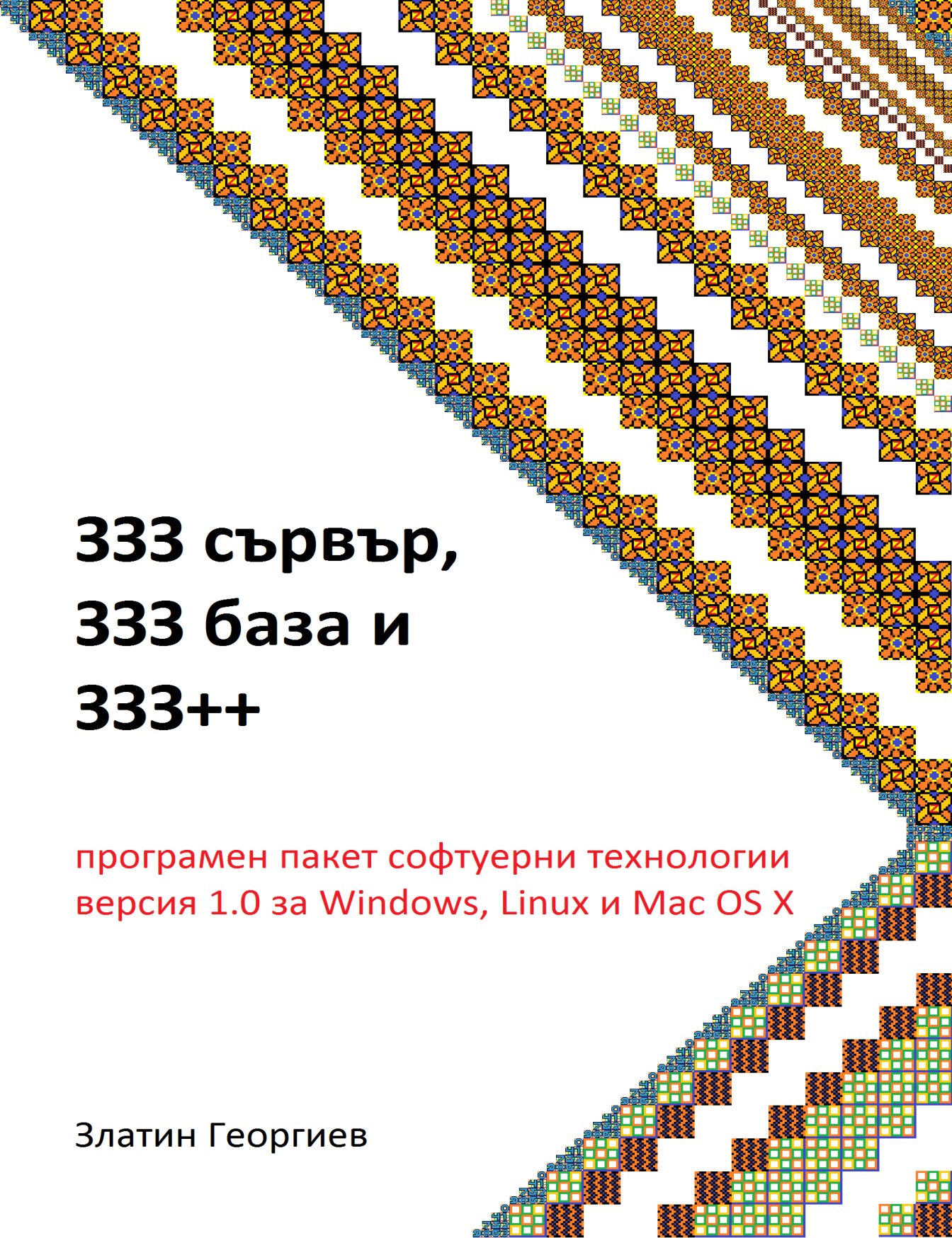

ЗЗЗ сървър, ЗЗЗ база и ЗЗЗ++

програмен пакет софтуерни технологии
версия 1.0 за Windows, Linux и Mac OS X

Златин Георгиев

333 сървър, 333 база и 333++

NoSQL програмен пакет софтуерни технологии версия 1.16 за Windows™, Mac OS X™, Linux (включително за Raspberry PI) и Android

Златин Георгиев

This book is for sale at http://leanpub.com/introinzzzserver

This version was published on 2017-04-16

Благодаря на моето семейство, че ме подкрепя винаги!

Contents

1. Въведение

Забравете за таблиците при работа с бази данни! Описвайте информацията така както е подредена в природата. Забравете за ограниченията в обема информация, която можете да обработвате - няма такива! Вие не търсите информация, просто се разхождате в нея! В тази книга не се описва обикновена база данни, а работа със знания.

Книгата описва компактна сървърна NoSQL база данни, реализирана в единствен изпълним файл с размер приблизително 1 MB и предоставя:

- вграден Web сървър;
- лесно конфигуриране;
- транзакционна защита;
- вграден език за управление;
- възможност за разпределено изпълнение на заявки;
- скалируемост;
- работа както с малки, така и с много големи обеми от информация PETABYTES x N;
- обектно-ориентирана;
- лесно описание и съхранение на разнородна информация;
- реализации за операционните системи Windows™, Mac OS X™ и Linux (включително за Raspberry PI)

и още много други...

2. Какво ни е необходимо преди да продължим?

1. Софтуерен пакет[1] предоставящ технологиите ЗЗЗ сървър, ЗЗЗ база и ЗЗЗ++, който можете да изтеглите за предпочитаната от Вас операционна система Windows™, Mac OS X™ или Linux.
2. Текстов редактор, който поддържа UTF-8.
3. Пълното издание на настоящата книга.
4. Желание за развитие.

[1] http://demo.zzz.bg/

3. Глава I. ЗЗЗ сървър

ЗЗЗ сървър е socket и http многопотребителски сървър реализиран за различни операционни системи Windows™, Mac OS X™, Linux (включително за Raspberry PI). Той може да бъде достъпван чрез браузър или от други приложения написани на различни програмни езици. Има варианти, които поддържат TLS криптирани връзки, които няма да се разглеждат в настоящото издание на книгата.

Системни изисквания:

- минимални 0.4 MB RAM, 386 CPU и 1 MB място на диска;
- максимални до най-съвременните компютри;

Самостоятелно приложение - не изисква инсталиране на допълнителен софтуер.

3.1 Команден ред

Примерен команден ред:

За Windows™

```
ZZZServer.exe -p 80 -s 100
```

ZZZServer.exe - стартиращ файл

За Linux или за Mac OS X™

```
./ZZZServer -p 80 -s 100
```

ZZZServer - стартиращ файл
-p 80 – 80 е номерът на порта, на който слуша сървъра.
-s 100 – 100 е максималният брой програмни стъпки, които да бъдат изпълнени от сървъра за активния клиент преди да премине към следващия.

Описание на други параметри задавани чрез командния ред при стартиране на ЗЗЗ сървър:
-p - задава номера на порта, на който слуша сървъра по подразбиране е 3333;
-d - задава директрорията за документи за сървъра по подразбиране е `www`;
-c - максимален брой на едновременно обработваните клиенти от сървъра по подразбиране `100`;
-s - максимален брой програмни стъпки, които изпълнява сървъра за активния клиент преди да премине към следващия по подразбиране 30;
-h - извежда помощна информация;
-l - задава началото на имената на лог файлове, може да включва и име на папка, но ако папката не съществува не я създава. По подразбиране не прави лог файлове;
-lf - задава формат на времето към името на лог файловете. По подразбиране е `-%Y-%m-%d-%H.log`;

Възможните форматиращи кодове са:
- `%a` - съкратено име на ден от седмицата;
- `%A` - пълно име на ден от седмицата;
- `%b` - съкратено име на месец;
- `%B` - пълно име на месец;

- `%c` - представяне на датата и времето в зависимост от местното време;
- `%d` - ден от месеца като цяло число (01 – 31);
- `%H` - час в 24-часов формат (00 – 23);
- `%I` - час в 12-часов формат (01 – 12);
- `%j` - ден от годината като десетично число (001 – 366);
- `%m` - месец като десетично число (01 – 12);
- `%M` - минута като десетично число (00 – 59);
- `%p` - текущия A.M./P.M. индикатор за 12-часов формат на времето;
- `%S` - секунда като десетично число (00 – 59);
- `%U` - седмицата от годината като десетично число, с неделя като първи ден от седмицата (00 – 53);
- `%w` - ден от седмицата като десетично число (0 – 6; неделята е 0);
- `%W` - седмицата от годината като десетично число, с понеделник като първи ден от седмицата (00 – 53);
- `%x` - представяне на датата за текущите местни настройки;
- `%X` - представяне на времето за текущите местни настройки;
- `%y` - годината без столетията, като десетично число (00 – 99);
- `%Y` - годината със столетията, като десетично число;
- `%z, %Z` - времевата зона;
- `%%` - знака %;

-ban - забранява различни дейности:
-ban base create - забранява създаването на нови бази. Полезна е когато сървъра трябва да работи само с предварително създадени бази;
-ban base write - освен, че забранява създаването на нови бази, забранява и записа в съществуващите бази управлявани от сървъра. Позволява само четене от базите. Полезна при справочни системи или когато съдържанието на базите трябва да остане непроменено;

Следващите опции не се препоръчва да бъдат включвани за ЗЗЗ сървър, който е видим в Интернет.

-system - позволява изпълнение на системни команди по подразбиране е изключена;
-write - позволява командите за запис във файлове по подразбиране е изключена.

Избирането на папка, от която ЗЗЗ сървъра да чете става чрез конфигурационния файл `zzzserver.cfg`. Той трябва да се намира в работната папка на сървъра и съдържа последователни редове с име на хост след него знака равно и име на папка. Името на хост може да завършва с двоеточие и номер на порт. За неописаните в конфигурационния файл хостове, които обслужва сървъра се използва папката по подразбиране `www` или указаната с опцията `-d` от командния ред.

По този начин един ЗЗЗ сървър може да обслужва множество хостове.

Един примерен *zzzserver.cfg* файл е следния:

```
kasovbon.zzz.bg=wwwkasovbon
www.kasovbon.zzz.bg=wwwkasovbon
```

3.2 Управляващ език

Управляващият език е реализация на езика TTM с вградени команди за управление на ЗЗЗ бази, за свързване с други сървъри и за работа с "живи" обекти. Езикът TTM е рекурсивен функционален макро език (подобен на езика Lisp), който може да се използва в системи за изкуствен интелект.

3.2.1 Общи положения

Интерпретаторът на езика TTM обработва текстова последователност по следния простичък начин:

- Ако срещне `#[команда;параметри]` изпълнява командата с указаните параметри разделени с точка и запетая и подава резултата от изпълнението отново за интерпретиране.
- Ако срещне `##[команда;параметри]` изпълнява командата с указаните параметри разделени с точка и запетая и връща резултата от изпълнението.
- Ако срещне `[защитен текст]` връща защитения текст като премахва ограждащите скоби и не го изпълнява дори в него да има команди.
- Ако срещне символи за табулация или нов ред ги прескача.
- Връща всяка друга текстова последователност като резултат от изпълнението на TTM програмата.

пример: Следната програма на TTM извежда съобщението 'Hello world from ZZZ Server!'

```
#[cout;[Hello world from ZZZ Server!]]
```

Командата `cout` извежда низ.

3.2.2 Команди

Синтаксисът на командите е:

`#[команда;параметър1;параметър2;параметър3]` - активна команда, т.е. резултатът се интерпретира отново

`##[команда;параметър1;параметър2;параметър3]` - пасивна команда, т.е. резултатът се връща без да се интерпретира

Броят на параметрите е неограничен, ако е необходимо в параметъра да има точка и запетая, тя трябва да бъде заградена в квадратни скоби.

Описанието на командите е следното: име на командата, действие на командата, параметри, ако има, примерно използване в активен режим (аналогично е използването в пасивен режим, тогава пред командата има `##`).

ВНИМАНИЕ:

Командите могат да бъдат тествани като се стартира един ЗЗЗ сървър, и в браузъра от страницата http://localhost/ се избере ЗЗЗ дърво, в чието горно дясно текстово поле могат да се изписват командите и да се изпълняват с натискане на бутона `[Изпълни]`.

3.2.2.1 Основни команди

`cout+`

Включване на трасирането

Няма параметри.

пример:

```
#[cout+]
#[cout;Test trace on.]
#[cout-]
```

`cout-`

Изключване на трасирането

Няма параметри.

пример:

```
#[cout+]
#[cout;Test trace off.]
#[cout-]
```

=

Дефинира глобален низ

Първият параметър е името на променливата, в която се помни низа, а вторият - самият низ.

пример:

```
#[=;test;[This is the defined string.]]
#[cout;##[test]]
```

.=

Дефинира низ за съответната област на видимост

Първият параметър е името на променливата, в която се помни низа, а вторият - самият низ. Препоръчва се използването на настоящата функция пред използването на = навсякъде където е възможно, особено при работа с класове!

пример:

```
#[cout;
        #[.v;5]
        #[.=;text;#[.^]. [This is the defined string.]]
        #[.+]
                #[.=;text;#[.^]. [This is the defined string.]]
                ##[text]##[chr;10]
        #[.-]
        ##[text]
]
```

.v

Задава нивото на видимост.

Първият параметър е нивото като число.

пример:

```
#[cout;
        #[.v;5]
        #[.=;text;#[.^]. [This is the defined string.]]
        #[.+]
                #[.=;text;#[.^]. [This is the defined string.]]
                ##[text]##[chr;10]
        #[.-]
        ##[text]
]
```

.^

Прочита нивото на видимост.

пример:

```
#[cout;
        #[.v;5]
        #[.=;text;#[.^]. [This is the defined string.]]
        #[.+]
                #[.=;text;#[.^]. [This is the defined string.]]
                ##[text]##[chr;10]
        #[.-]
        ##[text]
]
```

.+

Увеличава нивото на видимост с единица.

пример:

```
#[cout;
        #[.v;5]
        #[.=;text;#[.^]. [This is the defined string.]]
        #[.+]
                #[.=;text;#[.^]. [This is the defined string.]]
                ##[text]##[chr;10]
        #[.-]
        ##[text]
]
```

.-

Намалява нивото на видимост с единица.

пример:

```
#[cout;
        #[.v;5]
        #[.=;text;#[.^]. [This is the defined string.]]
        #[.+]
                #[.=;text;#[.^]. [This is the defined string.]]
                ##[text]##[chr;10]
        #[.-]
        ##[text]
]
```

cout

Разпечатва низ в конзолата и го извежда в резултатния буфер

Функцията има един параметър - текстов низ, който трябва да бъде изведен.

пример:

```
#[cout;
                Hello World from ZZZ Server!<br />
                Hello World from ZZZ Server!<br />

        [
                Hello World from ZZZ Server!
                Hello World from ZZZ Server!
        ]
]
```

out

Извежда низ в резултатния буфер без да го разпечатва в конзолата

Функцията има един параметър - низът, който трябва да бъде изведен.

пример:

```
#[out;[Hello world from ZZZ Server!]]
```

+

Събира две числа с двойна точност

Първият и вторият параметър са събираемите. Третият незадължителен параметър е точността след десетичната запетая, която по подразбиране е 17.

пример:

```
#[cout;#[+;1;2]]
```

-

Изважда две числа с двойна точност

Първият параметър е умаляемото, вторият умалителят. Третият незадължителен параметър е точността след десетичната запетая, която по подразбиране е 17.

пример:

```
#[cout;#[-;2;1]]
```

*****

Умножава две числа с двойна точност

Първият и вторият параметър са множителите. Третият незадължителен параметър е точността след десетичната запетая, която по подразбиране е 17.

пример:

```
#[cout;#[*;2;3]]
```

/

Дели две числа с двойна точност

Първият параметър е делимото, а вторият - делителят. Третият незадължителен параметър е точността след десетичната запетая, която по подразбиране е 17.

пример:

```
#[cout;#[/;6;3]]
```

==

Сравнява първите два параметъра като низове и ако са равни - изпълнява третия, ако са различни - четвъртия.

