

# Алгоритмчлалын үндэс: Дэд алгоритм

## Сэдвүүд:

- Дэд алгоритм гэж юу вэ?
- Дэд алгоритмыг дуудах
- Дэд алгоритм ба блок-схем
- Дэд алгоритмын хэлбэрүүд
- Процедур төрлийн дэд алгоритм
- Функц төрлийн дэд алгоритм
- Хуурмаг (формал) ба бодит (актуал) параметр
- Рекурс дэд алгоритм

## Дэд алгоритм гэж юу вэ?

**Дэд** гэдэг нь **бүрдэл хэсэг** гэсэн утгыг илэрхийлнэ. Програмчлалд **дэд алгоритм**, **дэд програм** гэх мэт ойлголтууд байдаг. **Дэд алгоритм** гэдэг нь ямар нэг алгоритмын бүрдэлд орж ажилладаг алгоритм гэсэн үг. Тиймээс дэд алгоритм нь биеэ дааж ажилладаггүй. Тэгвэл бүх дэд алгоритмыг өөртөө нэгтгэж байгаа тэр алгоритмыг **үндсэн алгоритм** гэж нэрлэж болно. Үндсэн алгоритм нь биеэ дааж ажиллана, ямар нэг алгоритмын бүрдэлд орохгүй. Харин үндсэн болон дэд алгоритмуудыг хамтад нь **ерөнхий алгоритм** хэмээн нэрлэж болно.

Дэд алгоритм нь өөрөө бас дэд алгоритмууд агуулж болдог. Дэд алгоритм нь өөрийн гэсэн дүрслэлтэй (блок-схем) байдаг.

Дэд алгоритмын ач холбогдол юу вэ? Алгоритм бол ямар нэг бодлогыг бодон, тодорхой үр дүн гаргаж авахын тулд дараалан гүйцэтгэх ёстой үйлдлүүдийн жагсаалт гэдгийг бид мэднэ. Хэрэв бодлого маань их олон үйлдэл хийж бодогдохоор том хэмжээтэй эсвэл төвөгтэй байвал алгоритм ч гэсэн том хэмжээтэй болно. Тэр хэмжээгээрээ тийм алгоритмыг зохиож, дүрслэх нь төвөгтэй болж ирнэ. Гэтэл алгоритм зохиогчийн **зорилго** бол **бодлогын алгоритм үнэн зөв** байхаас гадна **бас аль болох цомхоноор, хялбараар дүрслэгдсэн байх** явдал. Учир нь ингэж дүрслэгдэж чадсан алгоритм хүн харахад ойлгомжтой, эмх цэгцтэй байх болно. Үүнийг хэрэгжүүлэх арга замын нэг нь дэд алгоритм ашиглах явдал юм.

Дэд алгоритм ашиглахын тулд юуны өмнө өгөгдсөн бодлогыг олон **дэд бодлого** болгон хуваах хэрэгтэй. Дэд бодлого гэдэг нь анхдагч бодлогын зөвхөн тодорхой нэг хэсгийг бодох бодлого юм. Дараа нь дэд бодлого тус бүрийн алгоритмыг зохионо. Дэд бодлогын алгоритм нь **дэд алгоритм** юм. Эцэст нь бүх дэд алгоритмыг нэгтгэсэн үндсэн алгоритмыг зохионо.

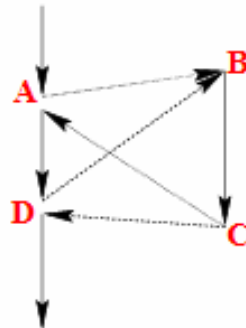
Дэд алгоритмуудаас тогтсон алгоритм ямар зарчмаар ажиллах вэ? Юуны өмнө, тухайн мөчид нэг л дэд алгоритм ажиллах байх ба нэгэн зэрэг нэгээс олон дэд алгоритм ажиллах ёсгүй. Ө.х. нэг дэд алгоритм ажиллаж байгаад нөгөө дэд алгоритм руу биелэлтийг шилжүүлнэ гэсэн үг. Ингэж шилжүүлэх механизмыг **дэд алгоритмыг дуудах (calling)** гэж нэрийднэ. Ингэж дуудахын тулд дэд алгоритм нь тодорхой нэрээр нэрлэгдсэн байна. Яг одоо ажиллаж буй дэд алгоритм нөгөө дэд алгоритмыг нэрээр нь дуудсанаар тэрхүү дуудагдсан дэд алгоритм ажиллаж эхлэнэ. Дуудагдсан дэд алгоритм ажиллаж дуусаад дуудсан дэд алгоритм руу биелэлтийг буцаан шилжүүлдэг.

