe einer Mailadresse ist optional¨ (das Attribut kann auch weggelassen werden). Hilfreich ist aber wenn die Dozenten mit der Gruppe fur Nachfragen in Kontakt treten k¨ onnen.¨

|  |  |
| --- | --- |
| {ReqNonFunc11} | Die beigestellte Datei students.xml soll von der Gruppe vollstandig bef¨ ullt sein¨ und in jeder der Abgabe enthalten sein (Format utf-8) todo ali |
| {ReqNonFunc12} | Die beigestellte Datei students.xml soll im Hauptverzeichnis der Abgabe liegen todo ali |
| {ReqNonFunc13} | Die beigestellte Datei students.xml soll die abgebende Person kennzeichnen (Attribute deliver ist auf true zu setzen bei der abgebenden Person). Fur die anderen¨  Studenten bleibt dieses Attribut nicht belegt (oder mit false belegen) todo ali |

Beispiel einer students.xml richtig befullt¨ Diese students.xml ist mit einer Gruppengroße 4 Studenten bef¨ ullt. Diese Abgabe ist Herr Max Mustermann zu zu ordnen.¨

|  |
| --- |
| *<*?xml version=”1.0” encoding=”UTF=8”?*>*  *<*students*>*  *<*student deliver=”true” surname=”Max” name=”Mustermann” mail=”optional” course=”TIT21”/*>*  *<*student surname=”Erika” name=”Mustermann” mail=”optional” course=”TIK21” /*>*  *<*student surname=”John” name=”Doe” mail=”optional” course=”TIM21”/*>*  *<*student surname=”Jane” name=”Doe” mail=”optional” course=”TIS21”/*> <*/students*>* |

2

4

6

Programmierung

In dieser Sektion sind die Codierrichtlinien enthalten die dafur sorgen dass der Code¨ der geforderten Qualitat entspricht. Diese Codierrichtlinien sind unbedingt einzuhalten.¨ Muss aus irgendeinem Grund davon abgewichen werden so ist dies in einem Kommentar zu begrunden.¨

|  |  |
| --- | --- |
| {ReqNonFunc14} | Daten und Methoden die nur einer internen Berechnung dienen (nicht zum Export) sollen gegen externen Aufruf oder auslesen geschutzt werden¨ todo elias ali yannic steve chris |
| {ReqNonFunc15} | Zeiger sollen immer initialisiert werden, Ausnahmen nur gestattet bei entsprechender Kommentierung todo chris konst/destructor |
| {ReqNonFunc16} | Der Gultigkeitsbereich einer Variablen soll beachtet werden¨ |
| {ReqNonFunc17} | Werden logische Verknupfungen gemacht so sind logische Operatoren zu verwenden¨ |
| {ReqNonFunc18} | Berechnungen in logischen Verknupfungen sind nicht gestattet¨ |
| {ReqNonFunc19} | Werte die im weiteren Kontext nicht mehr verandert werden sollen, sind konstant¨ zu halten |
| {ReqNonFunc20} | Zeiger die im weiteren Kontext nicht mehr verandert werden sollen, sind konstant¨ zu halten |
| {ReqNonFunc21} | Das wegcasten von const ist nicht gestattet |
| {ReqNonFunc22} | Die Gultigkeit von Variablen soll an¨ Ubergabestellen¨ uberpr¨ uft werden¨ |
| {ReqNonFunc23} | Jede Header- und Source Datei soll am Anfang ein Hinweis auf den Bearbeiter enthalten todo ali elias yannic steve chris |
| {ReqNonFunc24} | Das kommentieren des Codes soll in deutscher Sprache erfolgen, optional in Englisch |
| {ReqNonFunc25} | Werden Variablen verwendet die eine fixe Bitlange ben¨ otigen so soll die Definition¨ aus cstdint verwendet werden |
| {ReqNonFunc26} | Berechnungen mit Fließkommazahlen sind nicht gestattet |
| {ReqNonFunc27} | Das verwenden von goto ist nicht gestattet |
| {ReqNonFunc28} | Das verwenden von #pragma ist nur gestattet solange dadurch keine Compilerabhangigkeit entsteht¨ |

### 2.3 Optionale Anforderungen

Optinale Requirements fliessen nicht in die Bewertung ein. Es hat sich aber in den letzten Jahren gezeigt, dass eine saubere Entwicklungsumgebung viele Fehler vermeidet. Es ist angeraten einige der optionalen Requirements mit um zu setzen.