пример:

```
#[==;first;second;[
                #[cout;Both parameters are equal.]
        ];[
                #[cout;Both parameters are different.]
        ]
]
```

==n

Сравнява първите два параметъра като числа и ако са равни - изпълнява третия, ако са различни - четвъртия.

пример:

```
#[==n;1;2;[
                #[cout;Both parameters are equal.]
        ];[
                #[cout;Both parameters are different.]
        ]
]
```

>=

Сравнява първите два параметъра като низове и ако първият е по-голям или равен на втория - изпълнява третия, в противен случай изпълнява четвъртия.

пример:

```
#[cout;
        #[>=;first;second;[
                        The first parameter is greater or equal to the second one.
                ];[
                        The second parameter is greater than the first one.
                ]
        ]
]
```

>=n

Сравнява първите два параметъра като числа и ако първия е по-голям или равен на втория - изпълнява третия, в противен случай изпълнява четвъртия.

пример:

```
#[cout;
        #[>=n;1;2;[
                        The first parameter is greater or equal to the second one.
                ];[
                        The second parameter is greater than the first one.
                ]
        ]
]
```

=.

Сегментира низ

Използва се за сегментиране на низ дефиниран с командата = като замества текста в дефинирания низ със съответните параметри от командата и ги номерира според тяхната последователност.

пример:

```
#[=;MyFunc;[
                [I am {TITLE} {FIRST_NAME} {LAST_NAME}.]
        ]
]
#[=.;MyFunc;{TITLE};{FIRST_NAME};{LAST_NAME}]

#[cout;#[MyFunc;Mr;Steven;Atanasov]]
```

#[...]

Изпълнява низ дефиниран с = като замества параметрите сегментирани със =.

пример:

```
#[=;MyFunc;[
                #[cout;[I am {TITLE} {FIRST_NAME} {LAST_NAME}.]]
        ]
]
#[=.;MyFunc;{TITLE};{FIRST_NAME};{LAST_NAME}]
#[MyFunc;Mr;John;Edwards]
```

##[...]

Връща низа дефиниран с =

пример:

```
#[=;test;[
                This is a defined string.
        ]
]
#[cout;##[test]]
```

is

Замества текста в първия параметър, който съответства на този от втория параметър, с текста от третия

пример:

```
#[=;test;[
                This is a defined string.
        ]
]
#[is;test;defined;stored]
#[cout;##[test]]
```

^.

Прочита поредния символ от дефиниран с = низ

Първият параметър е името на дефинирания низ. Вторият параметър е резултът при неуспех.

пример:

```
#[=;test;
        This is a defined string.
]
#[cout;
        #[^.;test;<end>]
        #[^.;test;<end>]
]
```

=x

Изтрива името на низ дефиниран с =

пример:

```
#[=;test;
        This is a defined string.
]
#[cout;
        ##[test]
        #[=x;test]
        ##[test]
]
```

=x..

Изтрива всички имена на низовете дефинирани с =

пример:

```
#[=;new line;##[chr;10]]
#[=;test1;
        This is a defined string 1.
]
#[=;test2;
        This is a defined string 2.
]
#[cout;
        ##[test1]##[new line]
        ##[test2]##[new line]
        #[=x..]
        ##[test1]##[new line]
        ##[test2]##[new line]
]
```

x*v

Позиционира указателя към текущия символ в началото на сегмент с указано име

Първият параметър е името на низа съдържащ сегмента. Вторият параметър е името на сегмента, след който трябва да позиционираме указателя към текущия символ. Третият параметър е резулататът, ако няма повече сегменти.

пример:

```
#[=;text;
        This is the defined string.
]
#[cout;
        #[x*v;text;the ;<end>]
        #[^.;text;<end>]
        #[^.;text;<end>]
]
```

***0**

Позиционира указателя към текущия символ в началото на низа дефиниран с =

пример:

```
#[=;text;
        This is the defined string.
]
#[cout;
        #[x*v;text;the ;<end>]

        #[^.;text;<end>]
        #[^.;text;<end>]

        #[*0;text]

        ##[chr;10]
        #[^.;text;<end>]
        #[^.;text;<end>]
]
```

ta

Сравнява два UNICODE низа и определя дали първият е по-голям или равен на втория по азбучен ред

пример:

```
#[cout;
        #[ta;abc;def;[
                        "abc" >= "def"
                ];[
                        "abc" < "def"
                ]
        ]
]
```

chr

Преобразува число към уникод

пример:

```
#[cout;
        <div style="font-size:2em">#[chr;425]</div>

        Copyright #[chr;169] 2015 - 2017 ZZZ Ltd. All rights reserved.
]
```

tcn

Преобразува уникод символ към шестнадесетично число

пример:

```
#[cout;
        symbol #[chr;#[refmtl;%04x;%d;#[tcn;W]]], code #[tcn;W]
]
```

^..

Прочита n символа от дефиниран с = низ или ако е достигнат края на низа връща последния параметър (#[^.;name;n;{after end}])

пример:

```
#[=;test;
        This is a defined string.
]
#[cout;
        #[^..;test;3;{after end}]##[chr;10]
        #[^..;test;4;{after end}]##[chr;10]
        #[^..;test;5;{after end}]##[chr;10]
        #[^..;test;6;{after end}]##[chr;10]
        #[^..;test;10;{after end}]##[chr;10]
        #[^..;test;10;{after end}]##[chr;10]
]
```

^(..*

Прочита дефиниран с = низ от началото му до указателя

пример:

```
#[=;test;
        This a defined string.
]
#[*v;test;10]
#[cout;#[^(..*;test]]
```

^*..)

Прочита дефиниран с = низ от указателя до края му

пример:

```
#[=;test;
        This a defined string.
]
#[*v;test;10]
#[cout;#[^*..);test]]
```

***^**

Прочита указателя на дефиниран с = низ

пример:

```
#[=;test;
        This a defined string.
]
#[*v;test;10]
#[cout;#[*^;test]]
```

***v**

Задава указателя на дефиниран с = низ

пример:

```
#[=;test;
        This a defined string.
]
#[*v;test;10]
#[cout;#[*^;test]]
```

trf

Заменя първия срещнат подниз с втория

пример:

```
#[=;test;
        This is a defined string.
]
#[trf;test;defined;stored]
#[cout;##[test]]
```

ts

Копира сегмент между два низа от низ дефиниран с =

пример:

```
#[=;test;
        This is the defined string.
]
#[cout;#[ts;test;the;string;{error}]]
```

tt

Тест за наличие на текст в дефиниран с = низ

пример:

```
#[=;test;
        This is a defined string.
]
#[cout;
        #[tt;test;defined;[
                        on true
                ];[
                        on false
                ]
        ]
]
```

ttc

Тест за наличие на текст в дефиниран с = низ, без значение на главни и малки букви

пример:

```
#[=;test;
        This is a defined string.
]
#[cout;
        #[ttc;test;Defined;[
                        on true
                ];[
                        on false
                ]
        ]
]
```

`|`

Извършва ИЛИ между две цели числа

пример:

```
#[cout;#[|;1;2]]
```

`&`

Извършва И между две цели числа

пример:

```
#[cout;#[&;1;2]]
```

`~`

Изпълнява изключващо ИЛИ над аргумента си

пример:

```
#[cout;#[~;1]]
```

`ln`

Извежда списък на дефинираните с = имена разделени с указания параметър (list named)

пример:

```
#[=x..]
#[=;test1;
        Test one.
]
#[=;test2;
        Test two.
]
#[cout;##[ln;##[chr;10]]]
```

oi

Отваря файл за четене (open input)

Първият параметър е описател на файла. Вторият параметър е името на файла. Третият параметър е резултатът при неуспех.

пример:

```
#[.=;MyFile;docfiles/MyFile.txt]
#[cout;
        #[==;;#[oi;fileHandleRead1;##[MyFile];{error}];[
                        #[frs;fileHandleRead1]
                        #[cf;fileHandleRead1]
                ];[
                        Error: Unable to open "##[MyFile]" for read!
                ]
        ]
]
```

oo

Отваря файл за запис (open output)

Работи само, ако е включена опцията `-write` от командния ред на 333 сървъра!

Първият параметър е описател на файла. Вторият параметър е името на файла. Третият параметър е резултатът при неуспех.

пример:

```
#[.=;MyFile;docfiles/MyFile.txt]
#[cout;
        #[==;;#[oo;fileHandleWrite1;##[MyFile];{error}];[
                        #[fps;fileHandleWrite1;Hello World from MyFile!]
                        #[cf;fileHandleWrite1]
                ];[
                        Error: Unable to open "##[MyFile]" for write!
                ]
        ]
]
```

oa

Отваря файл за добавяне (open append)

Работи само, ако е включена опцията `-write` от командния ред на 333 сървъра!

Първият параметър е описател на файла. Вторият параметър е името на файла. Третият параметър е резултатът при неуспех.

пример:

```
#[.=;MyFile;docfiles/MyFile.txt]
#[cout;
        #[==;;#[oa;fileHandleAppend1;##[MyFile];{error}];[
                        #[fps;fileHandleAppend1;Hello World from MyFile!]
                        #[cf;fileHandleAppend1]
                ];[
                        Error: Unable to open "##[MyFile]" for append!
                ]
        ]
]
```

`cf`

Затваря файл (close file)

Първият параметър е описателят на файла.

пример:

```
#[.=;MyFile;docfiles/MyFile.txt]
#[cout;
        #[==;;#[oi;fileHandleRead1;##[MyFile];{error}];[
                        #[frs;fileHandleRead1]
                        #[cf;fileHandleRead1]
                ];[
                        Error: Unable to open "##[MyFile]" for read!
                ]
        ]
]
```

`frs`

Прочита низ от файл до подразбиращ се мета символ (file read string)

пример:

```
#[.=;MyFile;docfiles/MyFile.txt]
#[cout;
        #[==;;#[oi;fileHandleRead1;##[MyFile];{error}];[
                        #[frs;fileHandleRead1]
                        #[cf;fileHandleRead1]
                ];[
                        Error: Unable to open "##[MyFile]" for read!
                ]
        ]
]
```

frc

Прочита символ от файл (file read char)

пример:

```
#[.=;MyFile;docfiles/MyFile.txt]
#[cout;
        #[==;;#[oi;fileHandleRead1;##[MyFile];{error}];[
                        #[frc;fileHandleRead1]
                        #[frc;fileHandleRead1]
                        #[frc;fileHandleRead1]
                        #[frc;fileHandleRead1]
                        #[frc;fileHandleRead1]!

                        #[cf;fileHandleRead1]
                ];[
                        Error: Unable to open "##[MyFile]" for read!
                ]
```

```
        ]
]
```

fcm

Променя подразбиращия се мета символ (file change meta)

пример:

```
#[.=;MyFile;docfiles/MyFile.txt]
#[cout;
        #[==;;#[oi;fileHandleRead1;##[MyFile];{error}];[
                        #[fcm;fileHandleRead1;#[chr;32]]
                        #[frs;fileHandleRead1] #[frs;fileHandleRead1]!

                        #[cf;fileHandleRead1]
                ];[
                        Error: Unable to open "##[MyFile]" for read!
                ]
        ]
]
```

fps

Записва низ във файл (file print string)

пример:

```
#[.=;MyFile;docfiles/MyFile.txt]
#[cout;
        #[==;;#[oo;fileHandleWrite1;##[MyFile];{error}];[
                        #[fps;fileHandleWrite1;Hello World from MyFile!]
                        #[cf;fileHandleWrite1]
                ];[
                        Error: Unable to open "##[MyFile]" for write!
                ]
        ]
]
```

`load`

Зарежда файл като при активно извикване го интерпретира

Първият параметър е името на файла. Вторият параметър е резултатът при неуспех.

пример:

```
#[load;docfiles/MyProg.ttm;{error}]
```

`system`

Изпълнява команда от операционната система

Работи само, ако е включена опцията `-system` от командния ред на 333 сървъра!

Първият параметър е командата заедно с нейните параметри.

пример:

```
#[cout;
        #[system;ls]
]
```

s2us

Преобразува низ към низ с уникод кодове (string to unicode string)

пример:

```
#[cout;
        #[s2us;Това е тест.]
]
```

#

Използва се за вмъкване на коментари (null command)

пример:

```
#[cout;
        #[#; This is a comment. ]
        This isn't a comment.
]
```

exit

Спира интерпретирането

пример:

```
#[exit]
```

3.2.2.2 Допълнителни полезни функции

`fmtl`

Форматира дълго, цяло число като използва форматиращ низ стандартен за командата sprintf от C/C++ *Първият параметър е форматиращият низ. Вторият параметър е числото, което трябва да бъде форматирано.*

пример:

```
#[.=;text;текстов низ]
#[cout;
        дължина: ##[fmtl;[%08lX];##[(..);text]] шестнайсетично, ##[fmtl;[%08ld];##[\
(..);text]] десетично##[chr;10]
        елемент: ##[text]
]
```

`refmtl`

Преформатира дълго, цяло число като използва форматиращ низ стандартен за командата sprintf от C/C++ С тази команда примерно може бързо да бъде преобразувано едно число от 16-тично към 10-тично. *Първите два параметъра са форматиращи низове. Третият параметър е числото, което трябва да бъде преформатирано.*

пример:

```
#[cout;
        #[refmtl;%I64X;%I64d;AF]
]
```

fmtd

Форматира реално, цяло число с двойна точност като използва форматиращ низ стандартен за командата sprintf от C/C++ *Първият параметър е форматиращият низ. Вторият параметър е числото, което трябва да бъде форматирано.*

пример:

```
#[cout;
        #[fmtd;%0.2f;5.3]
]
```

getTime

Връща броят на милисекундите от 1-ви януари, 1970 г., 00:00:00 GMT

пример:

```
#[cout;
        ##[getTime]
]
```

fmtTime

Форматира време *Първият параметър е форматиращият низ. Вторият параметър е времето в милисекунди.*

пример:

```
#[cout;
        #[.=;NOW;##[getTime]]
        ##[fmtTime;[%d.%m.%Y година, %H:%M:%S.%s часа];##[NOW]]
]
```

getFileTime

Връща времето от последната промяна на файл, в милисекунди *Първият параметър е пътят до файла.*

пример:

```
#[.=;_base_;docbases/zzzbase]
#[.=;_LAST_MODIFIED_;##[getFileTime;##[_base_]1]]
#[cout;
        Базата "##[_base_]" е променяна в ##[fmtTime;[%H:%M:%S.%s часа на %d.%m.%Y \
година];##[_LAST_MODIFIED_]].
]
```

getFileSize

Връща размера на файла *Първият параметър е пътят до файла.*

пример:

```
#[.=;_base_;docbases/zzzbase]
#[.=;_base_size_;##[fmtd;%0.f;##[+;##[getFileSize;##[_base_]1];##[getFileSiz\
e;##[_base_]2]]]]
#[.=;_KB_;##[/;##[_base_size_];1024]]
#[.=;_MB_;##[/;##[_KB_];1024]]
#[.=;_GB_;##[/;##[_MB_];1024]]
#[.=;_TB_;##[/;##[_GB_];1024]]
#[cout;
        Базата "##[_base_]" заема:##[chr;10]
          ##[_base_size_] байта[;]##[chr;10]
          ##[fmtd;%0.2f;##[_KB_]] килобайта[;]##[chr;10]
          ##[fmtd;%0.4f;##[_MB_]] мегабайта[;]##[chr;10]
          ##[fmtd;%0.6f;##[_GB_]] гигабайта[;]##[chr;10]
          ##[fmtd;%0.8f;##[_TB_]] терабайта.
]
```

outLoadBinary

Извежда файл в двоичен формат *Първият параметър е пътят до файла. Вторият параметър е резултатът, който връща в случай на грешка.*

пример:

```
##[outLoadBinary;www/images/icon_64x64.png;{error}]
```

base64LoadBinary

Кодира файл в base64 *Първият параметър е пътят до файла. Вторият параметър е резултатът, който връща в случай на грешка.*

пример:

```
#[cout;
        <img src="data:image/png[;]base64,##[base64LoadBinary;www/images/icon1_64x6\
4.png;{error}]" alt="Using binary image data to display an image in HTML" />
        <img src="data:image/png[;]base64,##[base64LoadBinary;www/images/icon_64x64\
.png;{error}]" alt="Using binary image data to display an image in HTML" />
        <img src="data:image/png[;]base64,##[base64LoadBinary;www/images/lin64.png;\
{error}]" alt="Using binary image data to display an image in HTML" />
        <img src="data:image/png[;]base64,##[base64LoadBinary;www/images/mac64.png;\
{error}]" alt="Using binary image data to display an image in HTML" />
        <img src="data:image/png[;]base64,##[base64LoadBinary;www/images/win64.png;\
{error}]" alt="Using binary image data to display an image in HTML" />
        <img src="data:image/png[;]base64,##[base64LoadBinary;www/images/android.pn\
g;{error}]" alt="Using binary image data to display an image in HTML" />
        <img src="data:image/png[;]base64,##[base64LoadBinary;www/images/ios.png;{e\
rror}]" alt="Using binary image data to display an image in HTML" />
        <img src="data:image/png[;]base64,##[base64LoadBinary;www/images/windows-ph\
one.png;{error}]" alt="Using binary image data to display an image in HTML" \
/>
]
```

^UID

Връща уникален за текущото стартиране на 333 сървъра номер като максималната му стойност е 0x7fffffffffffffffI64

пример:

```
#[cout;
        ##[^UID]
]
```

4. Глава II. ЗЗЗ база

ЗЗЗ база е NoSQL система за управление на бази данни и знания.

