



Програм хангамжийн бүтээлт F.CS311

5 зарчим - principle

5 дүрмээс 5 зарчим гарна.

- Linguistic modular units - Модулын хэлзүйн нэгж
- Self-documentation - Баримжуулалтын мөн чанар
- Uniform access - Нэг хэв загварын хандалт
- Open-Close - Нээлттэй ба хаалттай байх зарчим
- Single choice - Цор ганц сонголт



Модулын хэлзүйн нэгж

- Модулын хэлзүйн нэгжийн зарчим нь системийн тодорхойлолт, загварчлал, хэрэгжүүлэх зэрэг програм хангамжын янз бүрийн төвшинг тайлбарлахад хэрэглэгддэг дүрслэлүүд модулын хэлбэрт тусгагдах ёстой.



Модулын хэлзүйн нэгж

Тодорхойлолт

Модулиуд нь системийг илэрхийлэх хэллэгт ашиглагддаг өгүүлбэр зүйн нэгжүүдийг ашигласан байх ёстой.



Баримжуулалтын мөн чанар

Тодорхойлолт

Модулыг бичигч нь модулын хэсэг бүрийн тухай
бүх мэдээллийг өөрөө бий болгохыг хичээнэ.



Нэг хэв загварын хандалт

Энэ нь зөвхөн тэмдэглэгээний асуудалыг зохицуулна.
Нэг хэв загварын хандалтын зарчим бол объект хандалтат зохиомж ба тэмдэглэгээний тухай олон асуудлуудтай холбоотой зохиомжын дүрэмтэй холбоотой.

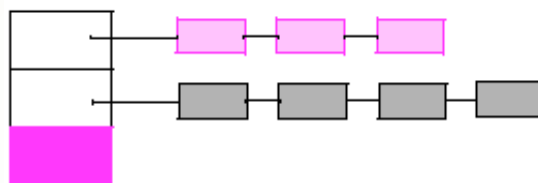
Нэг хэв загварын хандалт

- Тодорхой мэдээлэл рүү ханддаг нэрийг X , (цаашид объект) X -ийн хэрэгцээт төлөвийг F (цаашид функц) гэе. X нь банкны дансыг илэрхийлдэг, F нь дансны төлвийг илэрхийлдэг. X -ийн F төлөвийн үр дүнг F –ийг хэрхэн хэрэглэж байгаагаас үл хамааран тэмдэглэх илэрхийлэхийг Нэг хэв загварын хандалт харуулна.

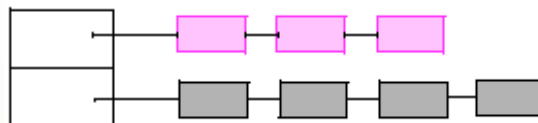
Нэг хэв загварын хандалт

Банкны дансны хувьд дараах 2 аргаар аргаар шийдэж болно.

A1 Орлогын жагсаалт
Зарлагын жагсаалт
Үлдэгдэл



A2 Орлогын жагсаалт
Зарлагын жагсаалт



- A1 нь үлдэгдэл нь бичлэгийн нэг багана болж хадгалагдана. Энэ тохиолдолд гүйлгээ бүрийн дараа үлдэгдэл бодогдож өөрчлөгдөнө.

- A2 нь үйлдэгдэл багана хадгалахгүйгээр шаардлагатай тохиолдолд тооцоолон гаргах боломжтой.

Нэг хэв загварын хандалт

- Паскаль, Ada, C, C++ ба Java хэлэнд ерөнхий тэмдэглэгээ нь A1 тохиолдолд X.F, харин A2 тохиолдолд F(X) байна. A1 нь тооцоолол хэмнэнэ, харин A2 нь зай хэмнэнэ. Програм хангамжыг хөгжүүлж байх үед тухайн үеийн техник болон програм хангамжын шаардлагын үүднээс 2 тохиолдол хооронд шилжих шаардлага байнга гардаг.

Нэг хэв загварын хандалт

- Иймээс тасралтгүй байх шаардлагаас үүдэн хандах төлвийн тэмдэглэгээ нь 2 тохиолдолд ижил байх ёстой ба хэрэгжүүлэх явцад тохиолдол хооронд шилжих тохиолдолд F хэрэглэж байгаа модуль дотор өөрчлөлт хийх шаардлагагүй болно. Үүнийг нэг хэв загварын хандалт гэнэ.



