Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Кафедра интеллектуальных информационных технологий

ОТЧЁТ

по лабораторной работе №1

по курсу «Логические основы интеллектуальных систем»

|  |  |
| --- | --- |
| Проверил:  Ивашенко В.П. | Выполнила:  студент гр. 121701  Волкова О. А. |

Минск, 2014

Для выполнения лабораторной работы, был выполнен набор шагов, отчет о прохождении которых представлен в данном отчете. При демонстрации результатов выполнения программы, из лога была исключена информация, не имеющая отношение к алгоритму выполнения программы (autosegment, process, unlink и проч.).

*1. Разархивировать fs\_repo.lib.rar в пустую папку.*

*2. Запустить консоль командного процессора, перейти в папку с разархивированным fs\_repo.lib.rar.*

*3. Выполнить команду clean для очистки репозитория.*

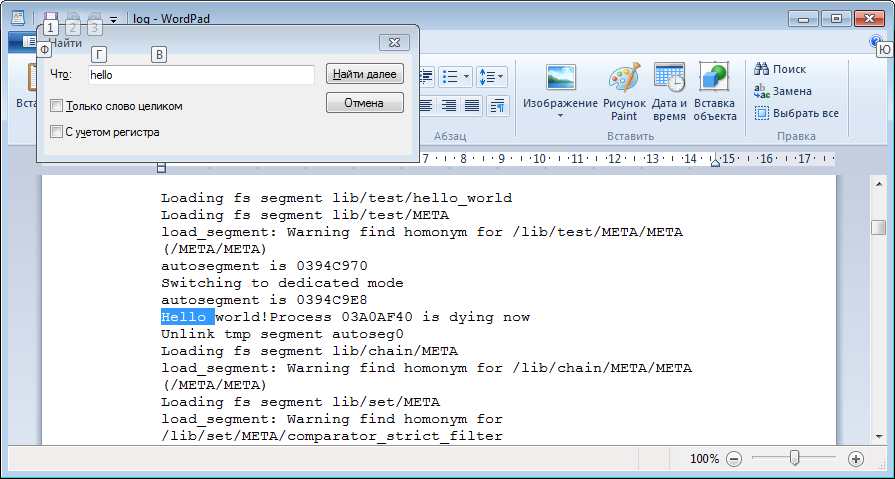
*4. Выполнить команду place для трансляции исходных текстов и размещения базы знаний в репозитории.*

*5. Выполнить команду start cmd.*

*6. В новом окне командного процессора выполнить команду run -d >log*

*7. Открыть файл log, убедиться, что тестовые процедуры отработали успешно, без ошибок.*

*8. Закрыть последнее окно командного процессора.*



*9. В файле /lib/test/main.m4scp вызовы тестовых процедур, кроме hello\_world, заключить в макрос DISABLE([ ... ]).*

*10. Повторить шаги 2-8.*

*11. Скопировать тестовую процедуру hello\_world в новый файл (/lib/test/reverse.m4scp) и соответственно её переименовать.*

*12. Добавить вызов в main.m4scp.*

*13. Проверить, повторяя шаги 2-8.*

*14. Реализовать в тестовой процедуре подготовку тестовых данных с помощью операторов генерации (genEl, genElStr3).*

*15. Распечатать тестовые данные, используя оператор printEl - аналог оператора print.*

*16. Проверить, повторяя шаги 2-8.*

*17. Реализовать в тестовой процедуре удаление всех тестовых данных с помощью операторов удаления (eraseEl).*

*18. Создать пустой узел протоколирования, для передачи в протоколирующие процедуры (logger), добавить оператор его удаления по завершении тестовой процедуры (eraseEl).*

*19. Реализовать формирование цепочки синглетонов тестовых данных, используя оператор call для вызова процедур push\_first\_singlet, push\_singlet.*

**Текст программы:**

[{

setA, setB,

undfA, undfB,

arcA, arcB,

chain\_top\_link, meta\_chain\_link, link,

logger, descr, node, empty

}],

{[]})

genEl([1\_: assign\_: node\_: const\_: setA])

genEl([1\_: assign\_: node\_: const\_: setB])

genEl([1\_: assign\_: node\_: const\_: logger])

genEl([1\_: assign\_: node\_: const\_: empty])

genElStr3([

1\_: fixed\_: setA,

2\_: assign\_: arc\_: permanent\_: pos\_: const\_: arcA,

3\_: assign\_: undf\_: const\_: undfA])

genElStr3([

1\_: fixed\_: setB,

2\_: assign\_: arc\_: permanent\_: pos\_: const\_: arcB,

3\_: assign\_: undf\_: const\_: undfB])

