1 Einführung ........................................................................................................... 3
  1.1 Umfang und Lebenszeit von Software ........................................................... 4
  1.2 Relevanz der Thematik ................................................................................. 5
  1.3 Image ........................................................................................................... 6
  1.4 Was ist Software-Wartung? ......................................................................... 7
  1.5 Warum ist Wartung schwierig (teuer)? ....................................................... 10
  1.6 Szenarien .................................................................................................... 14
2 Organisation von Wartungseinheiten (WE) ..................................................... 14
  2.1 Einbettung der WE in das Unternehmen ..................................................... 16
  2.2 Wer führt W durch? .................................................................................... 17
  2.3 Organisation von Wartungsprojekten .......................................................... 19
  2.4 Richtlinien (RL) ........................................................................................ 20
  2.5 Wissensmanagement ................................................................................. 21
3 Prozesse ............................................................................................................. 22
  3.1 Entwicklungsprozesse ................................................................................ 22
  3.2 Wartungsprozesse ...................................................................................... 23
  3.3 Änderungsmanagement .............................................................................. 25
  3.4 Konfigurationsmanagement ....................................................................... 30
  3.5 System-Bewertung .................................................................................... 40
4 Techniken .......................................................................................................... 43
  4.1 Software-Metriken ...................................................................................... 43
  4.2 Redundanlyse (engl.: Clone Detection) ...................................................... 55
  4.3 Software-Visualisierung ............................................................................ 59
  4.4 Software Dokumentation .......................................................................... 60
  4.5 Software Generieren .................................................................................. 67
  4.6 Program-Transformation .......................................................................... 67
  4.7 Impact Analyse (Slicing) ............................................................................ 68
  4.8 Flexibilisierung .......................................................................................... 68
  4.9 Release-Planning ....................................................................................... 69
Kap 1.2. Relevanz

/* ugly.c */

#include <stdio.h>
main(t,_,a)
char *a;
{
  return!0<t?t<3?main(-79,-13,a+main(-87,1-_ ,main(-
86,0,a+1)+a)):
1,t<_?main(t+1,_,a):3,main(-94,-27+t,a) &t==2?<_<13?
main(2, _+1,"%s %d %d\n":9:16:t<0?t<-72?main(_,t,
"@n'++,#"/*{w+/w#cdnr/+,{r/*de}+,*{+,/w{%+,/w#q#
+},/#{1+/,n{+,/+#n+},/#
;#q#n+/,+k#;*+,/'r :'d*'3,}w+K w'K:'+}e#';dq#'l 
q#'d'K#!/+k#;q#'r}eKK#}w'r}eKK{n1]'#/;#q'n'){}w'
{}{n1]'/+#n';d)rw' i;# 
){n1]!/n{n#' r{w'r nc{n1]'/#{1,+K {rw
iK{[n1]'/w#q#n'wk nw' 

iwk{KK{n1]'/w{%1##w' i;
{:n1]'/*{q#'ld;r'}{n1wb!/*de}'c 
; {n1}-{}rw}'/+},##'*}#nc,',#nw]'/+kd'+e}+;#'rdq#w!
nr'/ ') }+}{rl'#{n' ')#
}'+}##(!/*")
t<-50?_ ==a?putchar(31[a]):main(-
65,_,a+1):main(-(*a=='/')+t,_,a+1)
:0<t?main(2,2,"%s"):a=='/'||main(0,main(-61,*a,
"ek;dc i@B'(q)-[w]*%n+r3#1,}:

nuwloca-0;m .vpbks,fxntdCeghiry"),a+1);
}
On the first day of Christmas my true love gave to me
a partridge in a pear tree.

On the second day of Christmas my true love gave to me
two turtle doves
and a partridge in a pear tree.

On the third day of Christmas my true love gave to me
three French hens, two turtle doves
and a partridge in a pear tree.

On the fourth day of Christmas my true love gave to me
four calling birds, three French hens, two turtle doves
and a partridge in a pear tree.

On the fifth day of Christmas my true love gave to me
five gold rings;
four calling birds, three French hens, two turtle doves
and a partridge in a pear tree.

On the sixth day of Christmas my true love gave to me
six geese a-laying, five gold rings;
four calling birds, three French hens, two turtle doves
and a partridge in a pear tree.

On the seventh day of Christmas my true love gave to me
seven swans a-swimming,
six geese a-laying, five gold rings;
four calling birds, three French hens, two turtle doves
and a partridge in a pear tree.