Нэг хэв загварын хандалт

Нэг хэв загварын хандалтын зарчим:

Модул дотор тооцоолол хэмнэсэн эсвэл зай
хэмнэсэн аль аргыг хэрэглэсэн байсан объект
болон түүний төлөвт зөвхөн нэг л
тэмдэглэгээг хэрэглэнэ.




Нээлттэй ба хаалттай байх зарчим

Модулыг задлан шинжлэх гэдгээс нээлттэй ба хаалттай байх зарчим гарна. Өөрөөр хэлбэл модулыг задлах гэдэгтэй холбоотой.

Тодорхойлолт :


Модуль нь нээлттэй эсвэл хаалттай төлөвт байж болно.

Модул нээлттэй гэдэг нь түүнийг өргөтгөх боломжтой буюу түүний өгөгдлийн бүтцэд талбар нэмэх боломжтой гэсэн үг.




Нээлттэй ба хаалттай байх зарчим

- Модул хаалттай гэдэг нь бусад модулиуд түүнийг ашиглах боломжтой гэсэн үг. Энэ нь модуль тодорхойлогдож дууссан, тогтвортойг илэрхийлнэ. Мөн модуль хөрвүүлэгдэж ажиллах модулын санд хадгалагдсан байх ба бусад програмууд (клиент) дуудах боломжтой болсон байна.



Нээлттэй ба хаалттай байх зарчим

- Модулийг загварчлах тодорхойлох шатанд модулийг хаана гэдэг нь менежментийн хувьд зөвшөөрөл авах, модулыг төслийн хэсэг болгож нэмж, бусад модул зохиогчдийг танилцуулахын тулд тэдгээрийн интерфейсыг хэвлэж өгөх үйл ажиллагаа юм.



Нээлттэй ба хаалттай байх зарчим

- Модул хаалттай байх шаардлага, үлдсэн модул нээлттэй байх ялгаатай тохиолдлоос гарч ирнэ. Модулыг нээлттэй байлгасанаар програм хангамжыг кодчлох үйл явцыг хийнэ. Гэвч төслийн удирдагчын хүсэлтээр, мөн систем нь бие биеэсээ хамаарсан хэд хэдэн модулиас тогтдог тул модулыг дуусгалгүйгээр хаах шаардлага гардаг.