Хамгийн түрүүнд үндсэн алгоритм ажиллаж эхлэнэ. Тэгээд өөр нэг дэд алгоритмыг дуудаж биелэлтийг шилжүүлнэ. Тэр нь ажиллаж байгаад дахиад өөр дэд алгоритм дуудаж

болно. Иймэрхүү байдлаар дэд алгоритмууд дэс дараалан биелсээр байгаад хамгийн сүүлд биелэлт үндсэн алгоритм руу дахин орж ирээд тэгээд үндсэн алгоритм ажиллаж дуусна.

Дараах зурагт, үндсэн алгоритм (YA) нэг дэд алгоритмыг (ДА) тодорхой интервалтайгаар хоёр удаа дуудаж буй процессыг схемчлэн үзүүлжээ.

үндсэн алгоритм дэд алгоритм



YA нь A хүртэл биелээд ДА-ыг дуудаж байна. ДА нь В-ээс С хүртэл биелж дуусаад YA-д биелэлтийг шилжүүлнэ. YA биелэлтийг хүлээн авсан газраасаа D хүртэл биелээд дахиад ДА-ыг дуудаж байна. ДА биелж дуусаад биелэлтээ дахиад YA-д шилжүүлж байна. YA цааш үргэлжлэн биелэнэ. Хэрэв бид ВС хэсгийн үйлдлүүдийг ингэж дэд алгоритм болгож салгалгүй шууд үндсэн алгоритмд “залгасан” байсан бол үндсэн алгоритмын дүрслэл 2 удаа ВС-ийн “уртаар” “нэмэгдэх” байсан:

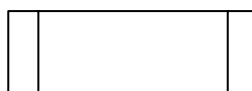
үндсэн алгоритм



Ө.х. дэд алгоритм ашигласан тохиолдолд үндсэн алгоритмын дүрслэл илүү цомхон болсон байжээ. Энд тэмдэглэж хэлэхэд, алгоритмын дүрслэл цомхогдоно гэдэг нь түүний бүтэц цомхогдож байна гэсэн үг биш гэдгийг ойлгоорой. Дэд алгоритм ашигласан тохиолдолд үндсэн алгоритмын дүрслэл л цомхогдож байгаагаас биш бүтэц нь цомхогдож байгаа хэрэг огтхон ч биш юм!

**Дэд алгоритмыг дуудах**

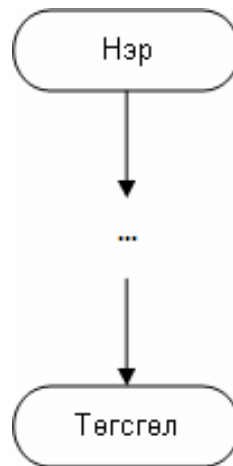
Дэд алгоритмыг дуудахдаа:



гэсэн элементийг ашиглана. Энэ нь **Predefined process** гэдэг элемент юм. Дуудаж буй дэд алгоритмын нэрийг дотор нь бичдэг.

### Дэд алгоритм ба блок-схем

Дэд алгоритмын блок-схем дүрслэл нь ердийн алгоритмын дүрслэлээс онц ялгаа байхгүй. Мөн л адилхан **Эхлэл/Төгсгөл**, **Процесс**, **Салаалалт**, **Давталт** бүтцүүд болон **Оролт/Гаралтын** үйлдэл агуулна. Гэхдээ **Эхлэл** элемент дотор дэд алгоритмын нэрийг бичнэ. Үүгээрээ л үндсэн алгоритмын блок-схемээс ялгагдана:



### Дэд алгоритмын хэлбэрүүд

Хоёр хэлбэрийн дэд алгоритм байдаг:

- Процедур (procedure)
- Функц (function)

Өөрийг нь дуудсан газарт биелснийхээ дараа ямар нэг утга болж үлддэг дэд алгоритмыг **функц** гэнэ. Харин **процедур** хэлбэрийн дэд алгоритм биелснийхээ дараа ингэж утга хэлбэрээр үлдэхгүй. Функц болон процедур дэд алгоритмын талаар хойно дэлгэрэнгүй үзнэ.

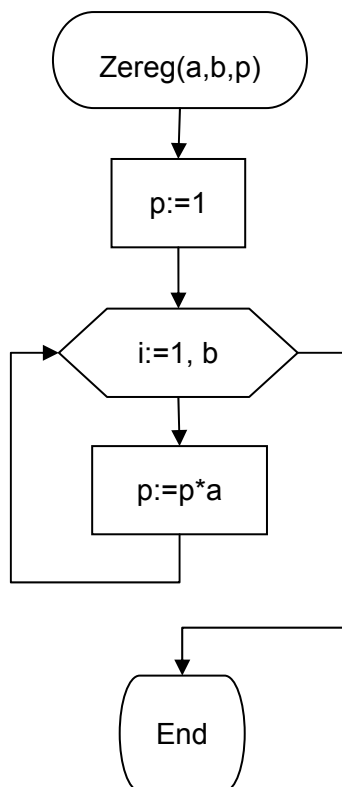
#### Процедур төрлийн дэд алгоритмын жишээ

