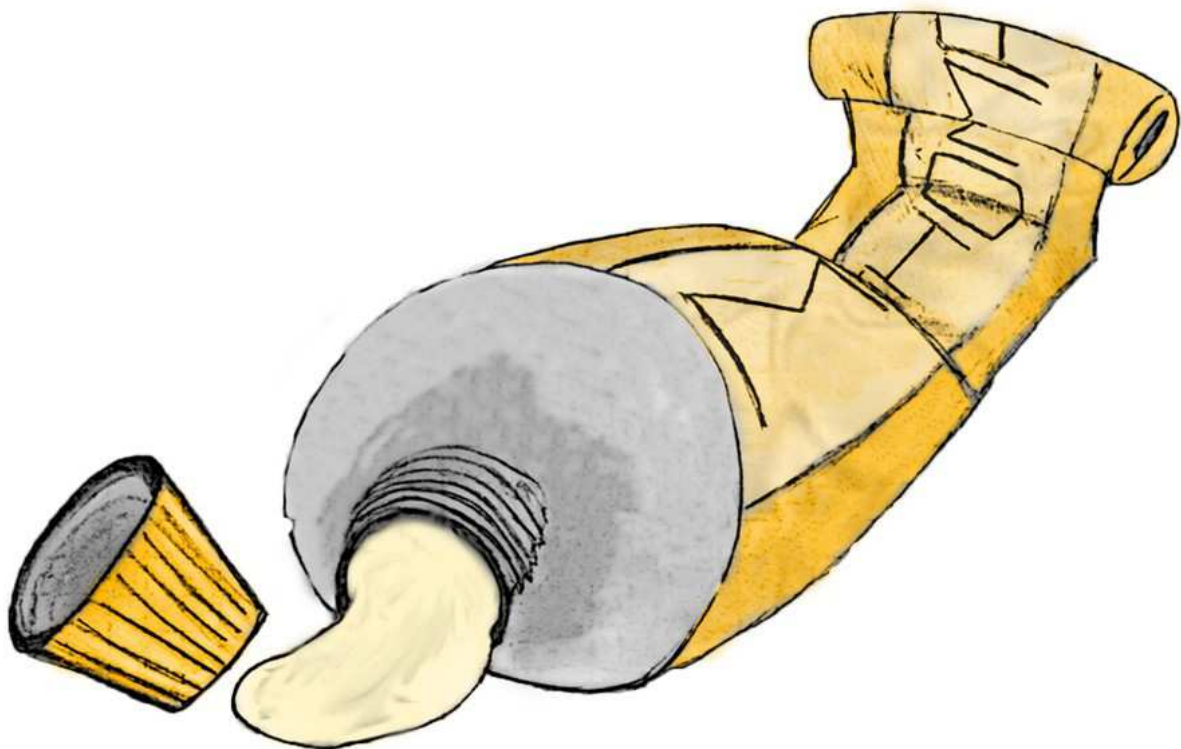


Qualitätshandbuch „MathoMadaja“



MATHOMADAJA

Inhaltsverzeichnis:

1. Getestete Funktionen mit Quellcode der Test-Funktion

1.1 Funktion kopieren.....	3
1.2 Funktion aufrufen.....	5
1.3 Funktion ändern.....	6
1.4 Funktion löschen.....	7
1.5 Funktion eingabe_neu.....	9
1.6 Funktion eingabeliste_Sichern.....	10
1.7 Funktion eingabeliste_Laden.....	11
1.8 Funktion logdatei.....	12

2. Test-Main-Funktion mit Quellcode.....13

1. Getestete Funktionen

1.1 Funktion kopieren getestet.

Erster Test: Erfolgreiches Ordner kopieren

Es wurde vom Ordner `C:\\Test Ordner 1\\uOrdner1` in den Ordner `C:\\Test Ordner 2` kopiert und von `C:\\Test Ordner 1\\uOrdner2` in den Ordner `C:\\Test Ordner 2` kopiert, dabei wurde mit `EQUAL` auf den Rückgabewert `ERFOLG` und mit `NOT_EQUAL` auf `SFEHLER` geprüft.

```
void Test_Kopieren_Ordner_E(void){
    CU_ASSERT_EQUAL (kopieren("C:\\Test Ordner 1\\uOrdner1", "C:\\Test
    Ordner 2"), ERFOLG);
    CU_ASSERT_NOT_EQUAL (kopieren("C:\\Test Ordner 1\\uOrdner2",
    "C:\\Test Ordner 2"), SFEHLER);
}
```