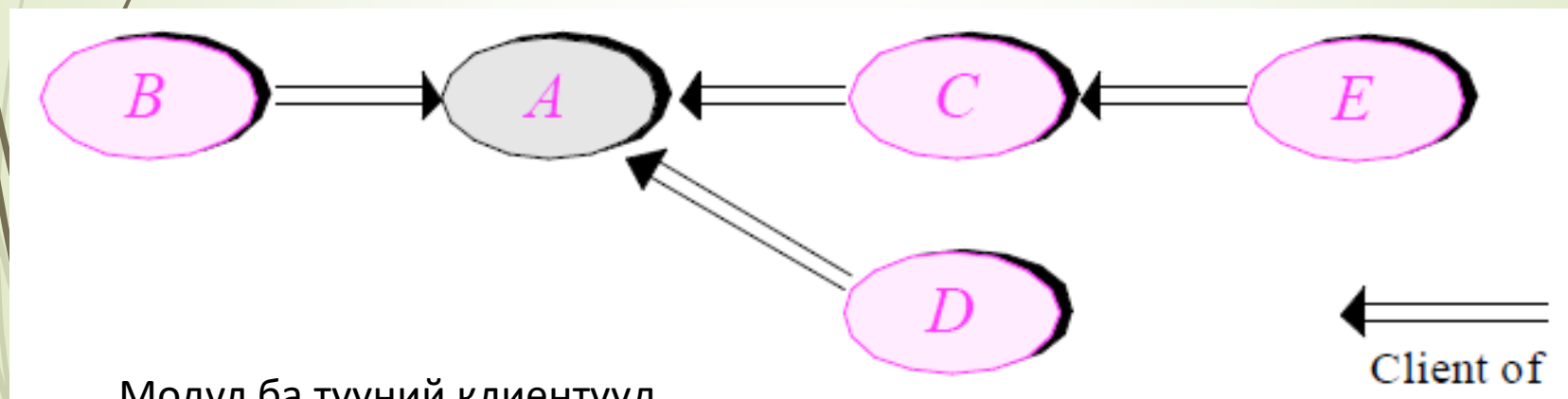


Нээлттэй ба хаалттай байх зарчим

- Жишээ нь : хэрэглэгчийн интерфейс модул нь команд шинжлэгч модул ба график модулаас хамаарна. Команд шинжлэгч модул нь үг зүйн шинжлэгээний модулаас хамаарна гэх мэт. Хэрэв модулийн бүх үйлдлүүдийг хийж дуустал модулыг хаахгүй байвал олон модултай програмчлал хэзээ ч дуусахгүй, модулийг хөгжүүлэгч бүр бусдынхаа дуусахыг хүлээсээр байна.

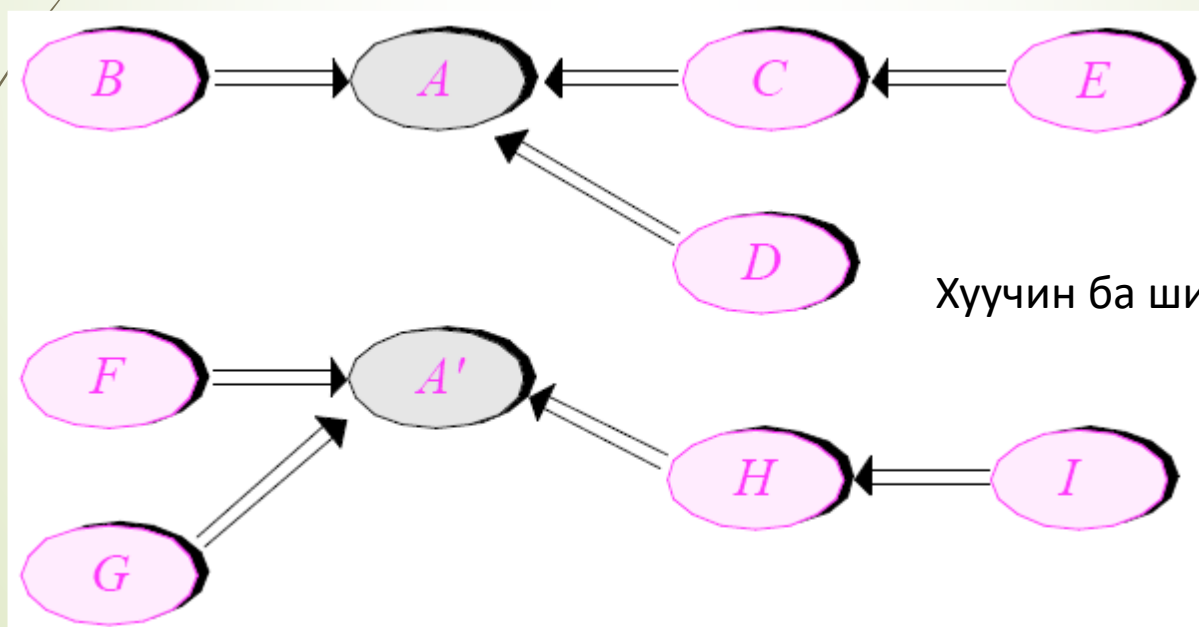
Нээлттэй ба хаалттай байх зарчим


- Доорх зурагт нээлттэй хаалттай байдлыг зохицуулахад төвөгтэй тохиолдлыг үзүүлэв. Эхний зурагт модул А –ийн клиент бөгөөд өөрийн гэсэн клиенттэй В,С,D модуль байна.



Нээлттэй ба хаалттай байх зарчим

- Дараагийн зурагт дээрх тохиолдлыг хүндрүүлэх байдлаар шинэ клиентүүд бий болж, А модулийг өргөтгөж А' модулыг үүсгэсэн.






