

# Функц

## Сэдвүүд:

- Програмчлалын технологийн тухай
- Процедурт програмчлал
- Функцын тухай
- Функц тодорхойлох (зохиох)
- Функц зарлах
- Функц дуудах
- Формал ба бодит параметрууд
- Функцтэй өгөгдөл солилцох
- Функцын заагч
- Рекурс функц
- Аргументын тоо нь хувьсдаг функц

## Програмчлалын технологийн тухай

Програмчлалын хэлээр програм бичихдээ тодорхой хэв маягийг (стиль) баримтлах ёстой. Үүнийг **програмчлалын технологи** эсвэл **програмчлах техник (programming technique)** гэдэг.

Програмчлалын үндсэн технологиуд:

- Процедурт програмчлал (Procedural programming)
- Модульт програмчлал (Modular programming)
- Объект хандлагат програмчлал (Object-Oriented Programming)

г.м.

Эдгээрээс процедурт програмчлалын технологийг авч үзье.

## Процедурт програмчлал

**Процедурт програмчлал (Procedural programming)** гэдэг нь, програмыг өөр хоорондоо холбоотой хэд хэдэн дэд хэсгүүд - **дэд програмуудаас** тогтсон, нарийн эмх журамтай цогц мэтээр авч үздэг технологи.

**Дэд** гэдэг нь **бүрдэл хэсэг** гэсэн утгыг илэрхийлж байгаа. **Дэд програм** гэдэг нь ямар нэг програмын бүрдэлд орж ажилладаг програм юм. Ө.х. биеэ дааж ажилладаггүй. Ер нь бол програмчлалын хэлээр илэрхийлэгдсэн дэд алгоритмыг дэд програм гэж хэлж болно.

Дэд програм нь өөрөө бас дэд програмууд агуулж болдог.

Дэд програм нь өөрийн гэсэн кодтой байна.

Дэд програмын ач холбогдол юу вэ?

Програм бол өгөгдсөн бодлогыг хэрхэн бодох вэ гэсэн заавруудыг (алгоритмыг) програмчлалын хэлээр илэрхийлсэн бичиглэл гэдгийг бид мэднэ. Хэрэв бодлого маань их олон үйлдэл хийж бодогдохоор том хэмжээтэй эсвэл төвөгтэй байвал програм ч гэсэн том хэмжээтэй болно. Энэ тохиолдолд, тухайн бодлогыг олон **дэд бодлого** болгон хувааж, дэд бодлого бүрийн програмыг тус бүрд нь бичиж эцэст нь нийлүүлэх маягаар ажиллавал нэн зохистой байж магадгүй юм. Дэд бодлого гэдэг нь анхдагч бодлогын зөвхөн тодорхой нэг хэсгийг бодох бодлого юм. Дэд бодлогын програм нь дэд програм байна.

Дэд програмуудаас тогтсон програм ямар зарчмаар биелдэг вэ? Юуны өмнө, тухайн мөчид нэг л дэд програм биелж байдаг ба нэгэн зэрэг нэгээс олон дэд програм биелдэггүй.

Ө.х. нэг дэд програм биелж дуусаад нөгөө дэд програм руу удирдлага шилжинэ гэсэн үг. Ингэж шилжүүлэх механизмыг **дэд програмыг дуудах (procedure calling)** гэж нэрийддэг. Ингэж дуудахын тулд дэд програм нь тодорхой нэрээр нэрлэгдсэн байна. Яг одоо биелж буй дэд програм нөгөө дэд програмыг нэрээр нь дуудсанаар тэрхүү дуудагдсан дэд програм биелж эхлэнэ. Дуудагдсан дэд програм биелж дуусаад дуудсан дэд програм руу биелэлтийг буцаан шилжүүлдэг.