/\* Create chain and push first singleton\*/

call([1\_: fixed\_: chain\_push\_first\_singlet, 2\_: fixed\_: prm1 = [{1\_: setA, 2\_: chain\_top\_link, 3\_: logger}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

/\*Push second singleton to chain\*/

call([1\_: fixed\_: chain\_push\_singlet, 2\_: fixed\_: prm3 = [{1\_: chain\_top\_link, 2\_: setB, 3\_: chain\_top\_link, 4\_: logger}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

M4SCS(

printEl([1\_: fixed\_: setA])

printEl([1\_: fixed\_: setB])

)

label(erase)

eraseEl([1\_: fixed\_: arcA])

eraseEl([1\_: fixed\_: arcB])

eraseEl([1\_: fixed\_: setA])

eraseEl([1\_: fixed\_: setB])

eraseEl([1\_: fixed\_: undfA])

eraseEl([1\_: fixed\_: undfB])

eraseEl([1\_: fixed\_: logger])

eraseEl([1\_: fixed\_: empty])

**Результат выполнения программы:**

chain\_push\_first\_singlet: entering

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_push\_first\_singlet: leaving

chain\_push\_singlet: entering

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_push\_singlet: leaving

printEl: node|const:/tmp/autoseg/@@5bf1bda6-16d6-4b62-8db1-27969224b8b2@9

Output arcs:

Input arcs:

printEl: node|const:/tmp/autoseg/@@5bf1bda6-16d6-4b62-8db1-27969224b8b2@7

Output arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@5bf1bda6-16d6-4b62-8db1-27969224b8b2@b >- undf|const:/tmp/autoseg/@@5bf1bda6-16d6-4b62-8db1-27969224b8b2@a

Input arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@5bf1bda6-16d6-4b62-8db1-27969224b8b2@15 -< node|const:/tmp/autoseg/@@5bf1bda6-16d6-4b62-8db1-27969224b8b2@11

printEl: node|const:/tmp/autoseg/@@5bf1bda6-16d6-4b62-8db1-27969224b8b2@8

Output arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@5bf1bda6-16d6-4b62-8db1-27969224b8b2@d >- undf|const:/tmp/autoseg/@@5bf1bda6-16d6-4b62-8db1-27969224b8b2@c

Input arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@5bf1bda6-16d6-4b62-8db1-27969224b8b2@22 -< node|const:/tmp/autoseg/@@5bf1bda6-16d6-4b62-8db1-27969224b8b2@1e

*20. Реализовать получение и распечатку тестовых данных из цепочки, используя операторы printEl, ifVarAssign и оператор call для вызова процедуры get\_next\_singlet.*

*21. Проверить, повторяя шаги 2-8.*

**Текст программы:**

label(print\_chain1)

call([1\_: fixed\_: chain\_get\_next\_singlet, 2\_: fixed\_: prmA = [{1\_: chain\_top\_link, 2\_: link, 3\_: node}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

ifVarAssign([1\_: node],,destroy)

label(print\_element1)

M4SCS(printEl([1\_: fixed\_: node]))

/\*Print other singletons in a chain\*/

call([1\_: fixed\_: chain\_get\_next\_singlet, 2\_: fixed\_: prmB = [{1\_: link, 2\_: link, 3\_: node}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

ifVarAssign([1\_: node],print\_element1,destroy)

label(destroy)

ifVarAssign([1\_: chain\_top\_link],,erase)

/\*Check if chain not empty\*/

call([1\_: fixed\_: chain\_get\_next\_singlet, 2\_: fixed\_: prm7 = [{1\_: chain\_top\_link, 2\_: link, 3\_: node}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

ifVarAssign([1\_: node],,destroy\_chain)

/\*Delete all chain elements\*/

call([1\_: fixed\_: chain\_pop\_destroy\_singlet, 2\_: fixed\_: prm5 = [{1\_: chain\_top\_link, 2\_: chain\_top\_link, 3\_: logger}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

ifVarAssign([1\_: chain\_top\_link],destroy,)

label(destroy\_chain)