Нээлттэй ба хаалттай байх зарчим

Энэ тохиолдолд дараах 2 шийдэл байж болно. Үүнд :

Шийдэл 1: шинэ клиентүүдийн шаардсан, A' модулийн үйлдлүүдээр өргөтгөгдсөн шинэ A модул үүсгэх

Шийдэл 2: A модулийг тэр чигээр нь үлдээгээд, A модулийг хуулбарлан нэрийг нь A' болгоод бүх шаардлагатай үйлдлүүдийг нэмэх. Ингэсний дараа A' модул ба A модул хооронд холболт байхгүй болно.




Нээлттэй ба хаалттай байх зарчим

Шийдэл1-ийн үед:

А модул үүсгэхэд хугацаа шаардлагатай, бас илүү олон клиенттэй болно.

Шинэ болон хүүчин хэрэглэгчийн шаардлагууд зөрчилдөж болно. Энэ нь төслийн удирдагчийн хувьд том хүндрэл болох ба програм хамгамжын хөгжүүлэх процесс болох тестчлэх, шалгах (debug), бичиг баримт бүрдүүлэх процесс дахиж хийгдэнэ.




Нээлттэй ба хаалттай байх зарчим

Шийдэл2-ийн үед:

Өмнөх шийдэл дээр гарах хүндрэл энд гарахгүй

Гэвч үйлдлүүдийн давхардал байна.

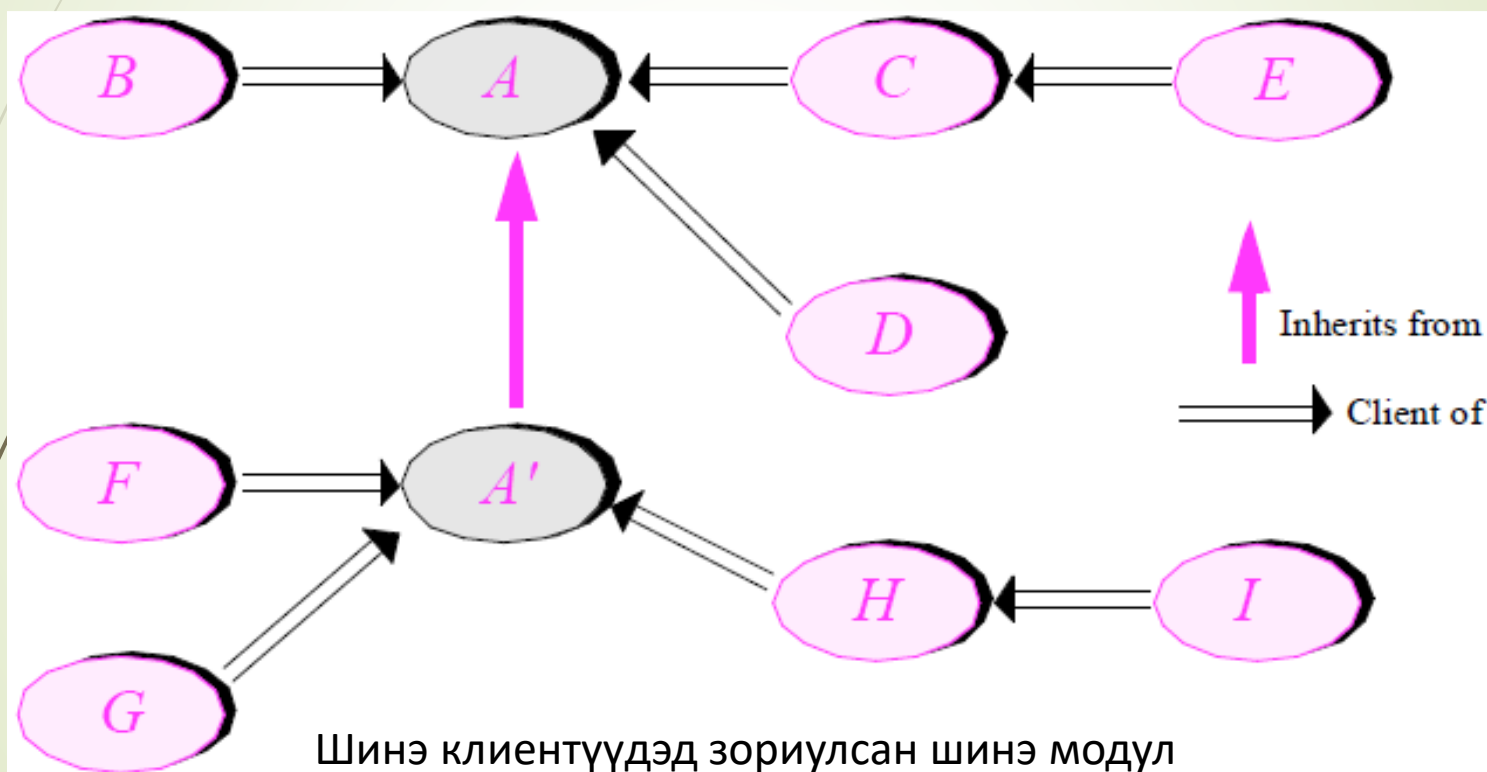
Мөн хэрэв А модульд өөрчлөлт гарах
шаардлагатай бол А болон А' модульд засвар
хийх шаардлагатай. Мөн эдгээр нь олон
клиентэд нөлөөлөх боломжтой.