Zweiter Test: Erfolgreiches Datei kopieren

Es wurde die Datei von `C:\\Test Ordner 1\\uOrdner1\\Test Datei 1.txt` in den Ordner `C:\\Test Ordner 2` kopiert und die Datei von `C:\\Test Ordner 1\\uOrdner1\\Test Datei 2.txt` in der Ordner `C:\\Test Ordner 2` kopiert, dabei wurden mit `EQUAL` auf den Rückgabewert `ERFOLG` und mit `NOT_EQUAL` auf `SFEHLER` geprüft.

```
void Test_Kopieren_Datei_E(void){
    CU_ASSERT_EQUAL (kopieren("C:\\Test Ordner 1\\uOrdner1\\Test Datei
    1.txt", "C:\\Test Ordner 2"), ERFOLG);
    CU_ASSERT_NOT_EQUAL (kopieren("C:\\Test Ordner 1\\uOrdner1\\Test
    Datei 2.txt", "C:\\Test Ordner 2"), SFEHLER);
}
```

Dritter Test: Fehlerhaftes Ordner kopieren

Es soll der **nicht** vorhanden Ordner `C:\\Ordner 3` in den Ordner `C:\\Ordner 2` kopiert werden und der **nicht** vorhanden Ordner `C:\\Ordner 1\\uOrdner3` in den Ordner `C:\\Ordner 2` kopiert werden, dabei wurde mit `EQUAL` auf den Rückgabewert `SFEHLER` und mit `NOT_EQUAL` auf `ERFOLG` geprüft.

```
void Test_Kopieren_Ordner_F(void){
    CU_ASSERT_EQUAL (kopieren("C:\\Ordner 3", "C:\\Ordner 2"), SFEHLER);
    CU_ASSERT_NOT_EQUAL (kopieren("C:\\Ordner 1\\uOrdner3", "C:\\Ordner
2"), ERFOLG);
}
```

Vierter Test: Fehlerhaftes Datei kopieren

Es sollte die **nicht** vorhandene Datei `C:\\Test Ordner 1\\Test Datei 3.txt` in den Ordner `C:\\Test Ordner 2` kopiert werden und die **nicht** vorhandene Datei `C:\\Test Ordner 1\\Test Datei 2.txt` in den Ordner `C:\\Test Ordner 2` kopiert werden, dabei wurde mit `EQUAL` auf den Rückgabewert `SFEHLER` und mit `NOT_EQUAL` auf `ERFOLG` geprüft.

```
void Test_Kopieren_Datei_F(void){
    CU_ASSERT_EQUAL (kopieren("C:\\Test Ordner 1\\Test Datei 3.txt",
"C:\\Test Ordner 2"), SFEHLER);
    CU_ASSERT_NOT_EQUAL (kopieren("C:\\Test Ordner 1\\Test Datei 2.txt",
"C:\\Test Ordner 2"), ERFOLG);
}
```

1.2 Funktion aufrufen.

Erster Test: Erfolgreiches aufrufen

Es wurde die Datei `C:\\Test Ordner 1\\uOrdner1\\Test Datei 1.txt` und die Datei `C:\\Test Ordner 1\\uOrdner1\\Test Datei 2.txt` aufgerufen, der Rückgabewert wurde mit `EQUAL` auf `ERFOLG` und mit `NOT_EQUAL` auf `SFEHLER` geprüft.

```
void Test_Aufrufen_E(void){
    CU_ASSERT_EQUAL (aufrufen ("C:\\Test Ordner 1\\uOrdner1\\Test Datei
    1.txt"), ERFOLG);
    CU_ASSERT_NOT_EQUAL (aufrufen ("C:\\Test Ordner 1\\uOrdner1\\Test
    Datei 2.txt"), SFEHLER);
}
```

Zweiter Test: Fehlerhaftes aufrufen

Es wurde die **nicht** vorhandene Datei `C:\\Test Ordner 1\\Test Datei 3.txt` und die **nicht** vorhandene Datei `C:\\Test Ordner 1\\Test Datei 3.txt` aufgerufen, der Rückgabewert wurde mit `EQUAL` auf `SFEHLER` und mit `NOT_EQUAL` auf `ERFOLG` geprüft.

```
void Test_Aufrufen_F(void){
    CU_ASSERT_EQUAL (aufrufen ("C:\\Test Ordner 1\\Test Datei 3.txt"),
    SFEHLER);
    CU_ASSERT_NOT_EQUAL (aufrufen ("C:\\Test Ordner 1\\Test Datei
    3.txt"), ERFOLG);
}
```