#### 2.3.1 Tools und Build Umgebungen

Libraries fur C++¨

Es kann der ganze Umfang der mitgelieferten Headerdateien (soweit unter Windows und UNIX gleich einsetzbar) verwendet werden. Alternativ muss per Praprozessor jeweils¨ eine Umschaltung erfolgen. Vordefinerte Macros sind hierbei WIN32 , UNIX etc. Dies sind dann ggf. in der Dokumentation das Compilers nachzuschlagen. Um die volle Funktionalitat einer C++ Implementierung auspielen zu k¨ onnen, kann sowohl die STL¨ wie auch Boost als Library verwendet werden.

Anmerkung: Die STL ist Bestandteil des g++ Compilers, Boost hingegen nicht. Diese kann leicht unter UNIX installiert werden (boost-dev). Bei MinGw64 wird geraten gleich ein fertiges Paket aus [MinGW64+Boost](https://nuwen.net/mingw.html) zu installieren.

{ReqOptFunc1} Die Library [STL](https://de.wikipedia.org/wiki/Standard_Template_Library) kann vollumfanglich verwendet werden¨

{ReqOptFunc2} Die Library [Boost](https://www.boost.org/) kann vollumfanglich verwendet werden¨

Logging von Ereignissen

Um den Bildschirm nicht mit Meldungen zu uberfrachten hat es sich eingeb¨ urgert, ein¨ Logging zu verwenden. Das ursprunglich f¨ ur Java entwickelte Log4J wurde in fast alle¨ Programmiersprachen uberf¨ uhrt. Im Unterricht wurden zumindestens eines der beiden¨ Logging-Libraries Easylogging++ oder Boost Logging besprochen. Es ist dringend angeraten dieses auch im Programmentwurf ein zu setzen. Es kann auch helfen im Fehlerfall im Dialog mit den Studenten eine lauffahige Version nach Abgabe zu erzeugen, falls diese¨ fehlschlagt.¨

{ReqOptFunc3} Das Logging der Zustande w¨ ahrend des Programmablauf kann mit der Library¨ [Easylogging++](https://github.com/amrayn/easyloggingpp) erfolgen

{ReqOptFunc4} Das Logging der Zustande w¨ ahrend des Programmablauf kann mit der Library¨ [Boost Logging](https://www.boost.org/doc/libs/1_77_0/libs/log/doc/html/index.html) erfolgen

Formatieren der Ausgabe

Die Formatierung der Ausgabe ist kein Bestandteil des Programmentwurfs. Daher sollte keine Zeit in die Formatierung des Ergebnisses verwendet werden. Um aber eine bessere Ubersicht im Arbeitsergebnis zu erhalten w¨ are eine Formatierung der Header- und Source¨ Datei vorteilhaft. Hierbei bietet sich astyle an.

{ReqOptFunc5} Das Formatieren des erzeugten Codes kann mit [Artistic Style](http://astyle.sourceforge.net/) erfolgen

Testen von Teilfunktionalit¨aten

Um die einzelnen Programmteile auf Funktionalitat zu untersuchen ist es angeraten¨ einen Unittest fur die einzelnen Programmteile zu erstellen. Es hat sich aber in der Ver-¨ gangenheit gezeigt, das sowohl die Studenten wie auch die Dozenten aus den Unittest nachvollziehen konnten wie das Programm funktioniert und haufig auch Fehler gefunden¨ wurden. Zwei Libraries sind hier von Bedeutung, einmal die einfach zu implemntierende Catch2 die nur als reine Header Datei includiert werden kann sowie das etwas aufwendigere Framework von Boost Test. Der Vorteil von Catch2 ist die leichtere Implementierbarkeit, der Vorteil von Boost Test der bessere Funktionsumfang und die bessere Integrierbarkeit von C++.