On the eighth day of Christmas my true love gave to me
eight maids a-milking, seven swans a-swimming,
six geese a-laying, five gold rings;
four calling birds, three French hens, two turtle doves
and a partridge in a pear tree.

On the ninth day of Christmas my true love gave to me
nine ladies dancing, eight maids a-milking, seven swans a-swimming,
six geese a-laying, five gold rings;
four calling birds, three French hens, two turtle doves
and a partridge in a pear tree.

On the tenth day of Christmas my true love gave to me
ten lords a-leaping,
nine ladies dancing, eight maids a-milking, seven swans a-swimming,
six geese a-laying, five gold rings;
four calling birds, three French hens, two turtle doves
and a partridge in a pear tree.

On the eleventh day of Christmas my true love gave to me
eleven pipers piping, ten lords a-leaping,
nine ladies dancing, eight maids a-milking, seven swans a-swimming,
six geese a-laying, five gold rings;
four calling birds, three French hens, two turtle doves
and a partridge in a pear tree.

On the twelfth day of Christmas my true love gave to me
twelve drummers drumming, eleven pipers piping, ten lords a-leaping,
nine ladies dancing, eight maids a-milking, seven swans a-swimming,
six geese a-laying, five gold rings;
four calling birds, three French hens, two turtle doves
and a partridge in a pear tree.
/* primenum.c */

#define BeginProgram void main(int argc, char *argv[])
#define CloseBrace }
#define CommandLineArgument -1
#define Declare int i,j,n,Flag=1;
#define EndOfProgram return;
#define False 0;
#define ForLoop ;for
#define GetCommandLineArgument n=atoi(argv[1]);
#define i F1ag
#define If if
#define Increment ++
#define Is ==
#define LessThan *(c&64)*
#define LessThanOrEqualTo !=
#define Modulo %
#define OpenBrace {
#define PossibleFactor j
#define PossiblePrime i
#define PossiblePrime (c=getchar())
#define PrimeNumber (c^(!i*n%64));
#define Print putchar
#define SetTo =
#define SmallestPrime 2
#define True 1
#define Variables char c;
#define Zero i%j

BeginProgram
OpenBrace
Declare Variables
GetCommandLineArgument

ForLoop (PossiblePrime SetTo SmallestPrime ;
    PossiblePrime LessThanOrEqualTo CommandLineArgument;
    Increment PossiblePrime)
OpenBrace
    Flag SetTo True
    ForLoop (PossibleFactor SetTo SmallestPrime ;
        PossibleFactor LessThan PossiblePrime ;
        Increment PossibleFactor)
        If (PossiblePrime Modulo PossibleFactor
            Is Zero)
            Flag SetTo False
    
If (Flag Is True)
    Print PrimeNumber

CloseBrace
CloseBrace
EndOfProgram
> echo "Hallo" | ./primenum.exe 10
Bkffe

> echo "Hallo" | ./primenum.exe 10 | ./primenum.exe 10
Hallo

> echo "THIS IS ALL IN CAPS" | ./primenum.exe 32
this is all in caps
rtx
expand_expr_real (tree exp, rtx target, enum machine_mode tmode,
        enum expand_modifer modifier, rtx *alt_rtl)
{
  rtx op0, op1, temp;
  tree type = TREE_TYPE (exp);
  int unsignedp = TREE_UNSIGNED (type);
  enum machine_mode mode;
  enum tree_code code = TREE_CODE (exp);
  optab this_optab;
  rtx subtarget, original_target;
  int ignore;
  tree context;

  /* Handle ERROR_MARK before anybody tries to access its type. */
  if (TREE_CODE (exp) == ERROR_MARK || TREE_CODE (type) == ERROR_MARK)
  {
    op0 = CONST0_RTX (tmode);
    if (op0 != 0)
      return op0;
    return const0_rtx;
  }
  mode = TYPE_MODE (type);
  /* Use subtarget as the target for operand 0 of a binary operation. */
  subtarget = get_subtarget (target);
  original_target = target;
  ignore = (target == const0_rtx
        || ((code == NON_LVALUE_EXPR || code == NOP_EXPR
        || code == CONVERT_EXPR || code == REFERENCE_EXPR
        || code == COND_EXPR || code == VIEW_CONVERT_EXPR)
        && TREE_CODE (type) == VOID_TYPE));