1.3 Funktion ändern.

Erster Test: Erfolgreiches Datei ändern

Es wurden die Dateien im Ordner `C:/Test Ordner 1/*.txt` auf `aendern_fertig#.txt` geändert und danach gleich wieder auf `aendern2_fertig#.txt` geändert, dabei wurde der Rückgabewert mit `EQUAL` auf `ERFOLG` und mit `NOT_EQUAL` auf `SFEHLER` geprüft.

```
void Test_Aendern_Datei_E(void){  
    CU_ASSERT_EQUAL (aendern("C:/Test Ordner 1/*.txt",  
        "aendern_fertig#.txt"), ERFOLG);  
    CU_ASSERT_NOT_EQUAL (aendern("C:/Test Ordner 1/*.*",  
        "aendern2_fertig#.txt"), SFEHLER);  
}
```

Zweiter Test: Fehlerhaftes Datei ändern

Es sollten die **nicht** vorhanden Dateien im Ordner `C:/Test Ordner 1/*.thomas_ist` auf `aendern3_fertig#.txt` geändert werden und danach die **nicht** vorhanden Dateien im Ordner `C:/Test Ordner 1/*.der_besste` auf `aendern4_fertig#.txt` geändert werden, dabei wurde der Rückgabewert mit `EQUAL` auf `SFEHLER` und mit `NOT_EQUAL` `ERFOLG` geprüft.

1.4 Funktion löschen.

Erster Test: Erfolgreiches Ordner löschen

Es wurde der Ordner `C:\\Test Ordner 2\\uOrdner1` gelöscht und der Ordner `C:\\Test Ordner 2\\uOrdner2` gelöscht, dabei wurde der Rückgabewert mit `EQUAL` auf `ERFOLG` und mit `NOT_EQUAL` auf `SFEHLER` geprüft.

```
void Test_Loeschen_Ordner_E(void){
    CU_ASSERT_EQUAL (loeschen("C:\\Test Ordner 2\\uOrdner1"), ERFOLG);
    CU_ASSERT_NOT_EQUAL (loeschen("C:\\Test Ordner 2\\uOrdner2"),
        SFEHLER);
}
```

Zweiter Test: Erfolgreiches Datei löschen

Es wurde die Datei `C:\\Test Ordner 2\\Test Datei 1.txt` gelöscht und die Datei `C:\\Test Ordner 2\\Test Datei 2.txt` gelöscht, dabei wurde der Rückgabewert mit `EQUAL` auf `ERFOLG` und mit `NOT_EQUAL` auf `SFEHLER` geprüft.

```
void Test_Loeschen_Datei_E(void){
    CU_ASSERT_EQUAL (loeschen("C:\\Test Ordner 2\\Test Datei 1.txt"),
        ERFOLG);
    CU_ASSERT_NOT_EQUAL (loeschen("C:\\Test Ordner 2\\Test Datei 2.txt"),
        SFEHLER);
}
```

Dritter Test: Fehlerhaftes Ordner löschen

Es sollte der **nicht** vorhandene Ordner `C:\\Test Ordner 3` gelöscht werden und der **nicht** vorhandene Ordner `C:\\Test Ordner 3` gelöscht werden, dabei wurde der Rückgabewert mit `EQUAL` auf `SFEHLER` und `NOT_EQUAL` auf `ERFOLG` geprüft.

```
void Test_Loeschen_Ordner_F(void){
    CU_ASSERT_EQUAL (loeschen("C:\\Test Ordner 3"), SFEHLER);
    CU_ASSERT_NOT_EQUAL (loeschen("C:\\Test Ordner 3"), ERFOLG);
}
```

Vierter Test: Fehlerhaftes Datei löschen

Es sollte die **nicht** vorhandene Datei `C:/Test Ordner 2/Test Datei 3.txt` gelöscht werden und die **nicht** vorhandene Datei `C:/Test Ordner 2/Test Datei 3.txt` gelöscht werden, dabei wurde der Rückgabewert mit `EQUAL` auf `SFEHLER` und `NOT_EQUAL` auf `ERFOLG` geprüft.

```
void Test_Loeschen_Datei_F(void){
    CU_ASSERT_EQUAL (loeschen("C:/Test Ordner 2/Test Datei 3.txt"),
    SFEHLER);
    CU_ASSERT_NOT_EQUAL (loeschen("C:/Test Ordner 2/Test Datei 3.txt"),
    ERFOLG);
}
```

Als nächstes wurden die „unter“ Funktionen getestet. Jede zu testende Funktion wurde in einer Test-Funktion geprüft.