Нээлттэй ба хаалттай байх зарчим

Бид хэрхэн нээлттэй ба хаалттай төлөвт байгаа
модултай байж болох вэ ? Хэрхэн А модулийг
хэвээр нь байлгаж, А' модулийг шинэ
клиентүүдийн хамт нэвтрүүлэх вэ ? Үүнийг
объект хандалтат технологийн удамшил
шийдвэрлэж чадна.

Нээлттэй ба хаалттай байх зарчим



Шинэ клиентүүдэд зориулсан шинэ модул
үүсгэх



Нээлттэй ба хаалттай байх зарчим

А модул өөрийн клиентүүдэд шаардлагатай үйлдлүүдтэй байна.

А' модул А модулийн үйлдлүүдээс гадна, шинэ клиентүүдийн үйлдлүүдийг агуулсан байна. А модулийн үйлдлүүд өөрчлөгдөхөд автоматаар А' модулийн үйлдлүүд шинэчлэгдэнэ.



Нээлттэй ба хаалттай байх зарчим

Ер нь програм хангамжийн модулийг засварлах үйл явц нь 2 байдлаар хийгдэж болно. Үүнд :

1. Програмыг дахин шинээр хийх. Энэ нь маш их цаг хугацаа шаардах боловч төгс арга
2. Нээлттэй ба хаалттай төлөв ашиглан, объект хандалтат технологийн удамшил ашиглаж шаардлагатай модулиудыг засварлах



Цор ганц сонголт

Програм хангамжийн хөгжүүлэлтийн үед сонголтийн үйлдлүүд гол үүрэгтэй байх тохиолдолд их байдаг. Жишээ нь :

- График системийн хувьд : дүрс нь полигон, тойрог, эллипс гэх мэт
- Програмчлалын хэлний хөрвүүлэгчийн хувьд : хэлний тэмдэглэгээ нь үйлдэл, илэрхийлэл, процедур байж болно.



Цор ганц сонголт

Мөн програмын хөгжүүлэлтийн явцад шинэ сонголт нэмэгдэх боломжтой.

Тодорхойлолт :

Програм хангамжын систем нь сонголтын хувилбаруудаар хангагдсан байх ёстой ба нэг эсвэл зөвхөн нэг модульд тэдгээр сонголтын хувилбарууд ыг тодорхойлсон байх ёстой.

Ингэснээр шинэ сонголт нэмэгдэхэд олон модульд өөрчлөлт оруулахгүй байх боломжтой болно.

Дахин ашиглах боломжийн асуудлууд - Reusability issues

- Зохион байгуулалтын асуудлууд:
- Техникийн асуудлууд: Ямар хэлбэрээр ?
 - Функц, процедуруудын сангууд - Routine libraries
 - Пакетууд - Packages (Ada)
 - Классын сангууд - Class libraries
 - Классуудын ямар хэлбэр ? What form of classes?