  /* If we are going to ignore this result, we need only do something
   if there is a side-effect somewhere in the expression. If there
   is, short-circuit the most common cases here. Note that we must
   not call expand_expr with anything but const0_rtx in case this
   is an initial expansion of a size that contains a PLACEHOLDER_EXPR. */
  if (ignore)
  {
    if (! TREE_SIDE_EFFECTS (exp))
      return const0_rtx;
    /* Ensure we reference a volatile object even if value is ignored, but
     don't do this if all we are doing is taking its address. */
    if (TREE_THIS_VOLATILE (exp)
        && TREE_CODE (exp) != FUNCTION_DECL
        && mode != VOIDmode && mode != BLKmode
        && modifier != EXPAND CONST_ADDRESS)
    {
      temp = expand_expr (exp, NULL_RTX, VOIDmode, modifier);
      if (GET_CODE (temp) == MEM)
        temp = copy_to_reg (temp);
      return const0_rtx;
  }
The fellow who designed it,
Is working far away;
The spec’s not been updated,
For many a livelong day.

The guy who implemented it is
Promoted up the line;
And some of the
enhancements
Didn’t match to the design.

They haven‘t kept the flowcharts,
The manual‘s a mess,
And most of what you need to
know
You‘ll simply have to guess.

We do not know the reason,
Why the bugs pour in like rain,
But don‘t just stand here gaping,
Get out there and MAINTAIN.

David H. Diamond
Kognitive Dissonanz: Beispiel

- Sachverhalt: Die Software funktioniert nicht so, wie sie sollte
- Wahrnehmung Programmierer: „Ich bin mit meiner Arbeit zufrieden. Ich finde mich gut.“ (ignoriert)
- Wahrnehmung QA: „Das funktioniert nicht.“
- Feedback QA: „In ihrem Programm ist ein Bug“
- -> kognitive Dissonanz: „Ich bin gut – Du hast Mist gebaut“
- Auflösung:
  - „Der Bug liegt sicher nicht bei mir! Fragen sie mal Kollegen X.“
  - „Es steht aber genau so in den Anforderungen! Lesen sie die mall!“
  - „Das ist kein Bug, die User haben es mir so erklärt.“
  - „Sie wissen ja nicht einmal, was ein Bug ist!“
- Kognitive Dissonanz führt also zu einer Wirklichkeitskonstruktion, die versucht, die Dissonanz aufzulösen

Beispiel für den psycho-sozialen Aspekt

- aus Petr Kroha „Softwaretechnologie“ [Kroha97]
- Windows of Opportunity [Yourdon92] - beschreiben günstigsten Zeitpunkt zur Neuentwicklung eines Softwaremoduls:
  - „der zuständige Programmierer geht in Rente oder kündigt
  - wenn die Komponente extrem gravierende Fehler aufweist und der Programmierer die Suche aufgegeben hat
  - wenn es endlich möglich ist, den Widerstand leistenden Programmierer zu entlassen. “
4.1 Writing Robust Programs

Avoid arbitrary limits on the length or number of any data structure, including file names, lines, files, and symbols, by allocating all data structures dynamically. In most Unix utilities, "long lines are silently truncated". This is not acceptable in a GNU utility. …
Kap 3.4. Change-Management
Figure 1-1. Software Reengineering Assessment (SRA) Process Overview
<table>
<thead>
<tr>
<th>Candidate Software Name &amp; Description:</th>
<th>Number of Questions Answered</th>
<th>Summation of Answers</th>
<th>Average: (Summation Divided by Number of Questions)</th>
<th>Indication of Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remaining Life (&gt; 3 yrs?)</td>
<td></td>
<td></td>
<td></td>
<td>yes no</td>
</tr>
<tr>
<td>Age (&gt; 5 yrs?)</td>
<td></td>
<td></td>
<td></td>
<td>yes no</td>
</tr>
<tr>
<td>Preparation¹</td>
<td></td>
<td></td>
<td></td>
<td>yes no</td>
</tr>
<tr>
<td>Importance¹</td>
<td></td>
<td></td>
<td></td>
<td>yes no</td>
</tr>
<tr>
<td>Maintenance Environment²</td>
<td></td>
<td></td>
<td></td>
<td>yes no</td>
</tr>
<tr>
<td>Redocument²</td>
<td></td>
<td></td>
<td></td>
<td>yes maybe no</td>
</tr>
<tr>
<td>Restructure²</td>
<td></td>
<td></td>
<td></td>
<td>yes maybe no</td>
</tr>
<tr>
<td>Translate Source Code²</td>
<td></td>
<td></td>
<td></td>
<td>yes maybe no</td>
</tr>
<tr>
<td>Data Reengineer²</td>
<td></td>
<td></td>
<td></td>
<td>yes maybe no</td>
</tr>
<tr>
<td>Retarget²</td>
<td></td>
<td></td>
<td></td>
<td>yes maybe no</td>
</tr>
<tr>
<td>Reverse Engineer</td>
<td></td>
<td></td>
<td></td>
<td>yes no</td>
</tr>
<tr>
<td>Redevelop</td>
<td></td>
<td></td>
<td></td>
<td>yes no</td>
</tr>
<tr>
<td>Status Quo</td>
<td></td>
<td></td>
<td></td>
<td>yes no</td>
</tr>
</tbody>
</table>
\[
\text{NPV} = \text{PV} (\text{Benefit}) - \text{PV} (\text{Investment}) = \text{PV}(O&S_1) - \text{PV}(O&S_2) - \text{PV}(\text{Investment}_2)
\]