Тя съчетава следните характеристики:

- йерархична;
- обектно-ориентирана;
- разпределена;
- скалируема;
- подходяща за обработка както на много малки(няколко байта), така и на невероятно големи обеми от информация(петабайти x n);
- всички елементи са индексирани;
- възможност за мигновено търсене на най-близките до несъществуващ елемент - трудна задача за повечето разпространени SQL и NoSQL бази данни при голямо количество на елементите;
- функционален, макро език за управление подобен на Lisp;
- права за достъп;
- възможност за лесно ползване от различни програмни езици;
- технология клиент-сървър, като е възможно един сървър да бъде клиент на друг сървър;
- технология за сортиране на елементи със сложност O(N);
- мигновено търсене на елемент в множество с огромен брой(трилиони) елементи;

4.1 Команди управляващи базата

Тези команди позволяват свободното избиране на ЗЗЗ сървъри и ЗЗЗ бази както и придвижване в техните множества/елементи. По време на придвижването можете да изтривате и променяте множества/елементи или да влизате в подмножествата и надмножествата им.

4.1.1 Избор на ЗЗЗ сървър

~+

Активира връзка към ЗЗЗ сървър

Първият параметър е името на хоста. Вторият параметър е номерът на порта.

пример:

```
#[~+;localhost;80]
#[cout;
        #[~>;localhost;80;10000;[#[cout;Hello world from another ZZZServer!]]]
]
#[~-;localhost;80]
```

~-

Освобождава връзка към ЗЗЗ сървър

Първият параметър е името на хоста. Вторият параметър е номерът на порта.

пример:

```
#[~+;localhost;80]
#[cout;
        #[~>;localhost;80;10000;[#[cout;Hello world from another ZZZServer!]]]
]
#[~-;localhost;80]
```

> ~>
>
> **Изпълнява отдалечено програма от връзка към ЗЗЗ сървър**
>
> Програмата трябва да започва с # *Първият параметър е името на хоста. Вторият параметър е номерът на порта. Третият параметър е програмата, която трябва да бъде изпълнена.*

пример:

```
#[~+;localhost;80]
#[cout;
        #[~>;localhost;80;10000;[#[cout;Hello world from another ZZZServer!]]]
]
#[~-;localhost;80]
```

4.1.2 Избор на ЗЗЗ база

> ~~+
>
> **Активира връзка към ЗЗЗ база**
>
> Ако базата не съществува я създава, ако не е включена някоя от опциите `-ban base create` или `-ban base write` от командния ред на ЗЗЗ сървъра!

Първият параметър е името на базата.

Програма ползваща командите за достъп до ЗЗЗ база::

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]
#[.=;_processId_;#[~~v;##[_base_];10000]]

#[==;true;#[~~>^;##[_base_];##[_processId_]];;]
#[~~x.;##[_base_];##[_processId_]]
#[==;true;#[~~+.;##[_base_];##[_processId_];ZZZ];;]
#[==;true;#[~~>v;##[_base_];##[_processId_]];;]

        #[==;true;#[~~+.;##[_base_];##[_processId_];плодове];;]
        #[==;true;#[~~>v;##[_base_];##[_processId_]];;]
                #[==;true;#[~~+.;##[_base_];##[_processId_];ябълки];;]
                #[==;true;#[~~+.;##[_base_];##[_processId_];круши];;]
                #[==;true;#[~~+.;##[_base_];##[_processId_];сливи];;]
                #[==;true;#[~~>^;##[_base_];##[_processId_]];;]

        #[==;true;#[~~+.;##[_base_];##[_processId_];зеленчуци];;]
        #[==;true;#[~~>v;##[_base_];##[_processId_]];;]
                #[==;true;#[~~+.;##[_base_];##[_processId_];моркови];;]
                #[==;true;#[~~+.;##[_base_];##[_processId_];краставици];;]
                #[==;true;#[~~+.;##[_base_];##[_processId_];зеле];;]
                #[==;true;#[~~>^;##[_base_];##[_processId_]];;]

#[~~.;##[_base_];##[_processId_]]
#[~~-;##[_base_]]

#[cout;
        #[ZZZ2SimpleTXTFromDividerPath;##[_base_];;/]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]#[ remove the base ]

        #[ add ;плодове]
        #[ subset ]
                #[ add ;ябълки]#[ add ;круши]#[ add ;сливи]
                #[ superset ]
        #[ add ;зеленчуци]
        #[ subset ]
                #[ add ;моркови]#[ add ;краставици]#[ add ;зеле]
                #[ superset ]

#[ close the base ]
#[ get the base as a simple text ]
```

> ~~-
>
> **Освобождава връзка към 333 база**
>
> *Първият параметър е името на базата.*

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]        -- Активира връзка към 333 база

        Тук се разполагат командите за достъп до базата
        ...

#[~~-;##[_base_]]        -- Освобождава връзка към 333 база

#[cout;
        #[ZZZ2SimpleTXTFromDividerPath;##[_base_];;/]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        Тук се разполагат командите за достъп до базата
        ...

#[ close the base ]
#[ get the base as a simple text ]
```

4.1.3 Транзакционна защита + управление на четенето и записа

∼∼v

Начало на процес за запис в ЗЗЗ база

Не позволява запис, ако е включена опцията `-ban base write` от командния ред на ЗЗЗ сървъра!

Първият параметър е името на базата. Вторият параметър е времето в милисекунди за изчакване на активирането на процеса за запис. Третият незадължителен параметър е име на потребител. Четвъртият незадължителен параметър е парола. Резултатът е номерът на процеса, който при успех е по-голям или равен на 0.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                    -- Актив
#[=;_processId_;#[~~v;##[_base_];10000]]              -- Начало на процес за запис в 33
 база

        Тук се разполагат командите за достъп до базата
        ...
        #[~~+.;##[_base_];##[_processId_];елемент]        -- Добавя елемент в главното мн
жество на базата

#[~~.;##[_base_];##[_processId_]]                                -- Край на процес за чет
 333 база
#[~~-;##[_base_]]                                                    -- Освоб

#[cout;
        #[ZZZ2SimpleTXTFromDividerPath;##[_base_];;/]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        Тук се разполагат командите за достъп до базата
        ...
        #[ add ;елемент]        -- Добавя елемент в главното множество на базата

#[ close the base ]
#[ get the base as a simple text ]
```

~~^

Начало на процес за четене от 333 база

Първият параметър е името на базата. Вторият незадължителен параметър е

име на потребител. Третият незадължителен параметър е парола. Резултатът е номерът на процеса, който при успех е по-голям или равен на 0.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                    -- Активира връз
#[=;_processId_;#[~~^;##[_base_]]]                          -- Начало на процес за четене о
за

        Тук се разполагат командите за достъп до базата
        ...
        #[cout;
                #[~~^.;##[_base_];##[_processId_]]
        ]

#[~~.;##[_base_];##[_processId_]]                           -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        Тук се разполагат командите за достъп до базата
        ...
        #[cout;
                #[ get ]
        ]

#[ close the base ]
```

~~.

Край на процес за четене или запис в 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                    -- Актив
#[=;_processId_;#[~~^;##[_base_]]]                          -- Начало на процес за
аза

        #[cout;
                ##[~~^.;##[_base_];##[_processId_]]
        ]

#[~~.;##[_base_];##[_processId_]]                           -- Край на процес за чет
 333 база
#[~~-;##[_base_]]                                                    -- Освоб
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        #[cout;
                #[ get ]
        ]                                       -- Прочита първия елемент от главното мн

#[ close the base ]
```

∼∼vx

Прекъсва процес за запис в 333 база като оставя базата с непроменено съдържание

Първият параметър е името на базата.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]
#[=;_processId_;#[~~v;##[_base_];10000]]                          -- Начало на процес за че
333 база

        ##[~~+.;##[_base_];##[_processId_];нов елемент]          -- Записва елемент в главн
то множество на базата

        #[~~vx;##[_base_]]
ис

#[~~.;##[_base_];##[_processId_]]                                 -- Край на проце
в 333 база
#[~~-;##[_base_]]

#[cout;
        #[ZZZ2SimpleTXTFromDividerPath;##[_base_];;/]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        #[ add ;нов елемент]        -- Записва елемент в главното множество на базата

        #[ undo changes ]               -- Отменя всички промени от текущия процес за за

#[ close the base ]
#[ get the base as a simple text ]
```

4.1.4 Работа с множествата и елементите на 333 база

∼∼+.

Добавя елемент към текущото множество на 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса. Третият параметър е самият елемент.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]
#[=;_processId_;#[~~v;##[_base_];10000]]                          -- Начало на процес за за
3 база

        ##[~~+.;##[_base_];##[_processId_];нов елемент]        -- Записва елемент в главн
то множество на базата

#[~~.;##[_base_];##[_processId_]]                                        -- Край на проце
в 333 база
#[~~-;##[_base_]]

#[cout;
```

```
        #[ZZZ2SimpleTXTFromDividerPath;##[_base_];;/]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        #[ add ;нов елемент]        -- Записва елемент в главното множество на базата

#[ close the base ]
#[ get the base as a simple text ]
```

~~/.

Променя текущия елемент от множество на 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса. Третият параметър е самата нова стойност на елемента.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]
#[=;_processId_;#[~~v;##[_base_];10000]]                                    -- Начало на проц
33 база

        ##[~~+.;##[_base_];##[_processId_];нов елемент]                -- Записва елемент
ото множество на базата
        #[~~/.;##[_base_];##[_processId_];променен елемент]        -- Променя текущия еле
ент в базата

#[~~.;##[_base_];##[_processId_]]                                                -- Край
 в 333 база
#[~~-;##[_base_]]
```

```
#[cout;
        #[ZZZ2SimpleTXTFromDividerPath;##[_base_];;/]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        #[ add ;нов елемент]                        -- Записва елемент в главното множест
        #[ change ;променен елемент]        -- Променя текущия елемент в базата

#[ close the base ]
#[ get the base as a simple text ]
```

> ～～-.
>
> **Изтрива текущия елемент от множество на 333 база**
>
> *Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.*

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]
#[=;_processId_;#[~~v;##[_base_];10000]]                                    -- Начало на проц
33 база

        ##[~~+.;##[_base_];##[_processId_];нов елемент]                -- Записва елемент
ото множество на базата
        #[~~-.;##[_base_];##[_processId_]]                                        -- Изтр

#[~~.;##[_base_];##[_processId_]]                                                -- Край
 в 333 база
#[~~-;##[_base_]]
```

```
#[cout;
        #[ZZZ2SimpleTXTFromDividerPath;##[_base_];;/]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        #[ add ;нов елемент]        -- Записва елемент в главното множество на базата
        #[ remove ]                                  -- Изтрива текущия елемент от базата

#[ close the base ]
#[ get the base as a simple text ]
```

> ∼∼^.
>
> **Прочита текущия елемент от множество на 333 база**
>
> *Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.*

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                              -- Актив
#[=;_processId_;#[~~^;##[_base_]]]                                  -- Начало на процес за
аза

        #[cout;##[~~^.;##[_base_];##[_processId_]]]        -- Прочита текущия елемент от
лавното множество на базата

#[~~.;##[_base_];##[_processId_]]                                   -- Край на процес за чет
 333 база
#[~~-;##[_base_]]                                                              -- Освоб
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        #[cout;#[ get ]]        -- Прочита текущия елемент от главното множество на базат

#[ close the base ]
```

~~>.

Придвижва се до елемент от множество на 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса. Третият параметър е стойността на търсения елемент.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                              -- Актив
#[=;_processId_;#[~~^;##[_base_]]]                                  -- Начало на процес за
аза

        ##[~~>.;##[_base_];##[_processId_];плодове]        -- Придвижва се до елемента "п
одове"
        #[cout;##[~~^.;##[_base_];##[_processId_]]]        -- Прочита текущия елемент от
лавното множество на базата

#[~~.;##[_base_];##[_processId_]]                                   -- Край на процес за чет
 333 база
#[~~-;##[_base_]]                                                              -- Освоб
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        #[ move to ;плодове]        -- Придвижва се до елемента "плодове"
        #[cout;#[ get ]]                -- Прочита текущия елемент от главното множество

#[ close the base ]
```

~~(.)

Връща дължината на елемент от множество на 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                                   -- Актив
#[=;_processId_;#[~~^;##[_base_]]]                                  -- Начало на процес за
аза

        ##[~~>.;##[_base_];##[_processId_];плодове]        -- Придвижва се до елемента "п
одове"
        #[.=;_елемент_;##[~~^.;##[_base_];##[_processId_]]]
        #[cout;
                ##[_елемент_]        ##[chr;9]- елемент##[chr;10]
                ##[~~(.);##[_base_];##[_processId_]]        ##[chr;9]- дължина на елемент
та в байтове##[chr;10]
                ##[(..);_елемент_]        ##[chr;9]- дължина на елемента като низ в симво
        ]

#[~~.;##[_base_];##[_processId_]]                                    -- Край на процес за чет
 333 база
#[~~-;##[_base_]]                                                                   -- Освоб
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        #[ move to ;плодове]        -- Придвижва се до елемента "плодове"
        #[.=;_елемент_;#[ get ]]
        #[cout;
                ##[_елемент_]               #[ tab ]- елемент#[ new line ]
                #[ length ]                       #[ tab ]- дължина на елемента в базата
                ##[(..);_елемент_]        #[ tab ]- дължина на елемента като низ в символ
        ]                                                -- Прочита текущия елемент и нег

#[ close the base ]
```

~~>v

Придвижва се до подмножеството на елемент от ЗЗЗ база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                              -- Актив
#[=;_processId_;#[~~^;##[_base_]]]                                  -- Начало на процес за
аза

        ##[~~>.;##[_base_];##[_processId_];плодове]        -- Придвижва се до елемента "п
одове"
        ##[~~>v;##[_base_];##[_processId_]]                          -- Придвижва се до под
текущия елемент
        #[cout;##[~~^.;##[_base_];##[_processId_]]]        -- Прочита текущия елемент от
азата

#[~~.;##[_base_];##[_processId_]]                                   -- Край на процес за чет
```

```
 333 база
#[~~-;##[_base_]]                                                    -- Освоб
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        #[ move to ;плодове]        -- Придвижва се до елемента "плодове"
        #[ subset ]                                -- Придвижва се до подмножеството на т
                #[cout;#[ get ]]        -- Прочита текущия елемент от базата

#[ close the base ]
```

> ~~>^
>
> **Придвижва се до надмножеството на елемент от 333 база**
>
> *Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.*

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                        -- Актив
#[=;_processId_;#[~~^;##[_base_]]]                              -- Начало на процес за
аза

        ##[~~>.;##[_base_];##[_processId_];плодове]        -- Придвижва се до елемента "п
одове"
        ##[~~>^;##[_base_];##[_processId_]]                        -- Придвижва се до над
текущия елемент
        #[cout;
                ##[~~^.;##[_base_];##[_processId_]]        ##[chr;9]- това е главният еле
жество на базата
        ]
```

```
#[~~.;##[_base_];##[_processId_]]                                    -- Край на процес за чет
 333 база
#[~~-;##[_base_]]                                                                    -- Освоб
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        #[ move to ;плодове]        -- Придвижва се до елемента "плодове"
        #[ superset ]                        -- Придвижва се до надмножеството на текущия
        #[cout;
                #[ get ]        #[ tab ]- това е главният елемент/множество на базата
        ]                                                -- Прочита текущия елемент от ба

#[ close the base ]
```

~~>(.