Reusability: Technical issues

► Хайлтын хэв загварын ерөнхий хэлбэр

The general pattern for a searching routine:

-- *exhausted* – *төгсгөлдөө хүрсэн*

has (*t*: *TABLE*; *x*: *ELEMENT*): *BOOLEAN* is

-- Does item *x* appear in table *t*?

local

pos: *POSITION*

do

from

pos := *initial_position* (*t*, *x*)

until

exhausted (*t*, *pos*) or else *found* (*t*, *x*, *pos*)

loop

pos := *next* (*t*, *x*, *pos*)

end

Result := *found* (*t*, *x*, *pos*)

end

Issues for a general searching module

► Type variation – Төрлийн хувилбар:

- What are the table elements?

► Routine grouping:

- Хайлтын функцээс гадна хүснэгт үүсгэх, элемент оруулах, устгах гэх мэт ф/п байх ёстой. A searching routine is not enough: it should be coupled with routines for table creation, insertion, deletion etc.

► Implementation variation:

- Many possible choices of data structures and algorithms: sequential table (sorted or unsorted), array, binary search tree, file, ...

Issues (cont'd)

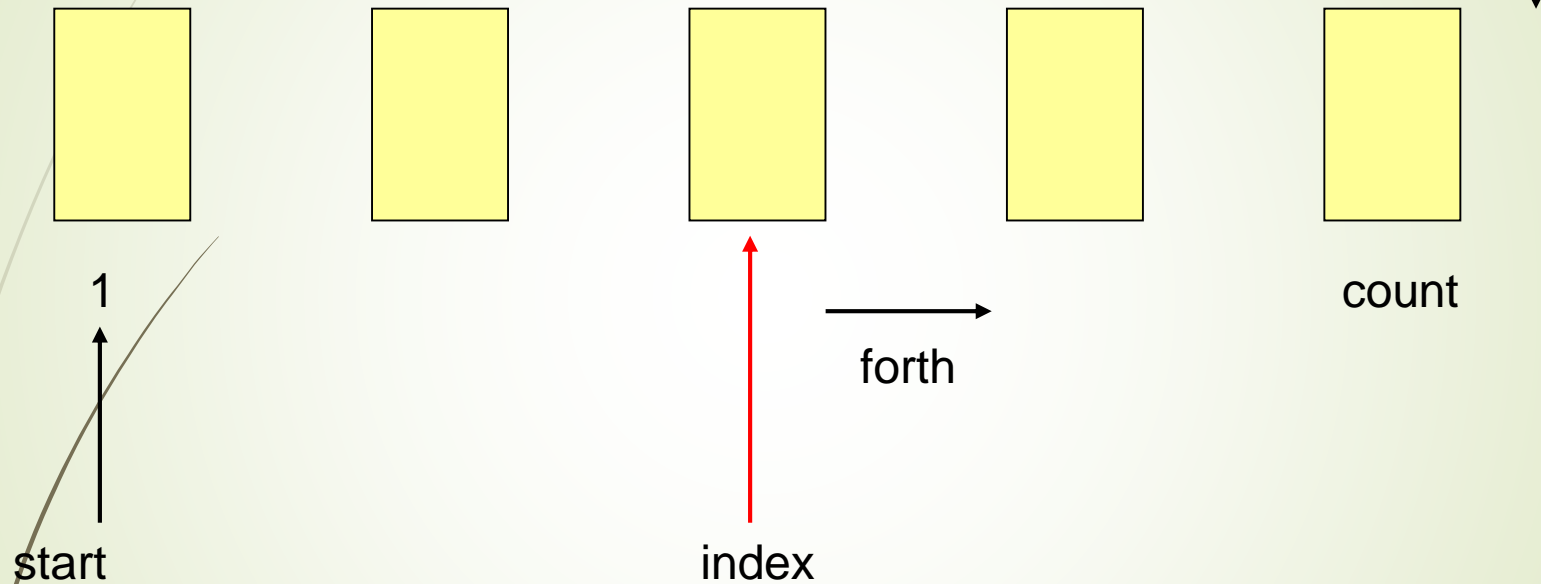
► Representation independence:

- Can a client request an operation such as table search (*has*) without knowing what implementation is used internally?
- Ямар зохион байгуулалт ашиглаж байгааг мэдэхгүйгээр хайх функцийг ингэж бичиж болох уу ?