/\*Destroy empty chain\*/

call([ 1\_: fixed\_: chain\_destroy, 2\_: fixed\_: prm6 = [{1\_: chain\_top\_link, 2\_: logger}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

label(erase)

eraseEl([1\_: fixed\_: logger])

eraseEl([1\_: fixed\_: empty])

eraseEl([1\_: fixed\_: empty\_chain])

**Результат выполнения программы:**

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

printEl: node|const:/tmp/autoseg/@@0322b660-6be3-4c19-b860-8b05746620dd@8

Output arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@0322b660-6be3-4c19-b860-8b05746620dd@d >- undf|const:/tmp/autoseg/@@0322b660-6be3-4c19-b860-8b05746620dd@c

Input arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@0322b660-6be3-4c19-b860-8b05746620dd@22 -< node|const:/tmp/autoseg/@@0322b660-6be3-4c19-b860-8b05746620dd@1e

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

printEl: node|const:/tmp/autoseg/@@0322b660-6be3-4c19-b860-8b05746620dd@7

Output arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@0322b660-6be3-4c19-b860-8b05746620dd@b >- undf|const:/tmp/autoseg/@@0322b660-6be3-4c19-b860-8b05746620dd@a

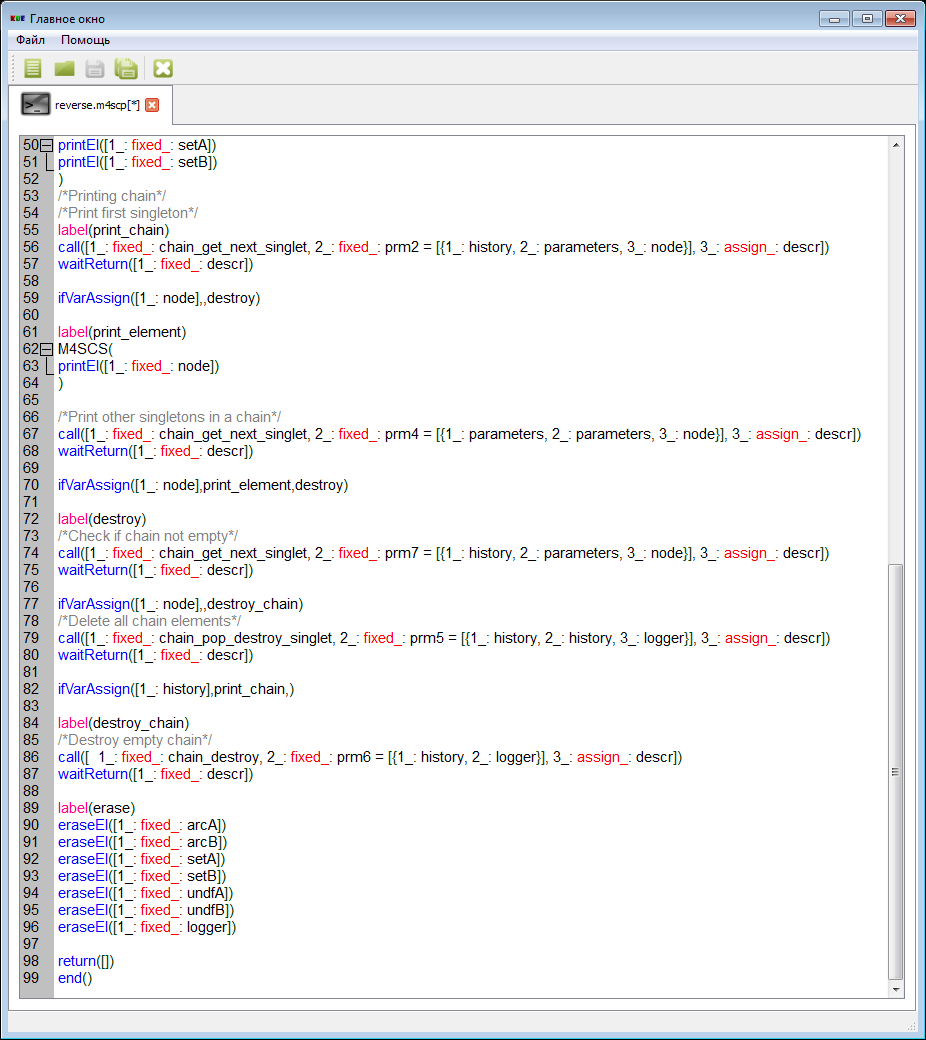
Input arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@0322b660-6be3-4c19-b860-8b05746620dd@15 -< node|const:/tmp/autoseg/@@0322b660-6be3-4c19-b860-8b05746620dd@11