Придвижва се до първия елемент от текущото множество на 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                                    -- Актив
#[=;_processId_;#[~~^;##[_base_]]]                                  -- Начало на процес за
аза

        ##[~~>.;##[_base_];##[_processId_];плодове]        -- Придвижва се до елемента "п
одове"
        ##[~~>(.;##[_base_];##[_processId_]]                -- Придвижва се до първия еле
екущото множество
```

```
        #[cout;
                ##[~~^.;##[_base_];##[_processId_]]        ##[chr;9]- това е първият елем
кущото множество
        ]

#[~~.;##[_base_];##[_processId_]]                                  -- Край на процес за чет
 333 база
#[~~-;##[_base_]]                                                                  -- Освоб
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        #[ move to ;плодове]        -- Придвижва се до елемента "плодове"
        #[ first ]                              -- Придвижва се до първия елемент в тек
        #[cout;
                #[ get ]        #[ tab ]- това е първият елемент в текущото множество
        ]                                                -- Прочита текущия елемент от ба

#[ close the base ]
```

~~>).

Придвижва се до последния елемент от текущото множество на 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                    -- Актив
#[=;_processId_;#[~~^;##[_base_]]]                                   -- Начало на процес за
аза

        ##[~~>.;##[_base_];##[_processId_];плодове]        -- Придвижва се до елемента "п
одове"
        ##[~~>).;##[_base_];##[_processId_]]                -- Придвижва се до последния
в текущото множество
        #[cout;
                ##[~~^.;##[_base_];##[_processId_]]        ##[chr;9]- това е последният е
 текущото множество
        ]

#[~~.;##[_base_];##[_processId_]]                                    -- Край на процес за чет
 333 база
#[~~-;##[_base_]]                                                    -- Освоб
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        #[ move to ;плодове]        -- Придвижва се до елемента "плодове"
        #[ last ]                               -- Придвижва се до последния елемент в т
        #[cout;
                #[ get ]        #[ tab ]- това е последният елемент в текущото множество
        ]                                                -- Прочита текущия елемент от ба

#[ close the base ]
```

∼∼>+.

Придвижва се до следващия елемент от текущото множество на 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                                -- Актив
#[=;_processId_;#[~~^;##[_base_]]]                                  -- Начало на процес за
аза

        ##[~~>+.;##[_base_];##[_processId_]]                 -- Придвижва се до следващия
в текущото множество
        #[cout;
                ##[~~^.;##[_base_];##[_processId_]]        ##[chr;9]- това е вторият елем
кущото множество
        ]

#[~~.;##[_base_];##[_processId_]]                                  -- Край на процес за чет
 333 база
#[~~-;##[_base_]]                                                                -- Освоб
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        #[ next ]                                -- Придвижва се до следващия елемент в т
        #[cout;
                #[ get ]        #[ tab ]- това е вторият елемент в текущото множество
        ]                                                -- Прочита текущия елемент от ба

#[ close the base ]
```

~~>-.

Придвижва се до предишния елемент от текущото множество на 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                    -- Актив
#[=;_processId_;#[~~^;##[_base_]]]                          -- Начало на процес за
аза

        ##[~~>).;##[_base_];##[_processId_]]                -- Придвижва се до последния
в текущото множество
        ##[~~>-.;##[_base_];##[_processId_]]                -- Придвижва се до предишния
в текущото множество
        #[cout;
                ##[~~^.;##[_base_];##[_processId_]]        ##[chr;9]- това е предпоследни
нт в текущото множество
        ]

#[~~.;##[_base_];##[_processId_]]                             -- Край на процес за чет
 333 база
#[~~-;##[_base_]]                                                    -- Освоб
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        #[ last ]                                   -- Придвижва се до последния елемент в т
        #[ previous ]                           -- Придвижва се до предишния елемент в текущ
        #[cout;
                #[ get ]        #[ tab ]- това е предпоследният елемент в текущото множес
        ]                                                   -- Прочита текущия елемент от ба

#[ close the base ]
```

~~>?

Придвижва се до един от най-близките до търсен елемент от текущото множество на 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса. Третият параметър е търсеният елемент.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                                   -- Актив
#[=;_processId_;#[~~^;##[_base_]]]                                  -- Начало на процес за
аза

        ##[~~>?;##[_base_];##[_processId_];копър]        -- Придвижва се до един от най-б
изките до елемент в текущото множество
        #[cout;##[~~^.;##[_base_];##[_processId_]]]        -- Прочита текущия елемент от
азата

#[~~.;##[_base_];##[_processId_]]                                   -- Край на процес за чет
 333 база
#[~~-;##[_base_]]                                                                   -- Освоб
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        #[ closest ;копър]        -- Придвижва се до първия най-близък елемент в текущото
множество
        #[cout;#[ get ]]        -- Прочита текущия елемент от базата

#[ close the base ]
```

~~>(?

Придвижва се до първия най-близък елемент от текущото множество на 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса. Третият параметър е търсеният елемент.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                              -- Актив
#[=;_processId_;#[~~^;##[_base_]]]                                  -- Начало на процес за
аза

        ##[~~>(?;##[_base_];##[_processId_];копър]        -- Придвижва се до първия най-б
изък елемент в текущото множество
        #[cout;##[~~^.;##[_base_];##[_processId_]]]        -- Прочита текущия елемент от
азата

#[~~.;##[_base_];##[_processId_]]                                  -- Край на процес за чет
 333 база
#[~~-;##[_base_]]                                                              -- Освоб
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        #[ closest next ;копър]        -- Придвижва се до първия най-близък елемент в тек
щото множество
        #[cout;#[ get ]]                -- Прочита текущия елемент от базата

#[ close the base ]
```

~~>)?

Придвижва се до последния най-близък елемент от текущото множество на ЗЗЗ база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса. Третият параметър е търсеният елемент.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                          -- Актив
#[=;_processId_;#[~~^;##[_base_]]]                                  -- Начало на процес за
аза

        ##[~~>)?;##[_base_];##[_processId_];копър]        -- Придвижва се до последния на
-близък елемент в текущото множество
        #[cout;##[~~^.;##[_base_];##[_processId_]]]        -- Прочита текущия елемент от
азата

#[~~.;##[_base_];##[_processId_]]                                  -- Край на процес за чет
 333 база
#[~~-;##[_base_]]                                                          -- Освоб
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        #[ closest previous ;копър]        -- Придвижва се до последния най-близък елемен
 в текущото множество
        #[cout;#[ get ]]                          -- Прочита текущия елемент от базата

#[ close the base ]
```

4.1.5 Пътища в ЗЗЗ база

∼∼*...

Връща пътя до текущия елемент от множество на ЗЗЗ база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса. Третият параметър указва дали функцията да запази текущата позиция.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]
#[=;_processId_;#[~~^;##[_base_]]]                                              -- Нача
 база

        ##[~~>(?;##[_base_];##[_processId_];копър]                          -- Придвижва се
близък елемент в текущото множество
        #[cout;##[~~*...;##[_base_];##[_processId_];true]]        -- Извежда пътя до теку
ия елемент

#[~~.;##[_base_];##[_processId_]]                                               -- Край
 в ЗЗЗ база
#[~~-;##[_base_]]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        #[ closest next ;копър]                          -- Придвижва се до първия най-близ
кущото множество
        #[cout;#[ get the base path ]]        -- Извежда пътя до текущия елемент

#[ close the base ]
```

~~>...

Придвижва се до елемент по указания път в 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса. Третият параметър е пътят до елемента.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                        -- Активира връзка към 333 база
#[=;_processId_;#[~~^;##[_base_]]]        -- Начало на процес за четене от 333 база

        -- Придвижва се до елемент по указания път
        #[==;true;##[~~>...;##[_base_];##[_processId_];00000003ZZZ00000009зеленчуци\
];[
                        -- Извежда елемента, който е достигнат по указания път
                        #[cout;#[~~^.;##[_base_];##[_processId_]]]
                ];[
                        #[cout;Не успях да се придвижа по указания път.]
                ]
        ]

#[~~.;##[_base_];##[_processId_]]        -- Край на процес за четене или запис в 33\
3 база
#[~~-;##[_base_]]                                        -- Освобождава връзка към 333 ба
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        -- Придвижва се до елемент по указания път
        #[==;true;#[ move to the base path? ;00000003ZZZ00000009зеленчуци];[
                        -- Извежда елемента, който е достигнат по указания път
                        #[cout;#[ get ]]
                ];[
                        #[cout;Не успях да се придвижа по указания път.]
                ]
        ]

#[ close the base ]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        -- Придвижва се до елемент по указания път
        #[==;true;#[ move to the path? ;/зеленчуци];[
                        -- Извежда елемента, който е достигнат по указания път
                        #[cout;#[ get ]]
                ];[
                        #[cout;Не успях да се придвижа по указания път.]
                ]
        ]

#[ close the base ]
```

~~^...

Прочита елемент от указания път в ЗЗЗ база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът

на процеса. Третият параметър е пътят до елемента.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                        -- Активира връзка към 333 база
#[=;_processId_;#[~~^;##[_base_]]]        -- Начало на процес за четене от 333 база

        -- Извежда елемента, който е достигнат по указания път
        #[cout;##[~~^...;##[_base_];##[_processId_];00000003ZZZ00000009зеленчуци]]

#[~~.;##[_base_];##[_processId_]]        -- Край на процес за четене или запис в 33\
3 база
#[~~-;##[_base_]]                                        -- Освобождава връзка към 333 ба
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        -- Извежда елемента, който е достигнат по указания път
        #[cout;#[ get from the base path ;00000003ZZZ00000009зеленчуци]]

#[ close the base ]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        -- Извежда елемента, който е достигнат по указания път
        #[cout;#[ get from the path ;/зеленчуци]]

#[ close the base ]
```

∼∼?.

Проверява за наличието на елемент в текущото множество от ЗЗЗ база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса. Третият параметър е елемента.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                        -- Активира връзка към ЗЗЗ база
#[=;_processId_;#[~~^;##[_base_]]]        -- Начало на процес за четене от ЗЗЗ база

        -- Извежда true, ако елемента "зеленчуци" е в текущото
        -- множество на базата и false в противен случай
        #[cout;##[~~?.;##[_base_];##[_processId_];зеленчуци]]

#[~~.;##[_base_];##[_processId_]]        -- Край на процес за четене или запис в ЗЗ\
З база
#[~~-;##[_base_]]                                        -- Освобождава връзка към ЗЗЗ ба
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        -- Извежда true, ако елемента "зеленчуци" е в текущото
        -- множество на базата и false в противен случай
        #[cout;#[ check for ;зеленчуци]]

#[ close the base ]
```

∼∼?...

Проверява за наличието на път в текущото множество от ЗЗЗ база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса. Третият параметър е пътят в базата.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                        -- Активира връзка към 333 база
#[=;_processId_;#[~~^;##[_base_]]]        -- Начало на процес за четене от 333 база

        -- Извежда true, ако елемента "моркови" е в подмножеството "зеленчуци"
        -- от текущото множество на базата и false в противен случай
        #[cout;##[~~?...;##[_base_];##[_processId_];00000003ZZZ00000009зеленчуци000\
00007моркови]]

#[~~.;##[_base_];##[_processId_]]        -- Край на процес за четене или запис в 33\
3 база
#[~~-;##[_base_]]                                        -- Освобождава връзка към 333 ба
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        -- Извежда true, ако елемента "моркови" е в подмножеството "зеленчуци"
        -- от текущото множество на базата и false в противен случай
        #[cout;#[ check the base path ;00000003ZZZ00000009зеленчуци00000007моркови]]

#[ close the base ]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for reading ]

        -- Извежда true, ако елемента "моркови" е в подмножеството "зеленчуци"
        -- от текущото множество на базата и false в противен случай
        #[cout;#[ check the path ;/зеленчуци/моркови]]

#[ close the base ]
```

~~+...

Добавя елемент по указания път в 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса. Третият параметър е пътят до елемента завършващ със самия елемент.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                    -- Активира връз
#[=;_processId_;#[~~v;##[_base_];10000]]        -- Начало на процес за четене от 33\
3 база

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        ##[~~+...;##[_base_];##[_processId_];00000003ZZZ00000009зеленчуци00000009па\
тладжан]
        #[cout;##[~~^...;##[_base_];##[_processId_];00000003ZZZ00000009зеленчуци000\
00009патладжан]]

#[~~.;##[_base_];##[_processId_]]                        -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        #[ add the base path ;00000003ZZZ00000009зеленчуци00000009патладжан]
        #[cout;#[ get from the base path ;00000003ZZZ00000009зеленчуци00000009патла\
джан]]

#[ close the base ]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        #[ add the path ;/зеленчуци/патладжан]
        #[cout;#[ get from the path ;/зеленчуци/патладжан]]

#[ close the base ]
```

4.1.6 Символи в ЗЗЗ база

~~> -@

Придвижва се до предходния символ на елемент в ЗЗЗ база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                    -- Активира връз
#[=;_processId_;#[~~v;##[_base_];10000]]        -- Начало на процес за четене от 33\
3 база

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        ##[~~+...;##[_base_];##[_processId_];00000003ZZZ00000009зеленчуци00000012па\
тладжан]
        -- Придвижва се до предходния символ на елемент в 333 база
        #[cout;##[~~>-@;##[_base_];##[_processId_]]]

#[~~.;##[_base_];##[_processId_]]                          -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        #[ add the path ;/зеленчуци/патладжан]
        -- Придвижва се до предходния символ на елемент в 333 база
        #[cout;#[ previous symbol? ]]

#[ close the base ]
```

~~>+@

Придвижва се до следващия символ на елемент в 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                    -- Активира връз
#[=;_processId_;#[~~v;##[_base_];10000]]        -- Начало на процес за четене от 33\
3 база

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        ##[~~+...;##[_base_];##[_processId_];00000003ZZZ00000009зеленчуци00000012па\
тладжан]
        -- Придвижва се до следващия символ на елемент в 333 база
        #[cout;##[~~>+@;##[_base_];##[_processId_]]]

#[~~.;##[_base_];##[_processId_]]                          -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        #[ add the path ;/зеленчуци/патладжан]
        -- Придвижва се до следващия символ на елемент в 333 база
        #[cout;#[ next symbol? ]]

#[ close the base ]
```

~~>^@

Придвижва се до горния символ на елемент в 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                    -- Активира връз
#[=;_processId_;#[~~v;##[_base_];10000]]        -- Начало на процес за четене от 33\
3 база

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        ##[~~+...;##[_base_];##[_processId_];00000003ZZZ00000009зеленчуци00000012па\
тладжан]
        -- Придвижва се до горния символ на елемент в 333 база
        #[cout;##[~~>^@;##[_base_];##[_processId_]]]

#[~~.;##[_base_];##[_processId_]]                          -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        #[ add the path ;/зеленчуци/патладжан]
        -- Придвижва се до горния символ на елемент в 333 база
        #[cout;#[ up symbol? ]]

#[ close the base ]
```