1.5 Funktion eingabe_neu.

Als erstes wurde bevor die Funktion getestet werden konnte eine `struct` vom typ `TERMIN` „befüllt“. Der zweite Schritt war das testen der Funktion mit der Übergabe der korrekten Werte. Daraufhin wurde die der Rückgabepointer mit `NOT_EQUAL` auf `NULL` geprüft.

```
void Test_Eingabe_neu(void){

    /* Variable für Termin */
    TERMIN T_term;

    time_t t_zeit;
    struct tm *tm_zeitinfo;

    /* damit die Datumsausgabe deutsch ist (Systemeinstellung) */
    setlocale(LC_ALL, "");
    /* aktuelle Kalenderzeit abrufen und speichern */
    time(&t_zeit);
    /* als lokale Zeit in Elemente der Struktur zerlegen */
    tm_zeitinfo = localtime(&t_zeit);
    T_term.D_Datum.iJahr      = tm_zeitinfo->tm_year - 100 + 2000;
    T_term.D_Datum.iMonat    = tm_zeitinfo->tm_mon + 1;
    T_term.D_Datum.iTag      = tm_zeitinfo->tm_mday;
    T_term.Z_Zeit.iStunde    = tm_zeitinfo->tm_hour;
    T_term.Z_Zeit.iMinute    = tm_zeitinfo->tm_min;
    T_term.Z_Zeit.iSekunde   = tm_zeitinfo->tm_sec;
    T_term.R_Regelmaessigkeit = MONATLICH;

    /** Test: Daten eingelesen
    */
    CU_ASSERT_NOT_EQUAL (eingabe_neu( "C:/Test Ordner 1", "C:/Test Ordner 2",
    "Test.txt", AUFRUFEN, T_term, 5 ), NULL);

}
```

1.6 Funktion eingabeliste_Sichern.

Als erstes wurde bevor die Funktion getestet werden konnte eine `struct` vom typ `EINGABELISTE` „befüllt“.

Der erste Test war das testen der Funktion mit der Übergabe der korrekten Werte.

Der zweite Teste mit der Übergabe eines **falschen** Pfads.

```
void Test_Eingabeliste_Sichern(void){

/* Schleife zur Erstellung der Engabeliste*/
    EINGABELISTE *EL_l = NULL;

    DOPPELANKER DA;
    TERMIN T_term;
    STATUS S_stat;

    const char * test = "HALLO_Test";
    time_t t_zeit;
    struct tm *tm_zeitinfo;

/* damit die Datumsausgabe deutsch ist (Systemeinstellung) */
    setlocale(LC_ALL, "");
/* aktuelle Kalenderzeit abrufen und speichern */
    time(&t_zeit);
/* als lokale Zeit in Elemente der Struktur zerlegen */
    tm_zeitinfo = localtime(&t_zeit);
    T_term.D_Datum.iJahr      = tm_zeitinfo->tm_year - 100 + 2000;
    T_term.D_Datum.iMonat    = tm_zeitinfo->tm_mon + 1;
    T_term.D_Datum.iTag      = tm_zeitinfo->tm_mday;
    T_term.Z_Zeit.iStunde    = tm_zeitinfo->tm_hour;
    T_term.Z_Zeit.iMinute    = tm_zeitinfo->tm_min;
    T_term.Z_Zeit.iSekunde   = tm_zeitinfo->tm_sec;
    T_term.R_Regelmaessigkeit = MONATLICH;
    eingabeliste_Einfuegen(&EL_l, "test", "\0", "\0", AENDERN, T_term,5);

/** Test: Erfolgreiches und nicht Erfolgreiches Engabeliste sichern
*/
    CU_ASSERT_EQUAL (eingabeliste_Sichern(EL_l, "C:/Test Ordner
1/Eingabe_Liste.mmdj"), ERFOLG);
    CU_ASSERT_EQUAL (eingabeliste_Sichern(EL_l, "C:/Test Ordner
3/Eingabe_Liste.mmdj"), SFEHLER);
}
```

1.7 Funktion eingabeliste_Laden.

Als erstes wurde bevor die Funktion getestet werden konnte eine `struct` vom typ `EINGABELISTE` „befüllt“.