\[
\text{BIR} = \text{PV} (\text{Benefit}) / \text{PV} (\text{Investment}) = (\text{PV}(O&S_1) - \text{PV}(O&S_2)) / \text{PV} (\text{Investment}_2)
\]

- \( > 0 \): Positive Return (Discounted $)
- \( < 0 \): Negative Return (Discounted $)
- \( = 0 \): Break-even (Current $)

**Figure 5.3. Net Present Value (NPV), Benefit Investment Ratio (BIR), and Break-even Point (BP)**
Initial Ground Rules and Assumptions:

☐ All estimates and comparisons will be made in discounted constant FY ___ dollars. (Fill in the blank to specify what fiscal year the dollars estimated represent.)

☐ Use the same cost element structure (CES) for all candidate strategies.

☐ The economic "clock" (or time = 0) for any candidate strategy begins when the strategy effort starts.

☐ Sunk costs, occurring prior to the start of the effort, will not be considered.

☐ A standard labor rate of $_______/man-month (or $_______/man-year) will be used for Government, and a standard labor rate of $_______/man-month (or $_______/man-year) will be used for contractors.

☐ Strategy 1 is reserved for the "status quo" candidate strategy.

☐ Operations & Support (O&S) cost for all candidate strategies includes O&S of the existing software during reengineering.

☐ The __________ software estimating model will be used for all estimates of reengineering and support efforts. Overall assessment (maturity, level of risk in using them, and differences between models), experience with, and track record using each model are/are not attached (circle are or are not).

☐ If code translation is indicated by the technical assessment, automated code translation will/will not be used to reduce the level of manual translation that must be performed (circle will or will not).

☐ For automated code translation, it will be assumed, that ___% of the code will be successfully translated by the automated tool based on the tool developer's specifications.

☐ If translation to Ada and restructuring are both indicated, the code will first be translated to Ada and then restructured using the Ada Programming Support Environment.

☐ The system documentation produced during the effort will be consistent with (full/tailored) (MIL-STD-xxx).

☐ To the maximum extent possible, automated CASE tools will be used to facilitate the effort. This includes the following tools: ..........

☐ The staff that is currently maintaining the software will/will not be the same staff that performs the effort. If not, the __________ will perform the work.

☐ The effort will/will not include IV&V.
Kap 4.2. Clone-Detection

Beispiel 1: identischer Code

```java
class Permutation {
    private final int[] path;

    public Permutation(int numNodes) {
        path = new int[numNodes];
    }

    public int get(int index) {
        return path[index]
    }
}
```

```java
private int pathDistance(Permutation path) {
    int result = 0;
    for (int i = 0; i < numNodes; i++) {
        int from = path.get(i);
        int to = path.get((i+1) % numNodes);
        result += distance[from][to];
    }
    return result;
}
```

```java
private double pathAvgDistance(Permutation path) {
    int result = 0;
    for (int i = 0; i < numNodes; i++) {
        int from = path.get(i);
        int to = path.get((i+1) % numNodes);
        result += distance[from][to];
    }
    return (double) result / numNodes;
}
```

Beispiel 2: identisch bis auf Identifikatoren

```java
int r = 0;
for (int j = 0; j < numNodes; j++) {
    int f = path.get(j);
    int t = path.get((j+1) % numNodes);
    r += distance[from][to];
}
```