~~>v@

Придвижва се до долния символ на елемент в 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                    -- Активира връз
#[=;_processId_;#[~~v;##[_base_];10000]]        -- Начало на процес за четене от 33\
3 база

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        ##[~~+...;##[_base_];##[_processId_];00000003ZZZ00000009зеленчуци00000012па\
тладжан]
        -- Придвижва се до долния символ на елемент в 333 база
        #[cout;##[~~>v@;##[_base_];##[_processId_]]]

#[~~.;##[_base_];##[_processId_]]                          -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        #[ add the path ;/зеленчуци/патладжан]
        -- Придвижва се до долния символ на елемент в 333 база
        #[cout;#[ down symbol? ]]

#[ close the base ]
```

4.1.7 Преобразуване на пътища от 333 база

~~///...

Преобразува път с разделители към път в 333 база

Първият параметър е пътят с разделители. Вторият параметър е разделителят.

пример:

```
#[cout;
        ##[~~///...;/зеленчуци/патладжан;/]##[chr;10]
        ##[~~///...;/зеленчуци/моркови;/]
]
```

~~...///

Преобразува път в ЗЗЗ база към път с разделители

Първият параметър е разделителят. Вторият параметър е пътят.

пример:

```
#[cout;
        ##[~~.../// ;/;[00000003ZZZ00000004test]]##[chr;10]
        ##[~~.../// ;/;[0000000000000009зеленчуци00000009патладжан]]##[chr;10]
        ##[~~.../// ;/;[0000000000000009зеленчуци00000007моркови]]
]
```

4.1.8 Заключване и отключване на елементи за запис и четене от ЗЗЗ база

~~#v

Заключва елемент за запис (изтриване) в ЗЗЗ база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                    -- Активира връз
#[=;_processId_;#[~~v;##[_base_];10000]]        -- Начало на процес за четене от 33\
3 база

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        ##[~~+...;##[_base_];##[_processId_];00000003ZZZ00000009зеленчуци00000009па\
тладжан]
        -- Заключва текущия елемент за запис в 333 база
        ##[~~#v;##[_base_];##[_processId_]]

#[~~.;##[_base_];##[_processId_]]                          -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        #[ add the path ;/зеленчуци/патладжан]
        -- Заключва текущия елемент за запис в 333 база
        #[ lock for write ]

#[ close the base ]
```

~~$v

Отключва елемент за запис (изтриване) в 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                    -- Активира връз
#[=;_processId_;#[~~v;##[_base_];10000]]        -- Начало на процес за четене от 33\
3 база

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        ##[~~+...;##[_base_];##[_processId_];00000003ZZZ00000009зеленчуци00000009па\
тладжан]
        -- Отключва текущия елемент за запис в 333 база
        ##[~~$v;##[_base_];##[_processId_]]

#[~~.;##[_base_];##[_processId_]]                          -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        #[ add the path ;/зеленчуци/патладжан]
        -- Отключва текущия елемент за запис в 333 база
        #[ unlock for write ]

#[ close the base ]
```

∼∼#^

Заключва елемент за четене от 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                    -- Активира връз
#[=;_processId_;#[~~v;##[_base_];10000]]        -- Начало на процес за четене от 33\
3 база

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        ##[~~+...;##[_base_];##[_processId_];00000003ZZZ00000009зеленчуци00000009па\
тладжан]
        -- Заключва текущия елемент за четене от 333 база
        ##[~~#^;##[_base_];##[_processId_]]

#[~~.;##[_base_];##[_processId_]]                          -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        #[ add the path ;/зеленчуци/патладжан]
        -- Заключва текущия елемент за четене от 333 база
        #[ lock for read ]

#[ close the base ]
```

~~$^

Отключва елемент за четене от 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                    -- Активира връз
#[=;_processId_;#[~~v;##[_base_];10000]]        -- Начало на процес за четене от 33\
3 база

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        ##[~~+...;##[_base_];##[_processId_];00000003ZZZ00000009зеленчуци00000009па\
тладжан]
        -- Отключва текущия елемент за четене от 333 база
        ##[~~$^;##[_base_];##[_processId_]]

#[~~.;##[_base_];##[_processId_]]                          -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        -- Добавя елемента "патладжан" в подмножеството "зеленчуци"
        #[ add the path ;/зеленчуци/патладжан]
        -- Отключва текущия елемент за четене от 333 база
        #[ unlock for read ]

#[ close the base ]
```

~~›./.

Премества множество от път към друг път в 333 база

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса. Третият параметър е пътят до елемента завършващ със самия

елемент. Четвъртият параметър е пътят където искаме да преместим елемента.

пример:

```
#[.=;_base_;docbases/zzzbase]

#[~~+;##[_base_]]                                                    -- Активира връз
#[=;_processId_;#[~~v;##[_base_];10000]]        -- Начало на процес за четене от 33\
3 база

        -- добавям нов елемент "растения"
        #[~~+.;##[_base_];##[_processId_];растения]

        -- прехвърлям множеството "зеленчуци" в "растения"
        ##[~~>./.;##[_base_];##[_processId_];
                ##[~~///...;/зеленчуци;/];
                ##[~~///...;/растения;/]
        ]
        -- прехвърлям множеството "плодове" в "растения"
        ##[~~>./.;##[_base_];##[_processId_];
                ##[~~///...;/плодове;/];
                ##[~~///...;/растения;/]
        ]

#[~~.;##[_base_];##[_processId_]]                        -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в
#[cout;
        #[ZZZ2SimpleTXTFromDividerPath;##[_base_];;/]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        -- добавям нов елемент "растения"
        #[ add ;растения]

        -- прехвърлям множеството "зеленчуци" в "растения"
        #[ move the base path ;
                0000000000000009зеленчуци;
                0000000000000008растения
        ]
        -- прехвърлям множеството "плодове" в "растения"
        #[ move the base path ;
                0000000000000007плодове;
                0000000000000008растения
        ]

#[ close the base ]
#[ get the base as a simple text ]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ open the base for writing ]

        -- добавям нов елемент "растения"
        #[ add ;растения]

        -- прехвърлям множеството "зеленчуци" в "растения"
        #[ move the path ;
                /зеленчуци;
                /растения
        ]
        -- прехвърлям множеството "плодове" в "растения"
        #[ move the path ;
                /плодове;
                /растения
```

```
        ]

#[ close the base ]
#[ get the base as a simple text ]
```

4.1.9 Команди за ограничаване на достъпа до база

~~u+

Добавя потребител с парола и режим за достъп до текущата база и при успех връща `true`

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса за запис. Третият параметър е името на потребителя. Четвъртият параметър е паролата на потребителя. Петият параметър е режима за достъп на потребителя до текущата база и има една от следните стойности:`admin` - има право да променя и добавя потребители, да записва и чете; `write` - има право да записва и чете; `read` - има право да чете; всички останали стойности ограничават достъпа на потребителя до базата.

Желателно е да добавите поне един потребител с режим на достъп `admin`, ако не го добавите няма да можете да променяте информацията за потребителите след края на текущия процес за запис в базата! Ако имате поне един потребител с режим `write` ще можете да изтриете служебното множество с потребителите на базата. Добавянето на потребители само с режим на достъп `read` е равносилно на заключване на базата за запис! Добавянето на потребители само с режим на достъп различен от `admin`, `write`, `read` е равносилно на заключване на базата за четене и запис. Така че преди да заключите база по такъв начин е препоръчително да й направите копие.

пример:

```
#[.=;_base_;docbases/zzzprotectedbase]

#[~~+;##[_base_]]                                                    -- Активира връз
#[.=;_processId_;#[~~v;##[_base_];10000]]        -- Начало на процес за запис в 333\
 база

        ##[~~u+;##[_base_];##[_processId_];[test];[1234];admin]

#[~~.;##[_base_];##[_processId_]]                          -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в

#[~~+;##[_base_]]                                                    -- Активира връз
#[.=;_processId_;#[~~v;##[_base_];10000;[test];[1234]]]        -- Начало на процес \
за запис в 333 база

        ##[~~+.;##[_base_];##[_processId_];елемент 1]
        ##[~~+.;##[_base_];##[_processId_];елемент 2]
        ##[~~+.;##[_base_];##[_processId_];елемент 3]

        #[cout;
                #[==;true;##[~~>(.;##[_base_];##[_processId_]];;]
                ##[~~^.;##[_base_];##[_processId_]]        ##[chr;10]
                #[==;true;##[~~>+.;##[_base_];##[_processId_]];;]
                ##[~~^.;##[_base_];##[_processId_]]        ##[chr;10]
                #[==;true;##[~~>+.;##[_base_];##[_processId_]];;]
                ##[~~^.;##[_base_];##[_processId_]]
        ]

#[~~.;##[_base_];##[_processId_]]                          -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ base ;docbases/zzzprotectedbase]
#[ open the base for writing ]
        #[ add the user ;test;1234;admin]
#[ close the base ]

#[ user and password ;test;1234]
#[ open the base for writing ]
        #[ add ;елемент 1]#[ add ;елемент 2]#[ add ;елемент 3]
        #[cout;
                #[ first ]        #[ get ]#[ new line ]
                #[ next ]        #[ get ]#[ new line ]
                #[ next ]        #[ get ]
        ]
#[ close the base ]
```

~~u/

Променя името или паролата на потребител и при успех връща `true`

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса за запис. Третият параметър е текущото име на потребителя. Четвъртият параметър е текущата парола на потребителя. Петият параметър е новото име на потребителя. Шестият параметър е новата парола на потребителя.

пример:

```
#[.=;_base_;docbases/zzzprotectedbase]

#[~~+;##[_base_]]                                                    -- Активира връз
#[.=;_processId_;#[~~v;##[_base_];10000;[test];[1234]]]        -- Начало на процес \
за запис в 333 база

        ##[~~u+;##[_base_];##[_processId_];[user2];[123456];admin]
        ##[~~u/;##[_base_];##[_processId_];[user2];[123456];[user3];[654321]]

#[~~.;##[_base_];##[_processId_]]                          -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в

#[~~+;##[_base_]]                                                    -- Активира връз
#[.=;_processId_;#[~~v;##[_base_];10000;[user3];[654321]]]        -- Начало на проц\
ес за запис в 333 база

        ##[~~+.;##[_base_];##[_processId_];елемент 4]

        #[cout;
                #[==;true;##[~~>(.;##[_base_];##[_processId_]];;]
                ##[~~^.;##[_base_];##[_processId_]]        ##[chr;10]
                #[==;true;##[~~>+.;##[_base_];##[_processId_]];;]
                ##[~~^.;##[_base_];##[_processId_]]        ##[chr;10]
                #[==;true;##[~~>+.;##[_base_];##[_processId_]];;]
                ##[~~^.;##[_base_];##[_processId_]]        ##[chr;10]
                #[==;true;##[~~>+.;##[_base_];##[_processId_]];;]
                ##[~~^.;##[_base_];##[_processId_]]
        ]

#[~~.;##[_base_];##[_processId_]]                          -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ base ;docbases/zzzprotectedbase]
#[ user and password ;test;1234]
#[ open the base for writing. ;[
        #[ add the user ;user2;123456;admin]
        #[ change the user ;user2;123456;user3;654321]
]]
#[ user and password ;user3;654321]
#[ open the base for writing. ;[
        #[ add ;елемент 4]
        #[cout;
                #[ first ]        #[ get ]#[ new line ]
                #[ next ]        #[ get ]#[ new line ]
                #[ next ]        #[ get ]#[ new line ]
                #[ next ]        #[ get ]
        ]
]]
```

~~u-

Изтрива потребител и при успех връща `true`

Първият параметър е името на базата. Вторият параметър е номерът/идентификаторът на процеса за запис. Третият параметър е текущото име на потребителя. Четвъртият параметър е текущата парола на потребителя.

пример:

```
#[.=;_base_;docbases/zzzprotectedbase]

#[~~+;##[_base_]]                                                    -- Активира връз
#[.=;_processId_;#[~~v;##[_base_];10000;[test];[1234]]]        -- Начало на процес \
за запис в 333 база

        ##[~~u-;##[_base_];##[_processId_];[user3];[654321];admin]

#[~~.;##[_base_];##[_processId_]]                          -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в

#[~~+;##[_base_]]                                                    -- Активира връз
#[.=;_processId_;#[~~v;##[_base_];10000;[user3];[654321]]]        -- Начало на проц\
ес за запис в 333 база

        ##[~~+.;##[_base_];##[_processId_];елемент 4]

        #[cout;
                #[==;true;##[~~>(.;##[_base_];##[_processId_]];;]
                ##[~~^.;##[_base_];##[_processId_]]        ##[chr;10]
                #[==;true;##[~~>+.;##[_base_];##[_processId_]];;]
                ##[~~^.;##[_base_];##[_processId_]]        ##[chr;10]
                #[==;true;##[~~>+.;##[_base_];##[_processId_]];;]
                ##[~~^.;##[_base_];##[_processId_]]        ##[chr;10]
                #[==;true;##[~~>+.;##[_base_];##[_processId_]];;]
                ##[~~^.;##[_base_];##[_processId_]]
        ]

#[~~.;##[_base_];##[_processId_]]                          -- Край на процес за четене или
333 база
#[~~-;##[_base_]]                                                    -- Освобождава в
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ base ;docbases/zzzprotectedbase]
#[ user and password ;test;1234]#[>=n;#[ open the base for writing? ];0;[
        #[ remove the user ;user3;654321;admin]
        #[ close the base ]
];]
#[ user and password ;user3;654321]#[>=n;#[ open the base for writing? ];0;[
        #[ add ;елемент 4]
        #[cout;
                #[ first ]        #[ get ]#[ new line ]
                #[ next ]        #[ get ]#[ new line ]
                #[ next ]        #[ get ]#[ new line ]
                #[ next ]        #[ get ]
        ]
        #[ close the base ]
];]
```

5. Глава III. ЗЗЗ++

ЗЗЗ++ е допълнителна функционалност, която позволява работа с класове и обекти, които живеят в мрежата от ЗЗЗ сървъри и ЗЗЗ бази. Това е изключително мощна функционалност, която дава нов поглед върху разработката на приложения. Те не са ограничени от размера на оперативната или дискова памет на един компютър, а могат да работят едновременно на множество компютри свързани в мрежа.

5.1 Команди за управление на класове

Примери за използването на следващите команди има в папката "`libs/src`" от демонстрационния пакет описан в раздела "Какво ни е необходимо преди да продължим?".

`::+`

Добавя клас

параметри:

`{CLASSBASE}`
`{CLASSNAME}`

пример:

```
#[.=;_class_base_;docbases/classes.zzz]
#[.=;_class_;/MyClass]

#[::-;##[_class_base_];##[_class_]]
#[::+;##[_class_base_];##[_class_]]

#[cout;
        #[ZZZ2SimpleTXTFromDividerPath;##[_class_base_];;/]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ class base ;docbases/classes.zzz]
#[ class ;/MyClass]

#[ remove class ]
#[ add class ]

#[ base ;##[_class_base_]]
#[ get the base as a simple text ]
```

::-

Изтрива клас

параметри:

{CLASSBASE}
{CLASSNAME}

пример:

```
#[.=;_class_base_;docbases/classes.zzz]
#[.=;_class_;/MyClass]

#[::-;##[_class_base_];##[_class_]]
#[::+;##[_class_base_];##[_class_]]

#[cout;
        #[ZZZ2SimpleTXTFromDividerPath;##[_class_base_];;/]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ class base ;docbases/classes.zzz]
#[ class ;/MyClass]

#[ remove class ]
#[ add class ]

#[ base ;##[_class_base_]]
#[ get the base as a simple text ]
```

::+#

Добавя коментар към класа

параметри:

{CLASSBASE}
{CLASSNAME}
{PATH}
{CONTENT}

пример:

```
#[.=;_class_base_;docbases/classes.zzz]
#[.=;_class_;/MyClass]

#[::-;##[_class_base_];##[_class_]]
#[::+;##[_class_base_];##[_class_]]