Der erste Test war das testen der Funktion mit der Übergabe der korrekten Werte.

Der zweite Teste mit der Übergabe eines **falschen** Pfads.

```
void Test_Eingabeliste_Laden(void){

    EINGABELISTE *EL_l = NULL;

    DOPPELANKER DA;
    TERMIN T_term;
    STATUS S_stat;

    const char * test = "HALLO_Test";
    time_t t_zeit;
    struct tm *tm_zeitinfo;

    /* damit die Datumsausgabe deutsch ist (Systemeinstellung) */
    setlocale(LC_ALL, "");
    /* aktuelle Kalenderzeit abrufen und speichern */
    time(&t_zeit);
    /* als lokale Zeit in Elemente der Struktur zerlegen */
    tm_zeitinfo = localtime(&t_zeit);
    T_term.D_Datum.iJahr      = tm_zeitinfo->tm_year - 100 + 2000;
    T_term.D_Datum.iMonat    = tm_zeitinfo->tm_mon + 1;
    T_term.D_Datum.iTag      = tm_zeitinfo->tm_mday;
    T_term.Z_Zeit.iStunde    = tm_zeitinfo->tm_hour;
    T_term.Z_Zeit.iMinute    = tm_zeitinfo->tm_min;
    T_term.Z_Zeit.iSekunde   = tm_zeitinfo->tm_sec;
    T_term.R_Regelmaessigkeit = MONATLICH;
    eingabeliste_Einfuegen(&EL_l, "test", "\0", "\0", AENDERN, T_term,
7);

/** Test: Erfolgreiches und nicht Erfolgreiches Eingabeliste_Laden
*/
    CU_ASSERT_EQUAL (eingabeliste_Laden(&EL_l, "C:/Test Ordner
1/Eingabe_Liste.mmdj"), ERFOLG);
    CU_ASSERT_EQUAL (eingabeliste_Laden(&EL_l, "C:/Test Ordner
3/Eingabe_Liste.mmdj"), SFEHLER);
}
```

1.8 Funktion logdatei.

Als erstes wurde bevor die Funktion getestet werden konnte eine `struct` vom typ `TERMIN` „befüllt“.

Der erste Test war das testen der Funktion mit der Übergabe der korrekten Werte.

Der zweite Teste mit der Übergabe eines **falschen** Pfads der anzulegenden logdatei.

```
void Test_Logdatei(void){

    /* Variable für Termin */
    TERMIN T_term;

    time_t t_zeit;
    struct tm *tm_zeitinfo;

    /* damit die Datumsausgabe deutsch ist (Systemeinstellung) */
    setlocale(LC_ALL, "");
    /* aktuelle Kalenderzeit abrufen und speichern */
    time(&t_zeit);
    /* als lokale Zeit in Elemente der Struktur zerlegen */
    tm_zeitinfo = localtime(&t_zeit);
    T_term.D_Datum.iJahr      = tm_zeitinfo->tm_year - 100 + 2000;
    T_term.D_Datum.iMonat    = tm_zeitinfo->tm_mon + 1;
    T_term.D_Datum.iTag      = tm_zeitinfo->tm_mday;
    T_term.Z_Zeit.iStunde    = tm_zeitinfo->tm_hour;
    T_term.Z_Zeit.iMinute    = tm_zeitinfo->tm_min;
    T_term.Z_Zeit.iSekunde   = tm_zeitinfo->tm_sec;
    T_term.R_Regelmaessigkeit = MONATLICH;

    /** Test: Erfolgreiches und nicht Erfolgreiches erstellen einer logdatei
    */
    CU_ASSERT_EQUAL (logdatei("C:/Test Ordner 1/Logdatei.txt", START,
        ERFOLG, T_term, LOESCHEN, "C:/Test Ordner 1/Eingabe_Liste.mmdj",
        NULL), ERFOLG);
    CU_ASSERT_EQUAL (logdatei("C:/Test Ordner 3/Logdatei.txt", START,
        ERFOLG, T_term, LOESCHEN, "C:/Test Ordner 1/Eingabe_Liste.mmdj",
        NULL), SFEHLER);
}
```