Одоо дэд алгоритмын талаар жишээ авч ярилцъя. Ийм бодлого байг. Өгөгдсөн  $x$ ,  $y$ ,  $N$  тоонуудыг ашиглан:

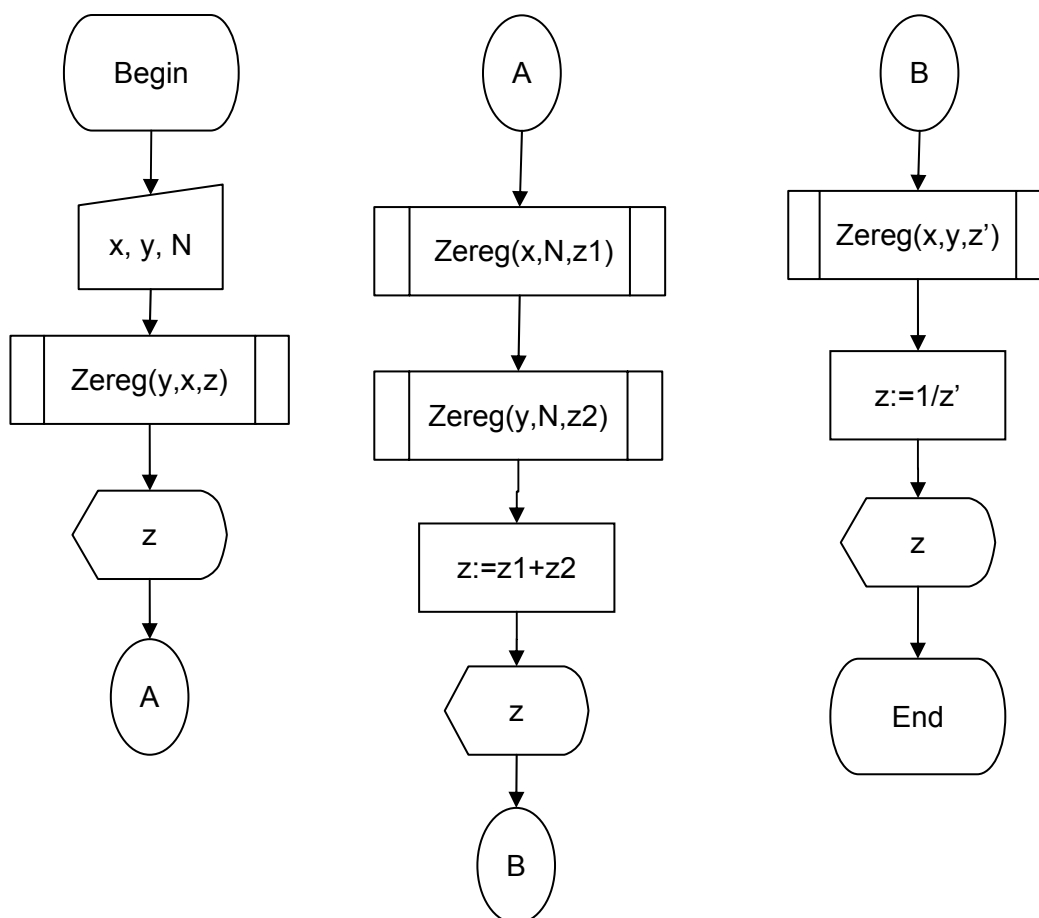
1.  $z=y^x$ -ийг олж гаралтанд өгдөг
2.  $z=x^N+y^N$ -ийг олж гаралтанд өгдөг
3. эцэст нь  $z=1/x^y$ -ийг олж гаралтанд өгдөг алгоритм зохио.

Эхлээд бодлогоо боломжит дэд бодлогуудад хуваана. Манай бодлогын хувьд зэрэгт дэвшүүлэх үйлдэл 4 удаа давтагдсан байгаа. Тиймээс уг үйлдлийг нэг дэд бодлого болгож аваад харгалзах дэд алгоритмыг зохиох хэрэгтэй.