#[::+#;##[_class_base_];##[_class_];Copyright;##[COPYRIGHT_ZZZ]]
#[::+#;##[_class_base_];##[_class_];history/2016-08-07;създаване]

#[cout;
        #[ZZZ2SimpleTXTFromDividerPath;##[_class_base_];;/]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ class base ;docbases/classes.zzz]
#[ class ;/MyClass]

#[ remove class ]
#[ add class ]

#[ add comment ;;Copyright;##[COPYRIGHT_ZZZ]]
#[ add comment ;;history/2016-08-07;създаване]

#[ base ;##[_class_base_]]
#[ get the base as a simple text ]
```

::+^

Добавя родителски клас

параметри:

{CLASSBASE}
{CLASSNAME}
{PARENTBASE}
{PARENTNAME}

пример:

```
#[.=;_class_base_;docbases/classes.zzz]

#[.=;_class_;/MyParentClass]
#[::-;##[_class_base_];##[_class_]]
#[::+;##[_class_base_];##[_class_]]

#[.=;_class_;/MyClass]
#[::-;##[_class_base_];##[_class_]]
#[::+;##[_class_base_];##[_class_]]
```

```
#[::+^;##[_class_base_];##[_class_];##[_class_base_];/MyParentClass]

#[cout;
        #[ZZZ2SimpleTXTFromDividerPath;##[_class_base_];;/]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ class base ;docbases/classes.zzz]

#[ class ;/MyParentClass]
#[ remove class ]
#[ add class ]

#[ class ;/MyClass]
#[ remove class ]
#[ add class ]

#[ add parent ;##[_class_base_];/MyParentClass]

#[ base ;##[_class_base_]]
#[ get the base as a simple text ]
```

::-^

Изтрива родителски клас

параметри:

{CLASSBASE}
{CLASSNAME}
{PARENTBASE}
{PARENTNAME}

пример:

```
#[.=;_class_base_;docbases/classes.zzz]

#[.=;_class_;/MyParentClass]
#[::-;##[_class_base_];##[_class_]]
#[::+;##[_class_base_];##[_class_]]

#[.=;_class_;/MyClass]
#[::-;##[_class_base_];##[_class_]]
#[::+;##[_class_base_];##[_class_]]

#[::+^;##[_class_base_];##[_class_];##[_class_base_];/MyParentClass]
#[::-^;##[_class_base_];##[_class_];##[_class_base_];/MyParentClass]

#[cout;
        #[ZZZ2SimpleTXTFromDividerPath;##[_class_base_];;/]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ class base ;docbases/classes.zzz]

#[ class ;/MyParentClass]
#[ remove class ]
#[ add class ]

#[ class ;/MyClass]
#[ remove class ]
#[ add class ]

#[ add parent ;##[_class_base_];/MyParentClass]
#[ remove parent ;##[_class_base_];/MyParentClass]

#[ base ;##[_class_base_]]
#[ get the base as a simple text ]
```

::+.

Добавя метод към клас

параметри:

{CLASSBASE}
{CLASSNAME}
{CLASSMETHODNAME}
{CLASSMETHODPARAMNAMES}
{CLASSMETHODBODY}
{CLASSMETHODPARAMS}

пример:

```
#[.=;_class_base_;docbases/classes.zzz]

#[.=;_class_;/MyClass]
#[::-;##[_class_base_];##[_class_]]
#[::+;##[_class_base_];##[_class_]]

#[::+#;##[_class_base_];##[_class_];Copyright;##[COPYRIGHT_ZZZ]]
#[::+#;##[_class_base_];##[_class_];history/2016-08-07;създаване]

#[::+.;##[_class_base_];##[_class_];constructors/public/constructor;
        [{OBJECTBASE};{OBJECTNAME};{CLASSBASE};{CLASSNAME}];
        [
                #[cout;
                        object base: {OBJECTBASE}##[chr;10]
                        object name: {OBJECTNAME}##[chr;10]
                        method: constructor
                ]
        ];
]
#[::+#;##[_class_base_];##[_class_]/constructors/public/constructor;history/\
2016-08-07;[created default constructor]]

#[::+.;##[_class_base_];##[_class_];destructors/public/destructor;
        [{OBJECTBASE};{OBJECTNAME};{CLASSBASE};{CLASSNAME}];
```

```
        [
                #[cout;
                        object base: {OBJECTBASE}##[chr;10]
                        object name: {OBJECTNAME}##[chr;10]
                        method: destructor
                ]
        ];
]
#[::+#;##[_class_base_];##[_class_]/destructors/public/destructor;history/20\
16-08-07;[created default destructor]]

#[::+.;##[_class_base_];##[_class_];methods/public/test;
        [{OBJECTBASE};{OBJECTNAME};{CLASSBASE};{CLASSNAME}];
        [
                #[cout;
                        object base: {OBJECTBASE}##[chr;10]
                        object name: {OBJECTNAME}##[chr;10]
                        method: test
                ]
        ];
]
#[::+#;##[_class_base_];##[_class_]/methods/public/test;history/2016-09-04;[\
created method test]]

#[::+;##[_class_base_];##[_class_]]

#[cout;
        #[ZZZ2SimpleTXTFromDividerPath;##[_class_base_];;/]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ class base ;docbases/classes.zzz]

#[ class ;/MyClass]
#[ remove class ]
#[ add class ]

#[ add comment ;;Copyright;##[COPYRIGHT_ZZZ]]
#[ add comment ;;history/2016-08-07;създаване]

#[ add method ;constructors/public/constructor;
        [{OBJECTBASE};{OBJECTNAME};{CLASSBASE};{CLASSNAME}];
        [
                #[cout;
                        object base: {OBJECTBASE}#[ new line ]
                        object name: {OBJECTNAME}#[ new line ]
                        method: constructor
                ]
        ];
]
#[ add comment ;/constructors/public/constructor;
        history/2016-08-07;
        [created default constructor]]

#[ add method ;destructors/public/destructor;
        [{OBJECTBASE};{OBJECTNAME};{CLASSBASE};{CLASSNAME}];
        [
                #[cout;
                        object base: {OBJECTBASE}#[ new line ]
                        object name: {OBJECTNAME}#[ new line ]
                        method: destructor
                ]
        ];
]
#[ add comment ;/destructors/public/destructor;
        history/2016-08-07;
        [created default destructor]]
```

```
#[ add method ;methods/public/test;
        [{OBJECTBASE};{OBJECTNAME};{CLASSBASE};{CLASSNAME}];
        [
                #[cout;
                        object base: {OBJECTBASE}#[ new line ]
                        object name: {OBJECTNAME}#[ new line ]
                        method: test
                ]
        ];
]
#[ add comment ;/methods/public/test;
        history/2016-09-04;
        [created method test]]

#[ add class ]

#[ base ;##[_class_base_]]
#[ get the base as a simple text ]
```

::-.

Изтрива метод от клас

параметри:

{CLASSBASE}
{CLASSNAME}
{CLASSMETHODNAME}
{CLASSMETHODPARAMS}

пример:

```
#[.=;_class_base_;docbases/classes.zzz]

#[.=;_class_;/MyClass]

#[::-.;##[_class_base_];##[_class_];constructors/public/constructor;]
#[::-.;##[_class_base_];##[_class_];destructors/public/destructor;]

#[cout;
        #[ZZZ2SimpleTXTFromDividerPath;##[_class_base_];;/]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ class base ;docbases/classes.zzz]

#[ class ;/MyClass]

#[ remove method ;constructors/public/constructor;]
#[ remove method ;destructors/public/destructor;]

#[ base ;##[_class_base_]]
#[ get the base as a simple text ]
```

::>.

Изпълнява метод от клас

параметри:

{CLASSBASE}
{CLASSNAME}
{CLASSMETHODNAME}
{CLASSMETHODPARAMS}

пример:

```
#[.=;_class_base_;docbases/classes.zzz]
#[.=;_object_base_;docbases/objects.zzz]

#[::>.;
        ##[_class_base_];/MyClass;methods/public/test;
        [{OBJECTBASE};{OBJECTNAME};{CLASSBASE};{CLASSNAME}]
]
```

5.2 Команди за управление на обекти

new-object

Създава нов обект

параметри:

```
{CLASSBASE}
{CLASSNAME}
{OBJECTBASE}
{OBJECTNAME}
{CLASSMETHODPARAMS}
```

пример:

```
#[.=;_class_base_;docbases/classes.zzz]
#[.=;_object_base_;docbases/objects.zzz]

#[new-object;
        ##[_class_base_];/MyClass;
        ##[_object_base_];/MyObject;
        [{OBJECTBASE};{OBJECTNAME};{CLASSBASE};{CLASSNAME}]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ class base ;docbases/classes.zzz]
#[ object base ;docbases/objects.zzz]

#[new-object;
        #[ class base ];/MyClass;
        #[ object base ];/MyObject;
        [{OBJECTBASE};{OBJECTNAME};{CLASSBASE};{CLASSNAME}]
]
```

Извеждане на базата с класовете чрез команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ base ;docbases/classes.zzz]
#[ get the base as a simple text ]
```

:>.

Изпълнява метод от обект

параметри:

{OBJECTBASE}
{OBJECTNAME}
{CLASSMETHODNAME}
{CLASSMETHODPARAMS}

пример:

```
#[.=;_class_base_;docbases/classes.zzz]
#[.=;_object_base_;docbases/objects.zzz]

#[:>.;
        ##[_object_base_];/MyObject;methods/public/test;
        [{OBJECTBASE};{OBJECTNAME};{CLASSBASE};{CLASSNAME}]
]
```

delete-object

Изтрива обект

параметри:

```
{OBJECTBASE}
{OBJECTNAME}
{CLASSMETHODPARAMS}
```

пример:

```
#[.=;_class_base_;docbases/classes.zzz]
#[.=;_object_base_;docbases/objects.zzz]

#[delete-object;
        ##[_object_base_];/MyObject;
        [{OBJECTBASE};{OBJECTNAME};{CLASSBASE};{CLASSNAME}]
]
```

Горната програма написана с команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ class base ;docbases/classes.zzz]
#[ object base ;docbases/objects.zzz]

#[delete-object;
        #[ object base ];/MyObject;
        [{OBJECTBASE};{OBJECTNAME};{CLASSBASE};{CLASSNAME}]
]
```

Извеждане на базата с класовете чрез команди за опростен достъп до базата реализирани в библиотеката: simplify.txt:

```
#[ base ;docbases/classes.zzz]
#[ get the base as a simple text ]
```

5.3 Команди за деклариране на 333++ програма

actualize-libs

Компилира променените класове, за които има изходен програмен код

пример:

```
#[actualize-libs]
```

6. Глава IV. Команди реализирани със средствата на езика за управление на базите

Това към момента са команди реализиращи различни видове цикли и такива за архивиране и разархивиране на база или части от нея.

6.1 Команди разширяващи възможностите на езика

Примери за използването на следващите команди има в папката "`libs/src`" от демонстрационния пакет описан в раздела "Какво ни е необходимо преди да продължим?".

6.1.1 Цикли

6.1.1.1 Команда `{ ?. }` за цикъл `ДОКАТО`. По аналогия с цикъла `while` в други програмни езици.

`{ ?. }`

Цикъл `ДОКАТО`
Проверява дали резултата от условието е `true` и ако е така изпълнява тялото на цикъла в противен случай продължава с изпълнението на следващата команда.

параметри:

`{CONDITION}` - условие
`{BODY}` - тяло на цикъла

реализация на ДОКАТО:

```
#[=;{ ?. };[
                #[==;true;{CONDITION};[
                                {BODY}

                                #[{ ?. };[{CONDITION}];[{BODY}]]
                        ];
                ]
        ]
]
#[=.;{ ?. };{CONDITION};{BODY}]
```

пример за ДОКАТО:

```
#[cout;
        #[.=;i;1]
        #[{ ?. };
                [        #[>=n;5;##[i];[true];]        ];                        #[#; <==
                [
                        ##[i]##[chr;10]

                        #[.=;i;#[fmtd;%0.0f;#[+;##[i];1]]]
                ]
        ]
]
```

6.1.1.2 Команда `{ .? }` за цикъл ПРАВИ ДОКАТО. По аналогия с цикъла `do while` в други програмни езици.

{ .? }

Цикъл ПРАВИ ДОКАТО
Изпълнява тялото на цикъла, след което проверява дали резултата от условието е `true` и ако е така отново изпълнява тялото на цикъла в противен случай продължава с изпълнението на следващата команда.

параметри:

```
{BODY} - тяло на цикъла
{CONDITION} - условие
```

реализация на ПРАВИ ДОКАТО:

```
#[=;{ .? };[
                {BODY}

                #[==;true;{CONDITION};[
                                #[{ .? };[{BODY}];[{CONDITION}]]
                        ];
                ]
        ]
]
#[=.;{ .? };{BODY};{CONDITION}]
```

използване на ПРАВИ ДОКАТО:

```
#[cout;
        #[ base ;docbases/zzzbase]
        #[ open the base for reading. ;[
                #[{ .? };
                        [                                                       #[#; <==
                                #[ get ]#[ new line ]
                        ];
                        [        #[ next? ]        ]                    #[#; <== COND
                ]
        ]]
]
```

6.1.1.3 Команда { _?._ } за цикъл ПРАВИ ЗА ВСЕКИ. По аналогия с цикъла for в други програмни езици.

{ _?._ }

Цикъл ПРАВИ ЗА ВСЕКИ
1. Инициализира цикъла.
2. Проверява дали резултата от условието е `true` и ако е така изпълнява тялото на цикъла.
3. Приключва цикъла.

параметри:

`{INIT}` - подготовка преди началото на цикъла
`{CONDITION}` - условие
`{BODY}` - тяло на цикъла
`{CLEAN}` - приключване след края на цикъла

реализация на ПРАВИ ЗА ВСЕКИ:

```
#[=;{ _?._ };[
                {INIT}

                #[{ ?. };[{CONDITION}];[{BODY}]]

                {CLEAN}
        ]
]
#[=.;{ _?._ };{INIT};{CONDITION};{BODY};{CLEAN}]
```

използване на ПРАВИ ЗА ВСЕКИ:

```
#[cout;
        #[{ _?._ };
                [
                        #[.=;i;1]
                ];
                [         #[>=n;5;##[i];[true];]         ];                          #[#; <==
                [
                        ##[i]##[chr;10]
```

```
                        #[.=;i;#[fmtd;%0.0f;#[+;##[i];1]]]
                ];
                [
                ]
        ]
]
```

{ _.?_ }

Цикъл ПРАВИ ЗА ВСЕКИ, като тялото му се изпълнява поне веднъж
1. Инициализира цикъла.
2. Изпълнява тялото на цикъла, проверява дали резултата от условието е `true` и ако е така отново изпълнява цикъла.
3. Приключва цикъла.

параметри:

`{INIT}` - подготовка преди началото на цикъла
`{BODY}` - тяло на цикъла
`{CONDITION}` - условие
`{CLEAN}` - приключване след края на цикъла

реализация на ПРАВИ ЗА ВСЕКИ - поне веднъж:

```
#[=;{ _.?_ };[
                {INIT}

                #[{ .? };[{BODY}];[{CONDITION}]]

                {CLEAN}
        ]
]
#[=.;{ _.?_ };{INIT};{BODY};{CONDITION};{CLEAN}]
```

използване на ПРАВИ ЗА ВСЕКИ - поне веднъж:

```
##[cout;
        #[{ _.?_ };
                [
                        #[ base ;docbases/zzzbase]
                        #[ open the base for reading ]
                ];
                [
                        #[ get ]##[chr;10]
                ];
                [        #[ next? ]        ];
                [
                        #[ close the base ]
                ]
        ]
]
```

6.2 Команди за управление на базите

Команди за архивиране и разархивиране на база или части от нея.

6.2.1 Запис на множество към текст (UTF-8)

ZZZ2TXT

Записва текущото множество от указаната база към текст

параметри:

{BASE} - името на базата, в която е множеството
{PID} - идентификатор на процеса за четене или запис

реализация:

```
#[=;ZZZ2TXTRecursionForEach;[
                ##[fmtl;[%016I64X];{LEVEL}],
                ##[=;ELEMENT;##[~~^.;{BASE};{PID}]]
                ##[fmtl;[%016I64X];##[(..);ELEMENT]],
                #[repeatPrint;##[chr;32];{LEVEL}]
                ##[ELEMENT]
                #[=x;ELEMENT]
                ##[chr;10]
                #[==;true;##[~~>v;{BASE};{PID}];[

                                #[ZZZ2TXTRecursion;{BASE};{PID};#[fmtd;%0.0f;##[+;1;{LEVE

                                #[==;true;##[~~>^;{BASE};{PID}];;]
                        ];
                ]
                #[==;true;##[~~>+.;{BASE};{PID}];[
                                #[ZZZ2TXTRecursionForEach;{BASE};{PID};{LEVEL}]
                        ];
                ]
        ]
]
#[=.;ZZZ2TXTRecursionForEach;{BASE};{PID};{LEVEL}]

#[=;ZZZ2TXTRecursion;[
                #[==;true;##[~~>(.;{BASE};{PID}];[
                                #[ZZZ2TXTRecursionForEach;{BASE};{PID};{LEVEL}]
                        ];
                ]
        ]
]
#[=.;ZZZ2TXTRecursion;{BASE};{PID};{LEVEL}]

#[=;ZZZ2TXT;[
                #[==;true;##[~~>v;{BASE};{PID}];[

                                #[ZZZ2TXTRecursion;{BASE};{PID};0]
```

```
                                #[==;true;##[~~>^;{BASE};{PID}];;]
                        ];
                ]
        ]
]
#[=.;ZZZ2TXT;{BASE};{PID}]
```

използване:

```
##[cout;
        #[.=;_base_;docbases/zzzbase]

        #[~~+;##[_base_]]
        #[.=;_^_;#[~~^;##[_base_]]]

        #[ZZZ2TXT;##[_base_];##[_^_]]

        #[~~.;##[_base_];##[_^_]]
        #[~~-;##[_base_]]
]
```

ZZZ2TXTFromPath

Записва множество от указаната база и път към текст

параметри:

{BASE} - името на базата, в която е множеството
{PATH} - път в базата

реализация:

```
#[=;ZZZ2TXTFromPath;[
                #[~~+;{BASE}]

                #[=;pIdR_ZZZ2TXTFromPath;##[~~^;{BASE}]]

                #[==;true;##[~~>...;{BASE};##[pIdR_ZZZ2TXTFromPath];{PATH};true];[
                                #[ZZZ2TXT;{BASE};##[pIdR_ZZZ2TXTFromPath]]
                        ];
                ]
                ##[~~.;{BASE};##[pIdR_ZZZ2TXTFromPath]]
                #[~~-;{BASE}]
        ]
]
#[=.;ZZZ2TXTFromPath;{BASE};{PATH}]
```

използване:

```
##[cout;
        #[ZZZ2TXTFromPath;docbases/zzzbase;]
]
```

Формата на един ред от текстовия резултат е следния:

[ниво на вложеност (64 битово шестнадесетично число)][,][дължина на елемента(64 битово шестнадесетично число)][,][подравняващи символи с брой равен на нивото на вложеност][елемент]

Пример: `0000000000000000,000000000000000F,Интернет адреси`

`0000000000000000` - главно ниво; `000000000000000F` - дължината на елемента е 15 символа; Няма подравняващи символи, защото нивото на вложеност е нула; `Интернет адреси` - самият елемент.

ZZZ2TXTFromDividerPath

Записва множество от указаната база и път с разделители към текст

параметри:

```
{BASE} - името на базата, в която е множеството
{PATH} - път в базата с разделители
{DIV} - разделител използван в пътя
```

реализация:

```
#[=;ZZZ2TXTFromDividerPath;[
                #[ZZZ2TXTFromPath;{BASE};##[~~///...;[{PATH}];[{DIV}]]]
        ]
]
#[=.;ZZZ2TXTFromDividerPath;{BASE};{PATH};{DIV}]
```

използване:

```
##[cout;
        #[ZZZ2TXTFromDividerPath;docbases/zzzbase;;/]
]
```

Формата на един ред от текстовия резултат е следния:

[ниво на вложеност (64 битово шестнадесетично число)][,][дължина на елемента(64 битово шестнадесетично число)][,][подравняващи символи с брой равен на нивото на вложеност][елемент]

Пример: `0000000000000000,000000000000000F,Интернет адреси`

`0000000000000000` - главно ниво; `000000000000000F` - дължината на елемента е 15 символа; Няма подравняващи символи, защото нивото на вложеност е нула; `Интернет адреси` - самият елемент.

6.2.2 Четене на множество от текст (UTF-8)

TXT2ZZZ

Прочита текущото множество в указаната база от текст и връща `true` при успех

параметри:

{BASE} - името на базата, в която е множеството
{PID} - идентификатор на процеса за четене или запис
{TXT} - текста, който трябва да преобразуваме в множество
{RESULT} - името на променливата, в която да запишем резултата от преобразуването

реализация:

```
#[=;skipSpace;[
                #[>=; ;##[^.;{FORM};<error>];[
                                #[skipSpace;{FORM}]
                        ];[
                                #[*v;{FORM};#[-;#[*^;{FORM}];1]]
                        ]
                ]
        ]
]
#[=.;skipSpace;{FORM}]

#[=;moveToRelativeLevel;[
                #[==;0;{LEVEL};;[
                                #[>=n;{LEVEL};1;[
                                                #[==;true;#[~~>v;{BASE};{PID}];;]
                                                #[moveToRelativeLevel;{BASE};{PID};#[fmtd
                                        ];[
                                                #[==;true;#[~~>^;{BASE};{PID}];;]
                                                #[moveToRelativeLevel;{BASE};{PID};#[fmtd
                                        ]
                                ]
                        ]
                ]
        ]
]
#[=.;moveToRelativeLevel;{BASE};{PID};{LEVEL}]

#[#;[
                НИВО                    ,ДЪЛЖИНА                ,[брой подравняващи
```

```
                0000000000000000,000000000000000F,Интернет адреси
#]]

#[=;TXT2ZZZRecursion;[
                #[skipSpace;TXT2ZZZ_TXT]
                #[=;LEVEL;#[refmtl;%I64X;%I64d;#[^..;TXT2ZZZ_TXT;16;<error>]]]
                #[*v;TXT2ZZZ_TXT;#[+;1;#[*^;TXT2ZZZ_TXT]]]
                #[=;LENGTH;#[refmtl;%I64X;%I64d;#[^..;TXT2ZZZ_TXT;16;<error>]]]
                #[*v;TXT2ZZZ_TXT;#[+;1;#[*^;TXT2ZZZ_TXT]]]
                #[*v;TXT2ZZZ_TXT;#[+;##[LEVEL];#[*^;TXT2ZZZ_TXT]]]
                #[=;ELEMENT;##[^..;TXT2ZZZ_TXT;##[LENGTH];]]
                #[==;;##[ELEMENT];;[
                                #[moveToRelativeLevel;{BASE};{PID};#[fmtd;[%0.0f];#[-;##[
]]
                                #[==;true;#[~~+.;{BASE};{PID};##[ELEMENT]];[
                                                #[=;{RESULT};true]
                                        ];
                                ]
                                #[TXT2ZZZRecursion;{BASE};{PID};##[LEVEL];{RESULT}]
                        ]
                ]
        ]
]
#[=.;TXT2ZZZRecursion;{BASE};{PID};{LEVEL};{RESULT}]

#[=;TXT2ZZZ;[
                #[=;TXT2ZZZ_TXT;[{TXT}]]
                #[==;true;##[~~>v;{BASE};{PID}];[

                                #[TXT2ZZZRecursion;{BASE};{PID};0;{RESULT}]

                                #[==;true;##[~~>^;{BASE};{PID}];;]
                        ];
                ]
                ##[{RESULT}]
                #[=x;TXT2ZZZ_TXT]
        ]
]
```

```
#[=.;TXT2ZZZ;{BASE};{PID};{TXT};{RESULT}]
```

използване:

```
##[cout;
        #[.=;_base_;docbases/zzzbase]
        #[.=;TXT2ZZZ_RESULT;false]

        #[~~+;##[_base_]]
        #[.=;_v_;#[~~v;##[_base_];10000]]

        #[==;true;#[~~›^;##[_base_];##[_v_]];;]
        #[TXT2ZZZ;##[_base_];##[_v_];[
                0000000000000000,000000000000000F,Интернет адреси
                0000000000000000,0000000000000011,Интернет адреси 1
];TXT2ZZZ_RESULT]

        #[~~.;##[_base_];##[_v_]]
        #[~~-;##[_base_]]
]
```

Формата на един ред от текстовия резултат е следния:

[ниво на вложеност (64 битово шестнадесетично число)][,][дължина на елемента(64 битово шестнадесетично число)][,][подравняващи символи с брой равен на нивото на вложеност][елемент]

Пример: `0000000000000000,000000000000000F,Интернет адреси`

`0000000000000000` - главно ниво; `000000000000000F` - дължината на елемента е 15 символа; Няма подравняващи символи, защото нивото на вложеност е нула; `Интернет адреси` - самият елемент.

TXT2ZZZToPath

Прочита множество в указаната база и път от текст

параметри:

`{BASE}` - името на базата, в която е множеството

{PATH} - път в базата
{TXT} - текста, който трябва да преобразуваме в множество

реализация:

```
#[=;TXT2ZZZToPath;[
                #[=;TXT2ZZZ_RESULT;false]

                #[~~+;{BASE}]

                #[=;pIdW_TXT2ZZZToPath;##[~~v;{BASE};100000]]

                #[==;true;##[~~+...;{BASE};##[pIdW_TXT2ZZZToPath];{PATH};true];[
                                #[TXT2ZZZ;{BASE};##[pIdW_TXT2ZZZToPath];[{TXT}];TXT2ZZZ_R
                        ];
                ]

                #[==;true;##[TXT2ZZZ_RESULT];[
                                ##[~~.;{BASE};##[pIdW_TXT2ZZZToPath]]
                        ];[
                                ##[~~vx;{BASE}]
                        ]
                ]
                #[~~-;{BASE}]

                #[=x;TXT2ZZZ_RESULT]
        ]
]
#[=.;TXT2ZZZToPath;{BASE};{PATH};{TXT}]
```

използване:

```
##[cout;
        #[TXT2ZZZToPath;docbases/zzzbase;;[
                0000000000000000,000000000000000F,Интернет адреси
                0000000000000000,0000000000000011,Интернет адреси 1
]]
]
```

Формата на един ред от текстовия резултат е следния:

[ниво на вложеност (64 битово шестнадесетично число)][,][дължина на елемента(64 битово шестнадесетично число)][,][подравняващи символи с брой равен на нивото на вложеност][елемент]

Пример: `0000000000000000,000000000000000F,Интернет адреси`

`0000000000000000` - главно ниво; `000000000000000F` - дължината на елемента е 15 символа; Няма подравняващи символи, защото нивото на вложеност е нула; `Интернет адреси` - самият елемент.

TXT2ZZZToDividerPath

Прочита множество в указаната база и път с разделители от текст

параметри:

`{BASE}` - името на базата, в която е множеството
`{PATH}` - път в базата с разделители
`{DIV}` - разделител използван в пътя
`{TXT}` - текста, който трябва да преобразуваме в множество

реализация:

```
#[=;TXT2ZZZToDividerPath;[
                #[TXT2ZZZToPath;{BASE};##[~~///...;[{PATH}];[{DIV}]];[{TXT}]]
        ]
]
#[=.;TXT2ZZZToDividerPath;{BASE};{PATH};{DIV};{TXT}]
```

използване:

```
##[cout;
        #[TXT2ZZZToDividerPath;docbases/zzzbase;;/;[
                0000000000000000,000000000000000F,Интернет адреси
                0000000000000000,0000000000000011,Интернет адреси 1
]]
]
```

Формата на един ред от текстовия резултат е следния:

[ниво на вложеност (64 битово шестнадесетично число)][,][дължина на елемента(64 битово шестнадесетично число)][,][подравняващи символи с брой равен на нивото на вложеност][елемент]

Пример: `0000000000000000,000000000000000F,Интернет адреси`

`0000000000000000` - главно ниво; `000000000000000F` - дължината на елемента е 15 символа; Няма подравняващи символи, защото нивото на вложеност е нула; `Интернет адреси` - самият елемент.

7. Глава V. Примерни програми за ползване на ЗЗЗ сървър от различни програмни езици

Навсякъде в примерите се използва име на хост “localhost” и номер на порт “3333”. Това са стойностите по подразбиране, но те могат да са различни в зависимост от вашата инсталация.

7.1 Примерна програма на C

zzzclient.h:

```
/*
 * zzzclient.h
 *
 * Copyright (C) 2016 ZZZ Ltd. - Bulgaria. All rights reserved.
 */

#ifndef __ZZZCLIENT_H__
#define __ZZZCLIENT_H__

#include <conio.h>        // getch
#ifdef _WIN32
#include <windows.h>
#include <io.h>
#define socklen_t int
#else
#include <stdlib.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/time.h>
```

```
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/select.h>
#include <netdb.h>
#include <stdint.h>
#include <fcntl.h>

#define closesocket close
#if !defined(O_BINARY)
#define O_BINARY 0
#endif
#endif

int zzzclient_init_socket();
void zzzclient_uninit_socket();
void zzzclient_init(int* sock);
void zzzclient_destroy(int* sock);
BOOL zzzclient_connect(int* sock, char* host, int port);
void zzzclient_close(int* sock);
BOOL zzzclient_send(int* sock, char* data);
char* zzzclient_receive(int* sock, char* result);
char* zzzclient_zzzprogram(char* host, int port, char* data, char* result);

#endif // __ZZZCLIENT_H__
```

zzzclient.c:

```
/*
 * zzzclient.c
 *
 * Copyright (C) 2016 ZZZ Ltd. - Bulgaria. All rights reserved.
 */

#include "zzzclient.h"


int zzzclient_init_socket()
```

```
{
#ifdef WIN32
        WSADATA wsaData;
        if (
                WSAStartup(MAKEWORD(2, 2), &wsaData) &&
                WSAStartup(MAKEWORD(2, 1), &wsaData) &&
                WSAStartup(MAKEWORD(1, 1), &wsaData)
                )
        {
                return 0;
        }
#endif
        return 1;
}

void zzzclient_uninit_socket()
{
#ifdef WIN32
        WSACleanup();
#endif
}

void zzzclient_init(int* sock)
{
    *sock = -1;

        zzzclient_init_socket();
}

void zzzclient_destroy(int* sock)
{
        zzzclient_close(sock);

        zzzclient_uninit_socket();
}

/**
    Connect to a host on a certain port number
```

```
*/
BOOL zzzclient_connect(int* sock, char* address , int port)
{
    struct sockaddr_in server;

    // create socket if it is not already created
    if(*sock == -1)
    {
        //Create socket
        *sock = (int)socket(AF_INET , SOCK_STREAM , 0);
        if (*sock == -1)
        {
            // Could not create socket
        }

        // Socket created
    }
    else    {   /* OK , nothing */  }

    // setup address structure
    if((int)inet_addr(address) == -1)
    {
        struct hostent *he;
        struct in_addr **addr_list;

        // resolve the hostname, its not an ip address
        if((he = gethostbyname(address)) == NULL)
        {
            // Failed to resolve hostname

            return FALSE;
        }

        // Cast the h_addr_list to in_addr , since h_addr_list also has the \
ip address in long format only
        addr_list = (struct in_addr **) he->h_addr_list;

        server.sin_addr = *addr_list[0];
```

```
    }
    // plain ip address
    else
    {
        server.sin_addr.s_addr = inet_addr(address);
    }

    server.sin_family = AF_INET;
    server.sin_port = htons( port );

    // Connect to remote server
    if (connect(*sock , (struct sockaddr *)&server , sizeof(server)) < 0)
    {
        // Error: Connect failed.
        return 1;
    }