{ReqOptFunc6} Das Testen der Module kann mit der Library [Catch2](https://github.com/catchorg/Catch2) erfolgen

{ReqOptFunc7} Das Testen der Module kann mit der Library [Boost Test](https://www.boost.org/doc/libs/1_79_0/libs/test/doc/html/index.html) erfolgen

### 2.4 Programmierung

{ReqOptFunc8} Das verwenden des ? Operators ist gestattet

{ReqOptFunc9} Bei einer Kombination aus if/else if das kein else hat kann das else mit einem leeren Codeblock mit entsprechender Kommentierung hinzugefugt werden¨

{ReqOptFunc10} Das verwenden von mehrdimensionalen Arrays soll vermieden werden

{ReqOptFunc11} Soweit technisch moglich sollten die Header- und Source Dateien in utf-8 Zeichen-¨ codierung erstellt werden

### 2.5 Programmbeispiele

**Hinweis:**Alle Beispiele sind auf Moodle in einer ZIP Datei eingestellt.

#### 2.5.1 Text Formatierung unter C++

Eine sehr gute Dokumentation der Formatierung unter ostream ist unter [COUT Formatierung](http://udojohn.de/pdf_dateien/cout_Formatierung.pdf) beschrieben.

**Anmerkung**: Zertifikat wird als nicht sicher angegeben. Es handelt sich hierbei um eine PDF.

#### 2.5.2 Text Formatierung unter C++ analog printf

Sehr prazise kann Text unter C formatiert werden. Im Prinzip gibt es diese Optionen¨ auch unter C++. Ein einfaches Mittel die C Formatierung von printf auch unter C++ zu ubernehmen ist die Template Klasse von Boost Format. Der Syntax ist identisch zu¨ printf und es gibt auch Erweiterungen.

|  |
| --- |
| //https ://www. boost . org/doc/ libs /1 75 0/ libs /format/doc/format . html  #include *<*iostream*>* using namespace std ;  #include *<*boost/format .hpp*>* using namespace boost ; |

2

4

6

|  |  |  |
| --- | --- | --- |
| int | main() {  cout *<<* format (”%=20s %s”) % ”Hello” % ”World” | *<<* endl ; |
| } | return 0; |  |

8

10

#### 2.5.3 String in Teilstrings zerlegen

Um Strings in Teilstrings zu zerlegen wird ein Tokenizer verwendet. Der unter C verwendete Tokenizer strtok() ist jedoch in mehrerer Hinsicht kritsich. Er nutzt statischen Speicher und ist nicht Threadfest. Ein Alternative hierfur ist der Tokenizer der in der¨ Boost Library Tokenizer angeboten wird.

|  |
| --- |
| //https ://www. boost . org/doc/ libs /1 79 0/ libs /tokenizer/doc/char separator .  htm  #include*<*iostream*>*  #include*<*boost/tokenizer . hpp*>* #include*<*string*>*  using namespace std ; using namespace boost ; const string s = {”1 ,3 ,5 ,7 ,8 ,9”};  int main() { char separator*<*char*>* sep (” ,”) ; tokenizer*<*boost : : char separator*<*char*>>* tok (s , sep ) ; for ( auto it=tok . begin () ; it !=tok . end () ; ++it ) { cout *<<* \* it *<<* endl ;  }  return 0;  } |

2

4

6

8

10

12

14

16

18

#### 2.5.4 Text in Zahlen wandeln

Wenn Texte in Zahlen gewandelt werden sollen wird meistens die Funktion atoi() und atod() verwendet.In der Boost Library LexicalCast gibt es spezialisierte Template Klasse die eine Wandlung durchfuhren k¨ onnen.¨

|  |
| --- |
| //https :// theboostcpplibraries .com/boost . lexical cast  #include*<*iostream*>*  #include *<*boost/ lexical cast . hpp*>* #include*<*string*>*  using namespace std ; using namespace boost ; |