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

*22. Реализовать удаление данных из цепочки и самой цепочки, используя оператор ifVarAssign и оператор call для вызова процедур pop\_destroy\_singlet и destroy.*



**Результат выполнения программы:**

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

chain\_pop\_destroy\_singlet: entering

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_pop\_destroy\_singlet: leaving

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

printEl: node|const:/tmp/autoseg/@@c25b81c2-bb0a-43ae-8f6f-8858a903d9d8@7

Output arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@c25b81c2-bb0a-43ae-8f6f-8858a903d9d8@b >- undf|const:/tmp/autoseg/@@c25b81c2-bb0a-43ae-8f6f-8858a903d9d8@a

Input arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@c25b81c2-bb0a-43ae-8f6f-8858a903d9d8@15 -< node|const:/tmp/autoseg/@@c25b81c2-bb0a-43ae-8f6f-8858a903d9d8@11

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

chain\_pop\_destroy\_singlet: entering

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_pop\_destroy\_singlet: leaving

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

chain\_destroy: entering

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_destroy: leaving

*23. Реализовать подпрограмму - процедуру построения перевёрнутой цепочки (/lib/chain/reverse/chain\_reverse)*

*24. Добавить вызов реализованной процедуры (/lib/chain/reverse/chain\_reverse) в тестовую процедуру.*

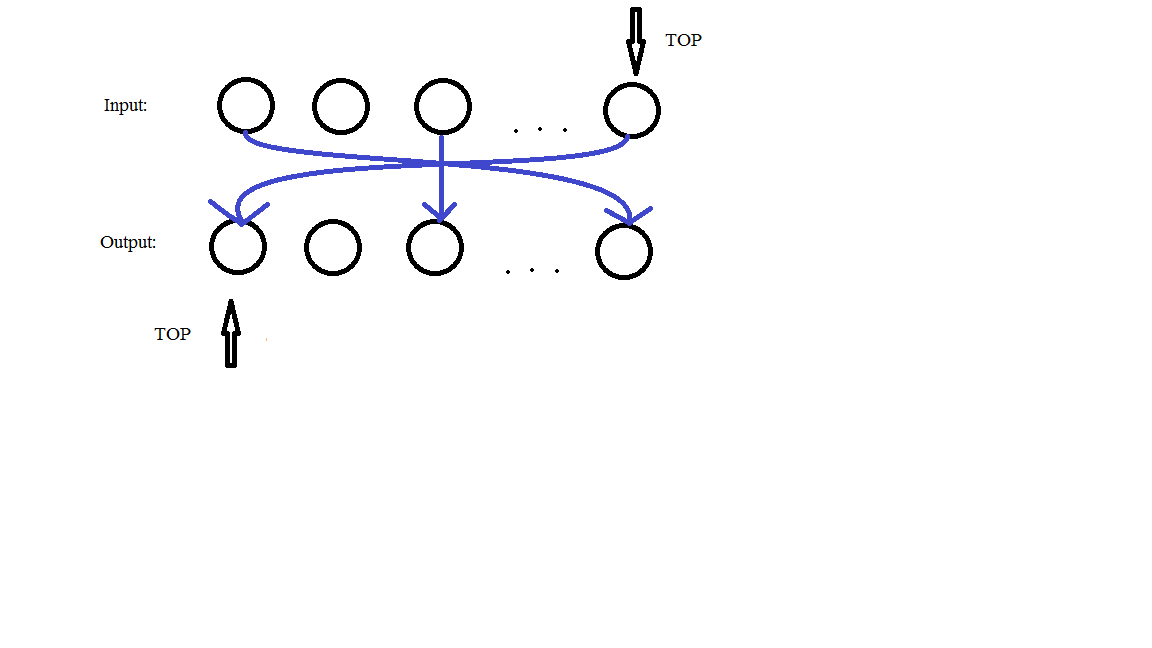
*25. Отладить и протестировать /lib/chain/reverse/chain\_reverse.*