#ifndef _WIN32
    {
                int sockFlags;
                sockFlags = fcntl(sock, F_GETFL, 0);
                sockFlags |= O_NONBLOCK;
                fcntl(sock, F_SETFL, sockFlags);
    }
#endif

    // Connected
    return TRUE;
}

/**
    Send data to the connected host
*/
BOOL zzzclient_send(int* sock, char* data)
{
    // Send some data
    if(send(*sock, data, (int)strlen(data)+1, 0) < 0)
    {
```

```
        // Send failed
        return FALSE;
    }
    // Data "data" send;

    return TRUE;
}

/**
    Receive data from the connected host
*/
char* zzzclient_receive(int* sock, char* result)
{
        char buffer[512];
        int len = 0;
        int all=0;
    result[0] = '\0';

        if(result == NULL)
                return result;

    // Receive a reply from the server
    while((len = (int)recv(*sock , buffer , sizeof(buffer) , 0)) > 0)
    {
                buffer[len] = '\0';
                memcpy(&result[all], buffer, len);
            all += len;
    }

    return result;
}

void zzzclient_close(int* sock)
{
        if(sock >= 0)
        {
                closesocket(*sock);
                *sock = -1;
```

```
        }
}

char* zzzclient_zzzprogram(char* host, int port, char* data, char* result)
{
        int sock;

        if(result == NULL)
                return result;

        zzzclient_init(&sock);

        if(zzzclient_connect(&sock, host, port))
        {
                zzzclient_send(&sock, data);
                result = zzzclient_receive(&sock, result);
                zzzclient_close(&sock);
        }

        zzzclient_destroy(&sock);

        return result;
}
```

main.c:

```
// main.c : Defines the entry point for the console application.
//

#include "zzzclient.h"
#include <stdio.h>


int main(int argc, char* argv[])
{
        char ch;
        char result[1024] = "";
```

```
        zzzclient_zzzprogram("localhost", 3333, "#[cout;Hello World from ZZZServer!\
]", result);

        printf("%s\n", result);

        printf("%s", "Press any key...");


#ifdef _WIN32
        ch = _getch();
#else
        ch = getch();
#endif

        return 0;
}
```

7.2 Примерна програма на C++

ZZZClient.h:

```
/*
 * ZZZClient.h
 *
 * Copyright (C) 2016 ZZZ Ltd. - Bulgaria. All rights reserved.
 */

#ifndef __ZZZCLIENT_H__
#define __ZZZCLIENT_H__

#include <iostream>        // cout
#include <string>         // string
#include <conio.h>         // getch
#ifdef _WIN32
#include <windows.h>
#include <io.h>
#define socklen_t int
#else
#include <stdlib.h>
#include <unistd.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <sys/select.h>
#include <netdb.h>
#include <stdint.h>
#include <fcntl.h>

#define closesocket close
#if !defined(O_BINARY)
#define O_BINARY 0
```

```
#endif
#endif

using namespace std;

class ZZZClient
{
private:
    int sock;
    struct sockaddr_in server;
        int InitSocket();
        void UninitSocket();

public:
    ZZZClient();
        virtual ~ZZZClient();
    BOOL Connect(string, int);
        void Close();
    BOOL Send(string data);
    string Receive();
        string ZZZProgram(string host, int port, string data);
};

#endif // __ZZZCLIENT_H__
```

ZZZClient.cpp:

```
/*
 * ZZZClient.cpp
 *
 * Copyright (C) 2016 ZZZ Ltd. - Bulgaria. All rights reserved.
 */

#include "ZZZClient.h"


int ZZZClient::InitSocket()
{
```

```
#ifdef WIN32
        WSADATA wsaData;
        if (
                WSAStartup(MAKEWORD(2, 2), &wsaData) &&
                WSAStartup(MAKEWORD(2, 1), &wsaData) &&
                WSAStartup(MAKEWORD(1, 1), &wsaData)
                )
        {
                return 0;
        }
#endif
        return 1;
}

void ZZZClient::UninitSocket()
{
#ifdef WIN32
        WSACleanup();
#endif
}

ZZZClient::ZZZClient()
{
    this->sock = -1;

        InitSocket();
}

ZZZClient::~ZZZClient()
{
        Close();

        UninitSocket();
}

/**
    Connect to a host on a certain port number
*/
```

```
BOOL ZZZClient::Connect(string address , int port)
{
    // create socket if it is not already created
    if(sock == -1)
    {
        //Create socket
        sock = (int)socket(AF_INET , SOCK_STREAM , 0);
        if (sock == -1)
        {
            // Could not create socket
        }

        // Socket created
    }
    else    {   /* OK , nothing */  }

    // setup address structure
    if((int)inet_addr(address.c_str()) == -1)
    {
        struct hostent *he;
        struct in_addr **addr_list;

        // resolve the hostname, its not an ip address
        if ( (he = gethostbyname( address.c_str() ) ) == NULL)
        {
            // Failed to resolve hostname

            return FALSE;
        }

        // Cast the h_addr_list to in_addr , since h_addr_list also has the \
ip address in long format only
        addr_list = (struct in_addr **) he->h_addr_list;

        server.sin_addr = *addr_list[0];
    }
    // plain ip address
    else
```

```
    {
        server.sin_addr.s_addr = inet_addr( address.c_str() );
    }

    server.sin_family = AF_INET;
    server.sin_port = htons( port );

    // Connect to remote server
    if (connect(sock , (struct sockaddr *)&server , sizeof(server)) < 0)
    {
        // Error: Connect failed.
        return 1;
    }

#ifndef _WIN32
    {
                int sockFlags;
                sockFlags = fcntl(sock, F_GETFL, 0);
                sockFlags |= O_NONBLOCK;
                fcntl(sock, F_SETFL, sockFlags);
    }
#endif

    // Connected
    return TRUE;
}

/**
    Send data to the connected host
*/
BOOL ZZZClient::Send(string data)
{
    // Send some data
    if( send(sock , data.c_str() , (int)strlen(data.c_str())+1 , 0) < 0)
    {
        // Send failed
        return FALSE;
    }
```

```
    // Data "data" send;

    return TRUE;
}

/**
    Receive data from the connected host
*/
string ZZZClient::Receive()
{
        char buffer[512];
    string reply = "";

    // Receive a reply from the server
        int len = 0;
    while((len = (int)recv(sock , buffer , sizeof(buffer) , 0)) > 0)
    {
                buffer[len] = '\0';
            reply += buffer;
    }

    return reply;
}

void ZZZClient::Close()
{
        if(this->sock >= 0)
        {
                closesocket(this->sock);
                this->sock = -1;
        }
}

string ZZZClient::ZZZProgram(string host, int port, string data)
{
        string result;

        if(Connect(host, port))
```

```
        {
                Send(data);
                result = Receive();
                Close();
        }
        return result;
}
```

main.cpp:

```
/*
 * main.cpp
 *
 * Copyright (C) 2016 ZZZ Ltd. - Bulgaria. All rights reserved.
 */

#include "ZZZClient.h"

int main(int argc, char* argv[])
{
        ZZZClient client;

        cout << client.ZZZProgram("localhost", 3333, "#[cout;Hello World from ZZZSe\
rver!]") << endl;

        cout << "Press any key...";

#ifdef _WIN32
        char ch = _getch();
#else
        char ch = getch();
#endif

        return 0;
}
```

7.3 Примерна програма на C#

Hello world from ZZZServer:

```
/*
 * Copyright (C) 2013-2016 ZZZ Ltd. - Bulgaria. All rights reserved.
 */

using System;
using System.Net.Sockets;
using System.Text;

namespace ZZZClientCS
{
    class Program
    {
        static string ZZZProgram(string serverHost, int serverPort, string p\
rogram)
        {
            try
            {
                if (serverHost.Equals("localhost"))
                    serverHost = "127.0.0.1";
                TcpClient tc = new TcpClient(serverHost, serverPort);
                NetworkStream ns = tc.GetStream();

                if (ns.CanWrite && ns.CanRead)
                {
                    // Do a simple write.
                    byte[] sendBytes = Encoding.UTF8.GetBytes(program + '\0'\
);
                    ns.Write(sendBytes, 0, sendBytes.Length);

                    // Read the NetworkStream into a byte buffer.
                    byte[] bytes = new byte[tc.ReceiveBufferSize];

                    StringBuilder returndata = new StringBuilder();
                    int receivedBytes = ns.Read(bytes, 0, tc.ReceiveBufferSi\
```

```
ze);
                    returndata.Append(Encoding.UTF8.GetString(bytes, 0, rece\
ivedBytes));
                    while (receivedBytes > 0)
                    {
                        receivedBytes = ns.Read(bytes, 0, tc.ReceiveBufferSi\
ze);
                        returndata.Append(Encoding.UTF8.GetString(bytes, 0, \
receivedBytes));
                    }

                    // Return the data received from the host.
                    return returndata.ToString();
                }
                else
                {
                    if (!ns.CanRead)
                    {
                        Console.WriteLine("cannot not read data from this st\
ream");
                        tc.Close();
                    }
                    else
                    {
                        if (!ns.CanWrite)
                        {
                            Console.WriteLine("cannot write data to this str\
eam");
                            tc.Close();
                        }
                    }
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e);
            }
            return String.Empty;
```

```
        }

        static void Main(string[] args)
        {
            Console.OutputEncoding = Encoding.UTF8;

            DateTime startTime = DateTime.UtcNow;

            Console.WriteLine(ZZZProgram("localhost", 3333, "#[cout;Hello Wo\
rld from ZZZServer!]"));

            DateTime stopTime = DateTime.UtcNow;
            long elapsedTime = stopTime.Millisecond - startTime.Millisecond;
            Console.WriteLine(elapsedTime.ToString() + " milliseconds");

            Console.ReadKey();
        }
    }
}
```

7.4 Примерна програма на Java

Hello world from ZZZServer:

```java
/*
 * Copyright (C) 2013-2016 ZZZ Ltd. - Bulgaria. All rights reserved.
 */
package zzzclientj;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;

/**
 *
 * @author ZZZ Ltd. - Bulgaria
 */
public class MainApp {

    public static String ZZZProgram(String serverHost, int serverPort,
            String program)
    {
        String result = "";

        try {
                if(serverHost.equals("localhost"))
                        serverHost = "127.0.0.1";
            Socket socket = new Socket(serverHost, serverPort);
            PrintWriter out = new PrintWriter(new OutputStreamWriter(
                    socket.getOutputStream(), "UTF-8"), true);
            BufferedReader in = new BufferedReader(
                    new InputStreamReader(socket.getInputStream(), "UTF-8"));

            out.print(program + '\0');
            out.flush();
```

```
            StringBuilder sb = new StringBuilder();
            int DEFAULT_BUFFER_SIZE = 1000;
            char buf[] = new char[DEFAULT_BUFFER_SIZE];
            int n;
            while ((n = in.read(buf)) > 0) {
                sb.append(buf, 0, n);

                if(!in.ready())
                    break;
            }
            result = sb.toString();

            out.close();
            in.close();
            socket.close();
        } catch(Exception e) {
            e.printStackTrace();
        }

        return result;
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
            long startTime = System.currentTimeMillis();

        System.out.println(ZZZProgram("localhost", 3333, "#[cout;Hello world\
 from ZZZServer!]"));

        long stopTime = System.currentTimeMillis();
        long elapsedTime = stopTime - startTime;
        System.out.println(elapsedTime + " milliseconds");
        }
}
```

7.5 Примерна програма на PHP

Hello world from ZZZServer:

```
<?php
//
// Copyright (C) 2013-2016 ZZZ Ltd. - Bulgaria. All rights reserved.
//
        class Socket
        {
                var $m_host  = "";
                var $m_port  = 0;
                var $m_sock;
                var $buf = "";

                function Socket($host, $port)
                {
                        $this->m_host = $host;
                        $this->m_port = $port;
                }

                function Connect()
                {
                        $this->m_sock = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
                        return socket_connect($this->m_sock, gethostbyname($this->m_host)
>m_port);
                }

                function send($data)
                {
                        if(socket_write($this->m_sock, $data) == false)
                        {
                                printf("** failed to send(%s) **\n");
                                return;
                        }
                }

                function read($length = 2048)
```

```
        {
                return socket_read($this->m_sock, $length);
        }

        function kill()
        {
                socket_close($this->m_sock);
        }

        function print($s)
        {
                $this->send($s);
        }

        function read()
        {
                $response = '';
                while($resp = socket_read($this->m_sock, 1024)) {
                        if(!$resp)
                                break;
                        $response .= $resp;
                }
                return $response;
        }
}

$host = "localhost";
$port = 3333;
function ZZZProgram($program) {
        global $host;
        global $port;
        $result="";
        $Connection = new Socket($host, $port);
        if($Connection->Connect()){
                $Connection->print($program + '\0');
                $result = $Connection->read();

                $Connection->kill();
```

```
                }
                return $result;
        }
        echo ZZZProgram("#[cout;[Hello world from ZZZServer!]]");
?>
```

7.6 Примерна програма на Python

Hello world from ZZZServer:

```
#
# Copyright (C) 2016 ZZZ Ltd. - Bulgaria. All rights reserved.
#
import socket, sys

def zzzprogram(host, port, program) :
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.settimeout(2)

        # connect to remote host
        try :
                s.connect((host, port))
        except :
                return

        s.send((program + '\0').encode('utf-8'))

        result = ''
        while True :
                data = s.recv(4096)
                if not data :
                        break
                result += data.decode('utf-8')

        s.close()
        return result

sys.stdout.write(
        zzzprogram('localhost', 3333,
                '#[cout;Hello World from ZZZServer!]'))
```

7.7 Примерна програма на `VisualBasic`

Hello world from ZZZServer:

```
'
' Copyright (C) 2013-2016 ZZZ Ltd. - Bulgaria. All rights reserved.
'

Imports System.Net.Sockets
Imports System.Text

Module ZZZClientVB

    Function ZZZProgram(ByVal serverHost As String, ByVal serverPort As Stri\
ng, ByVal program As String) As String
        Try
            If serverHost = "localhost" Then
                serverHost = "127.0.0.1"
            End If
            Dim tc As New TcpClient(serverHost, serverPort)
            Dim ns As NetworkStream = tc.GetStream()

            If ns.CanWrite And ns.CanRead Then
                ' Do a simple write.
                Dim sendBytes As [Byte]() = Encoding.UTF8.GetBytes(program &\
 Chr(0))
                ns.Write(sendBytes, 0, sendBytes.Length)

                ' Read the NetworkStream into a byte buffer.
                Dim bytes(tc.ReceiveBufferSize) As Byte
                Dim returndata As StringBuilder = New StringBuilder()
                Dim receivedBytes As Integer = ns.Read(bytes, 0, tc.ReceiveB\
ufferSize)
                returndata.Append(Encoding.UTF8.GetString(bytes, 0, received\
Bytes))
                While receivedBytes > 0
                    receivedBytes = ns.Read(bytes, 0, tc.ReceiveBufferSize)
                    returndata.Append(Encoding.UTF8.GetString(bytes, 0, rece\
```

```
ivedBytes))
                End While

                ' Return the data received from the host.
                Return returndata.ToString()
            Else
                If Not ns.CanRead Then
                    Console.WriteLine("cannot not read data from this stream\
")
                    tc.Close()
                Else
                    If Not ns.CanWrite Then
                        Console.WriteLine("cannot write data to this stream")
                        tc.Close()
                    End If
                End If
            End If
        Catch e As Exception
            Console.WriteLine(e)
        End Try

        Return ""
    End Function

    Sub Main()
        Console.OutputEncoding = Encoding.UTF8

        Dim startTime As DateTime = DateTime.UtcNow

        Console.WriteLine(ZZZProgram("localhost", 3333, "#[cout;Hello World \
from ZZZServer!]"))

        Dim stopTime As DateTime = DateTime.UtcNow
        Dim elapsedTime As Long = stopTime.Millisecond - startTime.Milliseco\
nd
        Console.WriteLine(elapsedTime.ToString + " milliseconds")

        Console.ReadKey()
```

```
    End Sub

End Module
```