2

4

6

8

|  |
| --- |
| const string s = {”123”};  int main() {  try  {  const int i = boost : : lexical cast*<*int*>*( s ) ; cout *<<* i *<<* endl ;  }  catch ( const boost : : bad lexical cast &e)  {  std : : cerr *<<* e . what() *<<* ’\n ’ ;  }  return 0;  } |

10

12

14

16

18

20

22

#### 2.5.5 Optionale Shortopts

Aus den Reihen der Studentenschaft kam der Einwand dass unter Windows keine optionale Shortopts moglich sind. Dieses Codebeispiel zeigt auf wie ein optionales Shortopt¨ Parsing zu programmieren ist.

|  |
| --- |
| // Beispiel fuer ein optionales getopt shortopt  // =O =*>* OK . . . option ’=O’ without argument  // =O5 =*>* OK . . . option =O with argument ’5 ’  #include *<*stdio .h*>*  #include *<*getopt .h*>*  #include *<*ctype .h*>*  int main ( int argc , char \*argv [ ] ) { int c ; opterr = 0; while (( c = getopt ( argc , argv , ”O: ”) ) != =1) switch (c)  {  case ’O’ : printf (”OK . . . option =%c with argument ’%s ’\n” , c , optarg ) ; break ;  case ’? ’ :  i f ( optopt == ’O’ && isprint ( optopt ) )  {  printf (”OK . . . option ’=O’ without argument \n”) ; break ;  }  else i f ( isprint ( optopt ) ) fprintf ( stderr , ”ERR . . . Unknown option ‘=%c ’.\n” , optopt )  ; else  fprintf ( stderr , ”ERR . . . Unknown option character ‘\\x%x ’.\n” , optopt ) ; |

2

4

6

8

10

12

14

16

18

20

22

24

26

|  |  |
| --- | --- |
| } | return =1;  default :  ;  }  return 0; |

28

30

32

34

#### 2.5.6 Text in Linien fester Breite wandeln

Um Texte in eine definierte Lucke zu pressen muss der Text in Teillinien gewandelt¨ werden. Das Verfahren heisst **Word Wrap**. Ein Beispiel wie so etwas geht wird in [Implementing Word Wrap in C#](https://www.codeproject.com/articles/51488/implementing-word-wrap-in-c) beschrieben. Diese ist einfach in C uberf¨ uhrbar, da¨ die Funktionalitat analog C# auch in C funktioniert. Der StringBuilder ist durch die¨ std::strstream zu ersetzen. Die White Spaces konnen mit der Funktion isspace() aus der¨ ctype Header Datei uberpr¨ uft werden.¨

#### 2.5.7 Unittest

Um Teileinheiten der Software (Klassen/Funktionen) zu testen ist ein Unittest das Mittel der Wahl. Gange Unittestframework helfen hier den Test auf zu setzen. Dabei wird eine¨ beim erstellen des Programm ein kompletter Menuaufbau erstellt.¨

Catch2 Catch2 ist eine Bibliothek die aus einer Header Datei generiert wird. Dazu muss in eine Datei die main erstellt werden was sehr lange dauert. Die Tests wiederum werden schnell compiliert. Am besten wird der Test dann mit cmake erstellt, so das die main nur einmal erstellt werden muss.

|  |
| --- |
| #define CATCH CONFIG MAIN  #define CATCH CONFIG FAST COMPILE  #include *<*catch .hpp*>*  // Statischer Teil der aus der Header Datei  // e r s t e l l t wird in eine Datei auslagern  //Das Object muss nur einmal e r s t e l l t werden |

2

4

6

Der eigentliche Test wird in eine andere Datei verlagert.

|  |
| --- |
| #define CATCH CONFIG FAST COMPILE  #include *<*catch .hpp*>*  bool isNotZero ( const int value ) { return value ;  }  TEST CASE(”Catch2Demonstrator”)  { |

2

4

6

8

|  |  |
| --- | --- |
| } | SECTION(”NotZeroTestCase”)  {  CHECK( isNotZero(=1) == true ) ; CHECK( isNotZero (1) == true ) ;  CHECK( isNotZero (0) == false ) ;  } |