**Алгоритм построения перевернутой цепочки:**

На вход подпрограммы приходит метацепочка history, в которой первым элементом является цепочка, которую необходимо перевернуть. Создаем новую цепочку, которая в конце выполнения программы стент перевернутой цепочкой синглетонов, относительно исходной. При помощи готовых подпрограмм манипулирования цепочкой синглетонов, происходит следующий алгоритм:

1. Читаем первый элемент из исходной цепочки.
2. Удаляем прочитанный элемент из цепочки.
3. Добавляем прочитанный элемент в новую цепочку.
4. Если исходная цепочка не пустая, переходим к пункту 1.

Визуально алгоритм перевернутой цепочки представлен на рисунке.



**Текст программы /lib/chain/reverse/chain\_reverse:**

{[

1\_: in\_: out\_: state,

2\_: in\_: out\_: true,

3\_: in\_: out\_: history,

4\_: in\_: parameters,

5\_: in\_: logging

]})

DBG(M4SCS(print([1\_: fixed\_: enter\_message])))

genEl([1\_: assign\_: node\_: const\_: empty])

/\* Извлекаем из метацепочки цепочку\*/

call([1\_: fixed\_: chain\_get\_next\_singlet, 2\_: fixed\_: prm1 = [{1\_: history, 2\_: link, 3\_: old\_chain}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

ifVarAssign([1\_: old\_chain],,destroy\_final)

/\*Извлекаем первый элемент цепочки\*/

call([1\_: fixed\_: chain\_get\_next\_singlet, 2\_: fixed\_: prm2 = [{1\_: old\_chain, 2\_: link, 3\_: node}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

ifVarAssign([1\_: node],,empty\_chain)

call([1\_: fixed\_: chain\_pop\_destroy\_singlet, 2\_: fixed\_: prm3 = [{1\_: old\_chain, 2\_: old\_chain, 3\_: logging}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

call([1\_: fixed\_: chain\_push\_first\_singlet, 2\_: fixed\_: prm4 = [{1\_: node, 2\_: new\_chain, 3\_: logging}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

label(next)

/\*Извлекаем первый элемент цепочки\*/

call([1\_: fixed\_: chain\_get\_next\_singlet, 2\_: fixed\_: prm5 = [{1\_: old\_chain, 2\_: link, 3\_: node}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

ifVarAssign([1\_: node],,push)

/\*Удаляем извлеченный элемент из исходной цепочи\*/

call([1\_: fixed\_: chain\_pop\_destroy\_singlet, 2\_: fixed\_: prm6 = [{1\_: old\_chain, 2\_: old\_chain, 3\_: logging}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

/\* Добавляем прочитанный элемент в новую цепочку \*/

сall([1\_: fixed\_: chain\_push\_singlet, 2\_: fixed\_: prm7 = [{1\_: new\_chain, 2\_: node, 3\_: new\_chain, 4\_: logging}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr],next,next)

/\*Если цепочка пустая, то добавляем ее в историю и выходим из программы\*/

label(empty\_chain)

call([1\_: fixed\_: chain\_push\_singlet, 2\_: fixed\_: prm8 = [{1\_: history, 2\_: old\_chain, 3\_: history, 4\_: logging}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr],destroy\_final,destroy\_final)

label(push)

/\* Если исходная цепочка пустая – включаем новую цепочку в метацепочку, удаляем исходную и выходим из программы \*/

call([1\_: fixed\_: chain\_push\_singlet, 2\_: fixed\_: prm8 = [{1\_: history, 2\_: new\_chain, 3\_: history, 4\_: logging}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

label(destroy)

call([ 1\_: fixed\_: chain\_destroy, 2\_: fixed\_: prm9 = [{1\_: old\_chain, 2\_: logging}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

label(destroy\_final)

eraseEl([1\_: fixed\_: f\_: empty])

DBG(M4SCS(print([1\_: fixed\_: leave\_message])))

return()

end()