Өгсөн  $a$  тоог өгсөн  $b$  натурал тоонд зэрэг дэвшүүлээд хариуг нь  $p$ -д хийдэг, ө.х.  $p=a^b$  олдог процедур төрлийн дэд алгоритмыг **Zereg** хэмээн нэрийдье. Түүний блок схем нь:



гэсэн хэлбэртэй байх болно. Одоо уг дэд алгоритмыг 4 удаа дуудаж ашигласан үндсэн алгоритмыг зурвал:



гэсэн хэлбэртэй байх болно. Хамгийн эхний удаа Zereg-ийг дуудахдаа үндсэн алгоритмаас түүн рүү x, y-ийн утгыг дамжуулж, z-ийн утгыг хүлээн авч байна. Ө.х. a-ийн оронд x-ийг, b-ийн оронд y-ийг, p-ийн оронд z-ийг дамжуулж байна. Тиймээс дэд алгоритм x-ийг y зэрэгт дэвшүүлж хариуг нь z рүү хийнэ. Тэгсний дараа үндсэн алгоритм z-ийг гаралтанд өгч байна. Үүний адилаар дэд алгоритмыг дахин гурван удаа дуудаж бодолт хийлгэсэн байгаа.

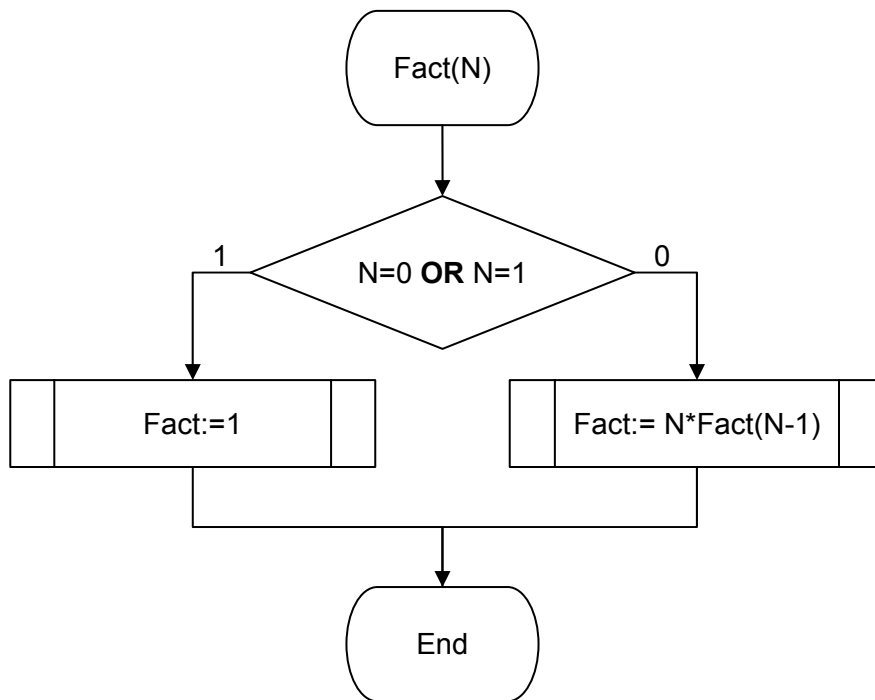
**Функц төрлийн дэд алгоритмын жишээ**

**Хуурмаг (формал) ба бодит (актуал) параметрууд**

**Рекурс дэд алгоритм**

Өөрөө өөрийгөө дууддаг дэд алгоритмыг **шууд рекурс** алгоритм гэнэ. Бас хоёр дэд алгоритм нэг нэгнээ харилцан дуудсан байж болдог. Үүнийг **шууд бус рекурс** алгоритмууд гэнэ.

Шууд рекурсын жишээ болгож өгөгдсөн натурал тооны факториалыг олдог функц төрлийн дэд алгоритм авч үзье. Энэ аргын гол санаа нь N-р тооны факториалыг олохын тулд түүнийг (N-1)-р тооны факториалаар үржих явдал юм.



N>1 тохиолдолд энэ дэд алгоритм өөрийгөө N-1 гэсэн параметртайгаар 1 удаа дуудна, тэр нь цаанаа N-2 гэсэн параметртайгаар өөрийгөө дахин дуудна, тэр нь цаанаа N-3 гэсэн параметртайгаар өөрийгөө дуудна гэх мэтээр нийт N-1 удаа өөрийгөө дуудах болно. Ингэснээр:

$$N!=N*(N-1)!=N*(N-1)*(N-2)!= N*(N-1)*(N-2)*(N-3)!=...= N*(N-1)*(N-2)*(N-3)*...*(N-(N-1))$$

гэсэн үржвэр хийгдэж байгаа юм. Өөрөө өөрийгөө дуудах процесс хамгийн сүүлд Fact(1)-ийг дуудсанаар зогсоно. Ж.нь N=3 байвал Fact(3)=3\*Fact(2)=3\*2\*Fact(1)=3\*2\*1=6 гэж гарна.

Мэдээж факториал олдог энэ бодлогыг рекурс ашиглахгүйгээр шийдэх боломжтой (ж.нь давталт ашиглан). Рекурс ашигласны давуу тал гэвэл алгоритмын дүрслэл цомхон болж өгч байгаа юм.