Beispiel 3: semantisch äquivalent (?)

```java
private int pathDistance(Permutation path, int i, int r) {
    if (i >= numNodes)
        return r;
    else
        return pathDistance(path, i+1,
            distance[path.get(i)][ path.get((i+1) % numNodes] + r)
}
```
<table>
<thead>
<tr>
<th>FreeBSD 4.0</th>
<th>Linux 2.4.0</th>
<th>NetBSD 1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1e+006</td>
<td>2e+006</td>
</tr>
<tr>
<td>1e+006</td>
<td>3e+006</td>
<td>4e+006</td>
</tr>
<tr>
<td>2e+006</td>
<td>5e+006</td>
<td>6e+006</td>
</tr>
<tr>
<td>3e+006</td>
<td>6e+006</td>
<td>7e+006</td>
</tr>
<tr>
<td>4e+006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5e+006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6e+006</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7e+006</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Case Study: Saladon (CRM)

Date: Chuckheit, Architect

„Most of the drop in the source code graph (core) is due to clone detection removal."

- Total LOC reduction over time: ~40%
- CloneDR runs every quarter; active clone remediation by development team
- Java Application: Core + 6 Customer Variants
<table>
<thead>
<tr>
<th>Application</th>
<th>Language</th>
<th>SLOC</th>
<th>Removable SLOC (est.)</th>
<th>Removed SLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Control</td>
<td>C</td>
<td>427921</td>
<td>54346 12.7%</td>
<td>N.A. (pre-removal capability for C 6.9%)</td>
</tr>
<tr>
<td>PARLANE Compiler</td>
<td>C</td>
<td>42413</td>
<td>4133 9.7%</td>
<td>2568 6.0%</td>
</tr>
<tr>
<td>CCC Registration</td>
<td>COBOL85</td>
<td>77631</td>
<td>15500(39967) 20.0 (51.5%)</td>
<td>15034 19.9%</td>
</tr>
<tr>
<td>Roccade Accounting</td>
<td>IBM COBOL</td>
<td>567798</td>
<td>130494 23%</td>
<td>N.A. (not yet performed)</td>
</tr>
<tr>
<td>SLC Insurance</td>
<td>PROGRESS</td>
<td>356413</td>
<td>58783 16.5%</td>
<td>N.A. (not yet performed)</td>
</tr>
<tr>
<td>SWING Toolkit</td>
<td>Java</td>
<td>241460</td>
<td>26355 10.9%</td>
<td>N.A. (not yet performed)</td>
</tr>
</tbody>
</table>

```c
#define clone27(a,b,c,d)
for (a=1,a<8,a++)
for (c=a,c<3,c--)
if (d[c]>d[c-1])
swap(d[a],d[c]);

... code block 1 ...
```

```c
for (a=1, a<8, a++)
for (c=a, c<3, c--)
if (d[c]>d[c-1])
swap(d[a],d[c]);
```

```
... code block 2 ...
```

```c
for (a=1, a<10, a++)
for (j=1, j<10, j--)
if (A[j]>A[j-1])
swap(A[j],A[j-1]);
```

```
... code block 3 ...
```

```c
for (i=1, i<2*Q, i++)
for (j=1, j<1, j--)
// exchange if less
if (K[i]>K[i-1])
swap(K[i],K[i-1]);
```

```
... code block 4 ...
```

```c
for (i=1, i<1000, i++)
for (j=1, j<1, j--)
if (B[j]>O[j-1])
swap(B[i],B[j]);
```

```
... code block 4 ...
```
FILESTATS View Options Help

file1.c
file2.c
file3.c
file4.c
file5.c
file6.c
file7.c

file8.a

file9.c
file10.c
file11.c
file12.c
file13.c
file14.c
file15.c

file16.c
file17.c
file18.c
file19.c

file20.c
file21.c
file22.c
file23.c
file24.c
file25.c
file26.c

file27.c
file28.c
file29.c
file30.c

file31.c
file32.c
file33.c
file34.c
file35.c
file36.c
file37.c

file134de340

stats 292/294
lines 1637/12031
files 39/39

/* Set the key in relation IS_LINE */
is_line.Is_module = brcs.ptr->Module;
is_line.fnr_id< = brcs.ptr->fn_i.+id; /* Access the relation IS_LINE */
/* Set RTval for RTERRASSRT/RTDCF message */
RTval = DBHeader(FLIS_LINE, &is_line);
if (RTval != GLSUCCESS)
{
    /* Assert and print DCF */
    RTERRASSRT(RTOBYPASS, RT_STATIC_ RTDCF, (long) &is_line);
    /* Print the error data */
}
Kap 4.4. Software-Dokumentation
(Visconti 2000) Umfrage zum Reifegrad von Dokumentationsprozessen;
Skala von „Very Low“ (VL) bis „Very High“ (VH).