**Текст программы /lib/test/reverse:**

/\*Reverse chain\*/

/\* Создаем метацепочку и помещаем в нее цепочку, которую необходимо удалить\*/

call([1\_: fixed\_: chain\_push\_first\_singlet, 2\_: fixed\_: prmD = [{1\_: chain\_top\_link, 2\_: meta\_chain\_link, 3\_: logger}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

/\* Переворачиваем цепочку\*/

call([1\_: fixed\_: chain\_reverse, 2\_: fixed\_: prm9 = [{1\_: empty, 2\_: empty, 3\_: meta\_chain\_link, 4\_: empty, 5\_: logger}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

/\* Читаем перевернутую цепочку\*/

call([1\_: fixed\_: chain\_get\_next\_singlet, 2\_: fixed\_: prmC = [{1\_: meta\_chain\_link, 2\_: link, 3\_: chain\_top\_link}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

/\* Распечатываем перевернутую цепочку\*/

label(print\_chain1)

call([1\_: fixed\_: chain\_get\_next\_singlet, 2\_: fixed\_: prmA = [{1\_: chain\_top\_link, 2\_: link, 3\_: node}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

ifVarAssign([1\_: node],,destroy)

label(print\_element1)

M4SCS(

printEl([1\_: fixed\_: node])

)

/\*Print other singletons in a chain\*/

call([1\_: fixed\_: chain\_get\_next\_singlet, 2\_: fixed\_: prmB = [{1\_: link, 2\_: link, 3\_: node}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

ifVarAssign([1\_: node],print\_element1,destroy)

label(destroy)

ifVarAssign([1\_: chain\_top\_link],,erase)

/\*Check if chain not empty\*/

call([1\_: fixed\_: chain\_get\_next\_singlet, 2\_: fixed\_: prm7 = [{1\_: chain\_top\_link, 2\_: link, 3\_: node}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

ifVarAssign([1\_: node],,destroy\_chain)

/\*Delete all chain elements\*/

call([1\_: fixed\_: chain\_pop\_destroy\_singlet, 2\_: fixed\_: prm5 = [{1\_: chain\_top\_link, 2\_: chain\_top\_link, 3\_: logger}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

ifVarAssign([1\_: chain\_top\_link],destroy,)

label(destroy\_chain)

/\*Destroy empty chain\*/

call([ 1\_: fixed\_: chain\_destroy, 2\_: fixed\_: prm6 = [{1\_: chain\_top\_link, 2\_: logger}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

**Результат выполнения программы:**

chain\_push\_singlet: entering

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_push\_singlet: leaving

/\* Распечатываем элементы, которые будут добавлены в цепочку\*/

printEl: node|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@7

Output arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@c >- undf|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@b

Input arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@16 -< node|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@12

printEl: node|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@8

Output arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@e >- undf|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@d

Input arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@23 -< node|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@1f

/\* Распечатываем цепочку и видим: первый элемент №8, второй - №7\*/

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

printEl: node|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@8

Output arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@e >- undf|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@d

Input arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@23 -< node|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@1f

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

printEl: node|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@7

Output arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@c >- undf|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@b

Input arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@16 -< node|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@12

/\* Создание метацепочки\*/

chain\_push\_first\_singlet: entering

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_push\_first\_singlet: leaving

chain\_reverse: entering

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

chain\_pop\_destroy\_singlet: entering

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_pop\_destroy\_singlet: leaving

chain\_push\_first\_singlet: entering

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_push\_first\_singlet: leaving

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

chain\_pop\_destroy\_singlet: entering

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_pop\_destroy\_singlet: leaving

chain\_push\_singlet: entering

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_push\_singlet: leaving

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

chain\_push\_singlet: entering

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_push\_singlet: leaving

chain\_destroy: entering

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_get\_next: entering

chain\_get\_next: leaving

chain\_destroy: leaving

chain\_reverse: leaving

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

/\* Распечатка перевернутой цепочки\*/

printEl: node|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@7

Output arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@c >- undf|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@b

Input arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@6f -< node|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@6b

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

printEl: node|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@8

Output arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@e >- undf|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@d

Input arcs:

arc|const|pos|temporary|actual:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@56 -< node|const:/tmp/autoseg/@@74c22f2b-20f9-474a-b015-cafe579c9aac@52

**Тестирование программы (крайние случаи):**

Поведение программы при попытке перевернуть пустую цепочку.