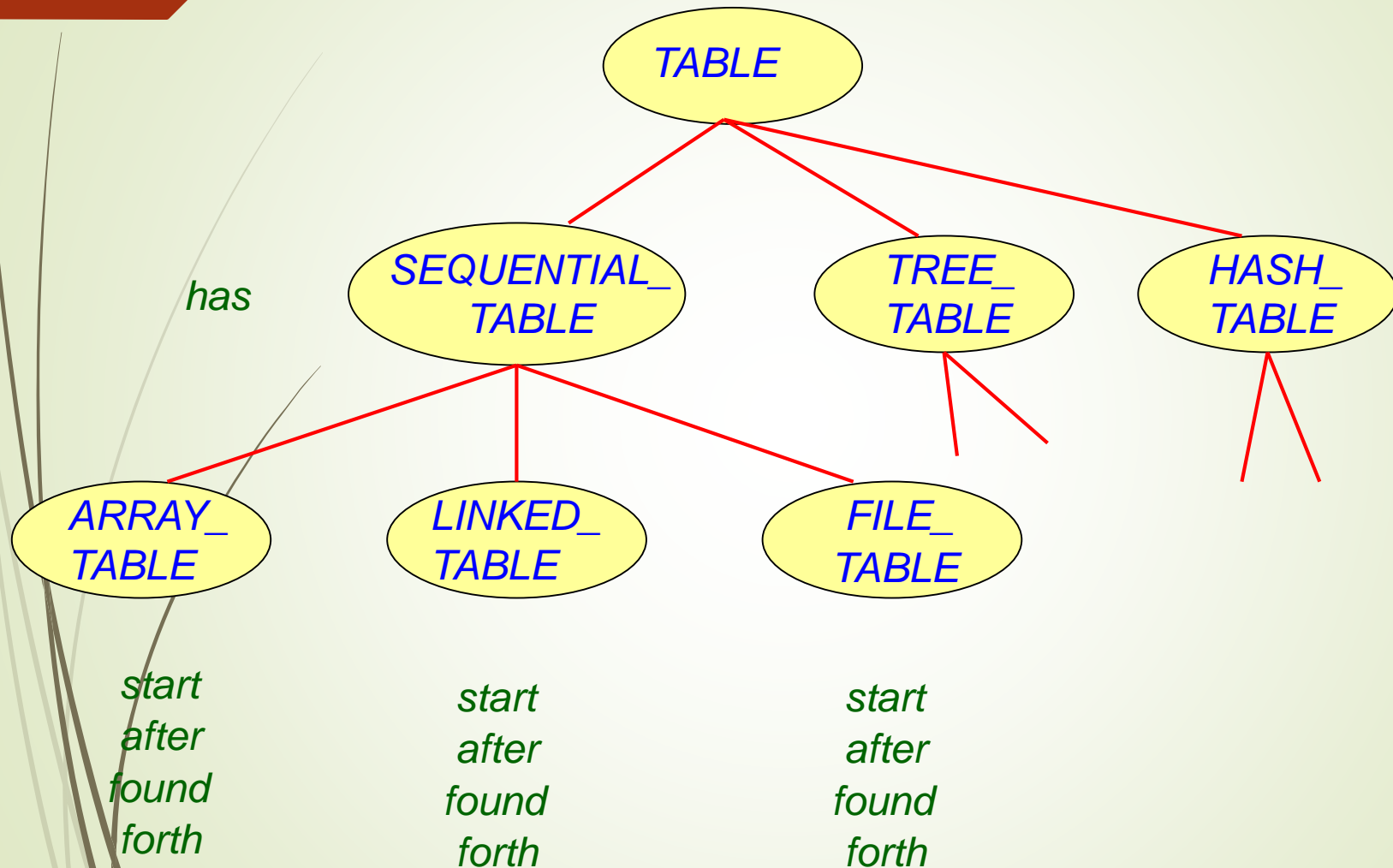
has (*tl*, *y*)

Factoring out commonality

нийтлэг байдлыг тодорхойлох



Factoring out commonality



Implementation variants

	start	forth	after	found (x)
Array	$i := 1$	$i := i + 1$	$i > \text{count}$	$t[i] = x$
Linked list	$c := \text{first_cell}$	$c := c.\text{right}$	$c := \text{Void}$	$c.\text{item} = x$
File	rewind	read	end_of_file	$f \uparrow = \xi$