<table>
<thead>
<tr>
<th>Key Practices</th>
<th>VL</th>
<th>L</th>
<th>M</th>
<th>H</th>
<th>VH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Creation of basic software documents.</td>
<td>0%</td>
<td>8%</td>
<td>25%</td>
<td>17%</td>
<td>50%</td>
</tr>
<tr>
<td>2. Management recognition of importance of documentation</td>
<td>0%</td>
<td>8%</td>
<td>25%</td>
<td>0%</td>
<td>67%</td>
</tr>
<tr>
<td>3. <strong>Existence of documentation policy or standard</strong></td>
<td>17%</td>
<td>8%</td>
<td>25%</td>
<td>0%</td>
<td>67%</td>
</tr>
<tr>
<td>4. <strong>Monitor implementation of policy or standard</strong></td>
<td>8%</td>
<td>8%</td>
<td>42%</td>
<td>25%</td>
<td>17%</td>
</tr>
<tr>
<td>5. Existence of a defined process for creation of documents</td>
<td>25%</td>
<td>0%</td>
<td>33%</td>
<td>33%</td>
<td>8%</td>
</tr>
<tr>
<td>6. Methods to assure quality of documentation</td>
<td>8%</td>
<td>25%</td>
<td>42%</td>
<td>25%</td>
<td>0%</td>
</tr>
<tr>
<td>7. Assessments of usability of documentation</td>
<td>8%</td>
<td>67%</td>
<td>h</td>
<td>18%</td>
<td>0%</td>
</tr>
<tr>
<td>8. Definition of software documentation quality and usability measures</td>
<td>67%</td>
<td>17%</td>
<td>8%</td>
<td>8%</td>
<td>0%</td>
</tr>
<tr>
<td>9. Collection and analysis of documentation quality measures</td>
<td>42%</td>
<td>42%</td>
<td>17%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>10. Collection and analysis of documentation usability measures</td>
<td>100%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>11. <strong>Process improvement feedback loop</strong></td>
<td>42%</td>
<td>33%</td>
<td>17%</td>
<td>0%</td>
<td>8%</td>
</tr>
</tbody>
</table>

I believe that the time is ripe for significantly better documentation of programs, and that we can best achieve this by considering programs to be works of literature. Hence, my title: "Literate Programming."
– D. Knuth. "Literate Programming"
6. An XString constructed from a standard string needs space to hold the characters:

(XString members and friends 6) \equiv

\begin{verbatim}
Xstring::Xstring(const char *s)
{
    p = new srep;
    (Allocate space for the string and put a copy of s there 7);
}
\end{verbatim}

See also sections 9, 10, 13, and 17.
This code is used in section 3.

7. There is always the possibility that a client will try something like “XString x = \Lambda.” We substitute the null string whenever we are given a null pointer.

(Allocate space for the string and put a copy of s there 7) \equiv

\begin{verbatim}
if (s \equiv \Lambda) s = "";
p = new char [strlen(s) + 1];
strcpy(p->s, s);
\end{verbatim}

This code is used in sections 6 and 13.
/**
 * Returns <tt>true</tt> if this list contains the specified element.
 * @param elem element whose presence in this List is to be tested.
 * @return <tt>true</tt> if the specified element is present;
 *         <tt>false</tt> otherwise.
 */

public boolean contains(Object elem) {
    return indexOf(elem) >= 0;
}

contains

public boolean contains(Object elem)

    Returns true if this list contains the specified element.

Specified by:
    contains in interface List
Overrides:
    contains in class AbstractCollection
Parameters:
    elem - element whose presence in this List is to be tested.
Returns:
    true if the specified element is present, false otherwise.
Klausurtermine
ca. 90 Minuten

• Mittwoch 28.07.04 10:15
• Freitag 30.07.04
• Mittwoch 11.08.04 10:15
• **Freitag 20.08.04 11:00**
• Mittwoch 29.09.04
• Freitag 01.10.04