10

12

14

16

Boost Test Ahnlich verh¨ alt es sich mit dem Framework Boost Test. Hier ist jedoch noch¨ entweder die statische Lib libboost unit test framework.a oder die dynamische Lib libboost unit test framework.so/libboost unit test framework.dll dazu zu linken. Bei Verwendung einer dynamsichen Lib wird das Programm wesentlich kleiner, Funktional gibt es keinen Unterschied. Von der Funktionalitat¨ ubertrifft Boost Test Catch2 bei weitem,¨ jedoch fur den Programmentwurf ist beides ausreichend gut¨

|  |
| --- |
| //g++ =o unittest unittest . cpp =lboost unit test framework  #define BOOST TEST MODULE BoostTestDemonstrator #include *<*boost/ test /included/ unit test . hpp*>*  bool isNotZero ( const int value ) { return value ;  }  BOOST AUTO TEST CASE( NotZeroTestCase )  {  BOOST TEST( isNotZero(=1) == true ) ; BOOST TEST( isNotZero (1) == true ) ;  BOOST TEST( isNotZero (0) == false ) ;  } |

2

4

6

8

10

12

14

#### 2.5.8 SAX mit Xerces-C

Die Xerces-C Library beinhaltet alles was ein guter XML Parser kann, SAX, Dome,

X-Path etc. Hier ein Beispiel mit SAX das ein gultiges XML Dokument parsen kann.¨

|  |
| --- |
| //g++ =o sax . exe sax . cpp =IC:\˜PathToLib˜\xerces=c\include =LC:\˜PathToLib  ˜\xerces=c\ lib =lxerces =c . dll =std=c++20  //Aufruf : sax datei . xml  #include *<*xercesc/ util /XMLString . hpp*>*  #include *<*xercesc/parsers/SAXParser . hpp*>*  #include *<*xercesc/sax/HandlerBase .hpp*>*  #include *<*xercesc/ util /OutOfMemoryException .hpp*>*  XERCES CPP NAMESPACE USE  #include *<*locale*>*  #include *<*codecvt*>*  #include *<*string*>* |

2

4

6

8

10

|  |
| --- |
| #include *<*iostream*>* using namespace std ;  class SimpleSAXParser : public HandlerBase {  public :  virtual void startDocument () { cout *<<* ”Dokument beginnt ! ” *<<* endl ;  }  virtual void endDocument() { cout *<<* ”Dokument ist zu Ende ! ” *<<* endl ;  }  virtual void startElement ( const XMLCh\* const name, AttributeList&  attributes ) { cout *<<* ”Start=Element : ” *<<* converter . to bytes (name) *<<* endl ;  for (XMLSize t i = 0; i *<* attributes . getLength () ; ++i ) cout *<<* converter . to bytes ( attributes . getName( i ) ) *<<* ’=’ *<<*  converter . to bytes ( attributes . getValue ( i ) ) *<<* endl ;  }  virtual void endElement( const XMLCh\* const name) { cout *<<* ”Element ist zu Ende: ” *<<* converter . to bytes (name) *<<* endl  ;  }  virtual void characters ( const XMLCh\* const chars , const XMLSize t  length ) { cout *<<* ”Buchstaben (” *<<* length *<<* ”) : ” *<<* converter . to bytes (  chars , chars + length ) *<<* endl ;  } private :  wstring convert*<*codecvt utf8 utf16*<*char16 t*>*, char16 t*>* converter ;  };  int main( int argc , char\* argv [ ] ) { try {  XMLPlatformUtils : : I n i t i a l i z e () ;  }  catch ( const XMLException& toCatch) { char\* message = XMLString : : transcode (toCatch . getMessage () ) ;  cerr *<<* ”Error during initialization ! :\n”  *<<* message *<<* ”\n” ;  XMLString : : release(&message) ; return 1;  }  SAXParser\* parser = { nullptr }; parser = new SAXParser ; int errorCount = {0};  try |