**Фрагмент измененного текста программы /lib/test/reverse:**

[{

chain\_top\_link, meta\_chain\_link, link,

logger, descr, node, empty, empty\_chain

}],

{[]})

genEl([1\_: assign\_: node\_: const\_: logger])

genEl([1\_: assign\_: node\_: const\_: empty])

genEl([1\_: assign\_: node\_: const\_: empty\_chain])

label(reverse)

/\*Reverse chain\*/

call([1\_: fixed\_: chain\_push\_first\_singlet, 2\_: fixed\_: prmD = [{1\_: empty\_chain, 2\_: meta\_chain\_link, 3\_: logger}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

call([1\_: fixed\_: chain\_reverse, 2\_: fixed\_: prm9 = [{1\_: empty, 2\_: empty, 3\_: meta\_chain\_link, 4\_: empty, 5\_: logger}], 3\_: assign\_: descr])

waitReturn([1\_: fixed\_: descr])

**Результат выполнения программы:**

chain\_push\_first\_singlet: entering

autosegment is 03380AE8

autosegment is 03380B60

autosegment is 03380BD8

chain\_get\_next: entering

chain\_get\_next: leaving

Process 03849FE8 is dying now

Unlink tmp segment autoseg5

Process 03849200 is dying now

Unlink tmp segment autoseg4

Process 03848530 is dying now

Unlink tmp segment autoseg3

chain\_push\_first\_singlet: leaving

Process 03847388 is dying now

Unlink tmp segment autoseg2

autosegment is 03380A70

chain\_reverse: entering

autosegment is 03380AE8

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

Process 03848D50 is dying now

Unlink tmp segment autoseg7

autosegment is 03380AE8

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

Process 03848C10 is dying now

Unlink tmp segment autoseg8

autosegment is 03380AE8

chain\_push\_singlet: entering

autosegment is 03380B60

autosegment is 03380BD8

autosegment is 03380C50

chain\_get\_next: entering

chain\_get\_next: leaving

Process 03907E70 is dying now

Unlink tmp segment autoseg12

Process 03907290 is dying now

Unlink tmp segment autoseg11

Process 0384A600 is dying now

Unlink tmp segment autoseg10

chain\_push\_singlet: leaving

Process 03849A70 is dying now

Unlink tmp segment autoseg9

chain\_reverse: leaving

Process 03847680 is dying now

Unlink tmp segment autoseg6

autosegment is 03380A70

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

Process 03849C78 is dying now

Unlink tmp segment autoseg13

autosegment is 03380A70

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

Process 03846AC8 is dying now

Unlink tmp segment autoseg14

autosegment is 03380A70

chain\_get\_next\_singlet: entering

chain\_get\_next\_singlet: leaving

Process 03848350 is dying now

Unlink tmp segment autoseg15

autosegment is 03380A70

chain\_destroy: entering

autosegment is 03380AE8

autosegment is 03380B60

autosegment is 03380BD8

chain\_get\_next: entering

chain\_get\_next: leaving

Process 03907128 is dying now

Unlink tmp segment autoseg19

Process 03848CD8 is dying now

Unlink tmp segment autoseg18

Process 038478B0 is dying now

Unlink tmp segment autoseg17

autosegment is 03380AE8

chain\_get\_next: entering

chain\_get\_next: leaving

Process 03849D40 is dying now

Unlink tmp segment autoseg20

chain\_destroy: leaving

Process 03847EC8 is dying now

Unlink tmp segment autoseg16

Process 03846690 is dying now

Unlink tmp segment autoseg1

Process 03845178 is dying now

Unlink tmp segment autoseg

cache hits: 113

cache misses:194

cache replaces:0

**Вывод:**

В ходе лабораторной работы, мной была изучена структура данных sc-цепочка синглетонов, принцип ее построения и возможные операции над ней. Также мной был разработан алгоритм построения перевернутой цепочки и реализован на языке программирования SCP. В результате была получена работающая программа, которая может быть использована для построения перевернутых цепочек синглетонов. Трудности, которые встретились в ходе работы, были преодолены и их появление простимулировало более глубокое понимание структур и алгоритмов, использованных в ходе работы. Также программа была протестирована на крайние случаи и в отчете отражена работа программы в приведенных условиях.