2. Test-Main-Funktion

In der testmain.cpp wurden als erstes die zwei Suites SuitesERFOLG und SuitesSFEHLER erstellt. Der nächste Schritt war das einbinden der Funktionen in der jeweiligen Suite, das aufrufen der Tests und ausführen des Basic-Tests.

```
int main(void){

/** Pointer auf die Suites
*/
    CU_pSuite SuiteERFOLG, SuiteSFEHLER;

/** Pointer auf einen aktuellen Test
*/
    CU_pTest Test = NULL;

/** Registry initialisieren
*/
    if (CUE_NOMEMORY == CU_initialize_registry()) {
        printf("\nInitialization of Test Registry failed.");
        return -1;
    }

/** Eine neue Suite wird erstellt für korrekten Vorgang
*/
    SuiteERFOLG = CU_add_suite("Prüfe Funktionen auf korrekten
Vorgang",NULL,NULL);

    Test = CU_add_test(SuiteERFOLG,"Erfolgreiches Ordner
kopieren",Test_Kopieren_Ordner_E);
    Test = CU_add_test(SuiteERFOLG,"Erfolgreiches Datei
kopieren",Test_Kopieren_Datei_E);
    Test = CU_add_test(SuiteERFOLG,"Erfolgreiches
aufrufen",Test_Aufrufen_E);
    Test = CU_add_test(SuiteERFOLG,"Erfolgreiches Datei
ändern",Test_Aendern_Datei_E);
    Test = CU_add_test(SuiteERFOLG,"Erfolgreiches Ordner
löschen",Test_Loeschen_Ordner_E);
    Test = CU_add_test(SuiteERFOLG,"Erfolgreiches Datei
löschen",Test_Loeschen_Datei_E);
    Test = CU_add_test(SuiteERFOLG,"eingabe_neu",Test_Eingabe_neu);
    Test =
    CU_add_test(SuiteERFOLG,"eingabeliste_Sichern",Test_Eingabeliste_Sich
ern);
    Test =
    CU_add_test(SuiteERFOLG,"eingabeliste_Laden",Test_Eingabeliste_Laden)
;
    Test = CU_add_test(SuiteERFOLG,"logdatei",Test_Logdatei);
```

```
/** Eine neue Suite wird erstellt für falschen Vorgang
*/
SuiteSFEHLER = CU_add_suite("Prüfe Funktionen auf falschen
Vorgang",NULL,NULL);

Test = CU_add_test(SuiteSFEHLER,"Fehlerhaftes Ordner
kopieren",Test_Kopieren_Ordner_F);
Test = CU_add_test(SuiteSFEHLER,"Fehlerhaftes Datei
kopieren",Test_Kopieren_Datei_F);
Test = CU_add_test(SuiteSFEHLER,"Fehlerhaftes
aufrufen",Test_Aufrufen_F);
Test = CU_add_test(SuiteSFEHLER,"Fehlerhaftes Datei
ändern",Test_Andern_Datei_F);
Test = CU_add_test(SuiteSFEHLER,"Fehlerhaftes Ordner
löschen",Test_Loeschen_Ordner_F);
Test = CU_add_test(SuiteSFEHLER,"Fehlerhaftes Datei
löschen",Test_Loeschen_Datei_F);

/* Run all tests using the basic interface */
CU_basic_set_mode(CU_BRM_VERBOSE);
CU_basic_run_tests();
printf("\n");
CU_basic_show_failures(CU_get_failure_list());
printf("\n\n");

/** Registry säubern
*/
CU_cleanup_registry();

return 0;
```

```
C:\WINDOWS\system32\cmd.exe

Suite: Prüfe Funktionen auf korrekten Vorgang
Test: Erfolgreiches Ordner kopieren ... passed
Test: Erfolgreiches Datei kopieren ... passed
Test: Erfolgreiches aufrufen ... passed
Test: Erfolgreiches Datei ändern ... passed
Test: Erfolgreiches Ordner löschen ... passed
Test: Erfolgreiches Datei löschen ... passed
Test: eingabe_neu ... passed
Test: eingabeliste_Sichern ... passed
Test: eingabeliste_Laden ... passed
Test: logdatei ... passed
Suite: Prüfe Funktionen auf falschen Vorgang
Test: Fehlerhaftes Ordner kopieren ... passed
Test: Fehlerhaftes Datei kopieren ... passed
Test: Fehlerhaftes aufrufen ... passed
Test: Fehlerhaftes Datei ändern ... passed
Test: Fehlerhaftes Ordner löschen ... passed
Test: Fehlerhaftes Datei löschen ... passed

--Run Summary: Type      Total    Ran    Passed    Failed
                 suites      2       2      n/a      0
                 tests     16      16      16      0
                 asserts   31      31      31      0

Drücken Sie eine beliebige Taste . . . _
```