12

14

16

18

20

22

24

26

28

30

32

34

36

38

40

42

44

46

48

50

52

54

56

58

60

|  |  |  |  |
| --- | --- | --- | --- |
|  | {  //Das eigentliche Parsen der Datei SimpleSAXParser handler ; parser=*>*setDocumentHandler(&handler ) ; parser=*>*parse ( argv [1]) ;  errorCount = parser=*>*getErrorCount () ;  }  catch ( const OutOfMemoryException&)  {  XERCES STD QUALIFIER cerr *<<* ”OutOfMemoryException” *<<*  XERCES STD QUALIFIER endl ;  }  catch ( const XMLException& toCatch)  { char\* message = XMLString : : transcode (toCatch . getMessage () ) ;  //XMLString : : release (message) ;  /\*  XERCES STD QUALIFIER cerr *<<* ”\nAn error occurred\n Error : | | ” |
|  | *<<* | StrX(toCatch . getMessage () ) |  |
|  | *<<* | ”\n” *<<* XERCES STD QUALIFIER endl ; | |
| } | \*/  cerr *<<* ”XMLException : ” *<<* message *<<* endl ;  }  catch ( . . . ) { cerr *<<* ”Unbekannter Fehler” *<<* endl ;  }  cout *<<* ”Anzahl Fehler : ” *<<* errorCount *<<* endl ;  //Parser sauber beenden delete parser ;  //Terminate muss immer am Schluss stehen XMLPlatformUtils : : Terminate () ; return 0; | | |

62

64

66

68

70

72

74

76

78

80

82

84

86

88

90

92

94

96

#### 2.5.9 XML, utf-16, utf-8 (ASCII)

XML Parser arbeiten mit utf-16. Die Weiterverarbeitung in utf-16 ist sperrig. Eine Wandlung in utf-8 bietet sich hier an. Da im Programmentwurf nur mit ASCII gearbeitet wird ist utf-8 das richtige Format, denn ASCII sind die ersten 127 Zeichen von utf-8. Anbei ein paar Beispiele wie sich die utf-16 in utf-8 wandeln lasst.¨

|  |
| --- |
| #include *<*xercesc/ util /XMLString . hpp*>* const XMLCh \* const UTF16TEXT = {u ’ ’ Ich bin ein Text im utf =16 Format ’ ’ };  //Eine Wandlung in utf =8 geht mit der statischen Klasse XMLString : :  transcode |

2

4

|  |
| --- |
| const char \*utf=8 = XMLString : : transcode (UTF16TEXT) ; cout *<<* utf=8 *<<* endl ;  XMLString : : release(&utf =8) ;  //Aufwendig ist das freigeben des Speichers unter transcode .  //Die STL bietet hier eine einfachere Methode wstring convert*<*codecvt utf8 utf16*<*char16 t*>*, char16 t*>* convUTF16UTF8;  // Falls sich die strlen mit der Methode XMLString : : stringLen ermittel  laesst  string tagvalue = convUTF16UTF8=*>*tobytes ( chars ) ; // Falls eine Laengenangabe mit verarbeitet werden muss string tagvalue = convUTF16UTF8=*>*tobytes ( chars , chars + length ) ; |

6

8

10

12

14

16

#### 2.5.10 Muster XML

|  |
| --- |
| *<*?xml version=”1.0” encoding=”UTF=8” ?*>*  *<*GetOptSetup SignPerLine=”79”*>*  *<*Author Name=”Thomas Staudacher , Andreas Maus” Phone=”07541=77=961003”  Mail=”thomas . staudacher@zf .com, andreas . maus@zf .com” /*>*  *<*HeaderFileName*>*options .h*<*/HeaderFileName*>*  *<*SourceFileName*>*options . cpp*<*/SourceFileName*>*  *<*NameSpace*>*DHBW*<*/NameSpace*>*  *<*ClassName*>*COptionParser*<*/ClassName*>*  *<*OverAllDescription*>*  *<*Block*>*Erstellt einen Rumpf zum einlesen von Argumente aus der  Kommandozeile .*<*/Block*>*  *<*Block*>*Es kann sowohl mit innenliegenden Container wie externer  Klassenanbindung eine Datenhaltung erfolgen .*<*/Block*>*  *<*Block*>*Sobald ein Methodenaufruf abstrakt ist , wird die Basisklasse abstrakt .*<*/Block*>*  *<*Block*>*Fuer die Formatierung der generierten Dateien wird astyle verwendet .*<*/Block*>*  *<*/OverAllDescription*>*  *<*SampleUsage*>*  *<*Sample*>*getoptgen [ options ] . . . QUELLE*<*/Sample*>*  *<*Sample*>*getoptgen [==out=path ] . . . QUELLE*<*/Sample*>*  *<*/SampleUsage*>*  *<*Options*>*  *<*!== Option help greift auf die interne Klasseninterne Methode printHelp zu . Ein gleichzeitiger Aufruf mit version und parse=only ist nicht erlaubt . ==*>*  *<*Option Ref=”1” ShortOpt=”h” LongOpt=”help” Exclusion=”2 ,3”  ConnectToInternalMethod=”printHelp” Description=”Diese Hilfe ausgeben und beenden” /*>*  *<*!== Option version greift auf die interne Klassenexterne ( abstrakte ) Methode printVersion zu . Ein gleichzeitiger Aufruf mit helpund parse= only ist nicht erlaubt . ==*>*  *<*Option Ref=”2” ShortOpt=”v” Interface=”Version” Exclusion=”1 ,3” ConnectToInternalMethod=”printVersion” Description=”Gibt die Version des Programms aus und beendet” /*>* |

2

4

6

8

10

12

14

16

18

20

22

|  |
| --- |
| *<*!== Option out=path braucht ein zusa tzliches Argument und schreibt¨ auf einen Klassenintern generierten String das Argument . Die Option help ,  version und parse=only darf nicht angegeben sein ==*>*  *<*Option LongOpt=”out=path” HasArguments=”Required” Exclusion=”1 ,2 ,3” Interface=”OutputPath” Description=”Der Pfad wo das Ergebnis hingenriert werden soll ( sonst ins aktuelle Verzeichnis )” /*> <*Option LongOpt=”astyle=path” HasArguments=”Required” Interface=” AstylePath” Exclusion=”1 ,2 ,3” Description=”Der Pfad wo die Astyle executable gefunden werden kann” /*>*  *<*!== Option sign=per=line kann ein zusa tzliches Argument¨ ubergeben¨ werden und schreibt auf einen Klassenintern generierten Integer  SignPerLine . Wenn kein Parameter ubergeben wird¨ ist der Defaultwert 79.  Die Option help , version und parse=only darf nicht angegeben sein ==*>*  *<*Option LongOpt=”sign=per=line ” HasArguments=”optional” Interface=”  SignPerLine” Exclusion=”1 ,2 ,3” ConvertTo=”Integer” DefaultValue=”79”  Description=”Die Anzahl der Zeichen pro Linie fur¨ den Helptext . Ohne  Argument wird der Standartwert genommen. ” /*>*  *<*Option ShortOpt=”n” LongOpt=”only=if =newer” Interface=”OnlyIfNewer” Exclusion=”1 ,2 ,3” Description=”Generiert nur wenn die Eingangsdatei neuer ist wie die bereits generierte ” /*>*  *<*Option LongOpt=”no=format” Interface=”NoFormat” Exclusion=”1 ,2 ,3”  Description=”Erzeugte Datei wird nicht formatiert” /*>*  *<*!== Option parse=only l i e s s t ein zusa tzliches Argument ein und ruft¨ die interne Klassenexterne ( abstrakte ) Methode ParseXML auf . Die Option  help , version darf nicht angegeben sein ==*>*  *<*Option Ref=”3” LongOpt=”parse=only” HasArguments=”Required”  ConnectToInternalMethod=”ParseXML” Exclusion=”1 ,2” Description=”Parst die Datei einmal und beendet das Programm” /*> <*/Options*>*  *<*/GetOptSetup*>* |

24

26

28

30

32