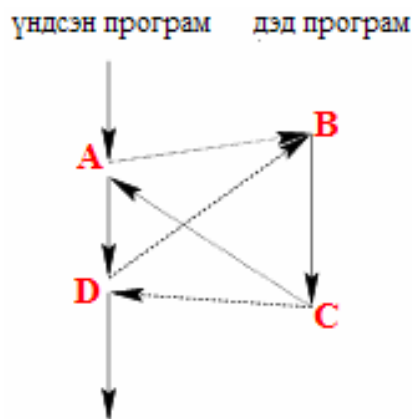
Ингэж дэд програмууд нэг нь нөгөөгөө дуудах зарчмаар програм бүхэлдээ биелэхийн тулд бусад бүх дэд програмуудаасаа давуу эрхтэй нэг дэд програм заавал байх ёстой болдог. Үүнийг **үндсэн дэд програм** буюу товчоор **үндсэн програм (main program)** гэдэг. Үйлдлийн системээс програмыг анх дуудахад (ачаалахад) чухамхүү үндсэн дэд програм л хамгийн түрүүнд биелж эхлэнэ. Тэгээд бусад дэд програмыг дуудаж биелүүлэлтийг шилжүүлдэг байна. Дуудагдсан дэд програмууд нь биелж байгаад өөрсдөө дахиад бас дэд програмууд дуудаж болно. Иймэрхүү байдлаар дэд програмууд дэс дараалан биелсээр хамгийн сүүлд биелэлт үндсэн програмд буцаж орж ирээд тэгээд үндсэн програм ажиллаж дуусна. Ө.х. програмын биелэлт үндсэн дэд програмаар эхлэж, үндсэн дэд програмаар дуусдаг байна.

Дурдсан бүхнийг нэгтгээд:

$$\text{Процедурт програм} = \text{Үндсэн (дэд) програм} + \text{Дэд програмууд}$$

гэж томъёолж болно.

Дараах зураг дээр, үндсэн програм (ҮП) нэг дэд програмыг (ДП) тодорхой интервалтайгаар хоёр удаа дуудаж буй процессыг схемчлэн үзүүлжээ.



ҮП нь А хүртэл биелээд ДП-ыг дуудаж байна. ДП нь В-ээс С хүртэл биелж дуусаад ҮП-д биелэлтийг шилжүүлнэ. ҮП биелэлтийг хүлээж авсан газраасаа D хүртэл биелээд дахиад ДП-ыг дуудаж байна. ДП биелж дуусаад биелэлтийг дахиад ҮП-д шилжүүлж байна. ҮП цааш үргэлжлэн биелэнэ. . Хэрэв бид ВС хэсгийн үйлдлүүдийг ингэж дэд програм болгож салгалгүй шууд үндсэн програмд “залгасан” байсан бол үндсэн програмын дүрслэл 2 удаа ВС-ийн “уртаар” “нэмэгдэх” байсан:

үндсэн алгоритм



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

Дэд програм нь:

- **процедур (procedure)**
- **функц (function)**

гэсэн хэлбэрүүдтэй байна.

Функц, процедур болгон өөрийн гэсэн оролт/гаралт буюу эхлэл/төгсгөлтэй байна. Тэдгээр нь үндсэн програмтай болон бусад дэд програмтай оролт/гаралтанд илгээгдсэн өгөгдлүүдээр холбогдоно.

Функц хэлбэрийн дэд програм нь утгатай (хариутай) байна. Үүнийг **функцын буцаах утга (return value)** гэдэг. Процедур төрлийн дэд програмд буцаах утга гэж байхгүй.

Дэд програмыг гарал үүслийн хувьд:

- **стандарт (standard)**
- **хэрэглэгчийн (user-defined)**

гэж ангилдаг.

Програмчлалын хэлэнд урьдаас тодорхойлогдож, түүний хөрвүүлэгчийн бүрдэлд орсон байдаг дэд програмыг **стандарт дэд програм** гэнэ. Байнга хийгдэж байдаг үйлдлүүдийг ингэж стандарт дэд програм болгодог. Ө.х. хэрэглэгч нь стандарт дэд програмыг бэлнээр нь шууд ашиглана.

Олон тооны стандарт дэд програмууд нийлээд тухайн хэлний **стандарт санг (standard library)** бүрдүүлнэ.

## Функцын тухай

Си хэлэнд зөвхөн функц төрлийн дэд програм байдаг. Си хэлний функцыг гарал үүслийн хувьд:

- **дотоод (intrinsic)** буюу **бэлэн (predefined)** функц
- **гадаад (external)** буюу **хэрэглэгчийн зохиосон (user-defined)** функц

гэж ангилна.

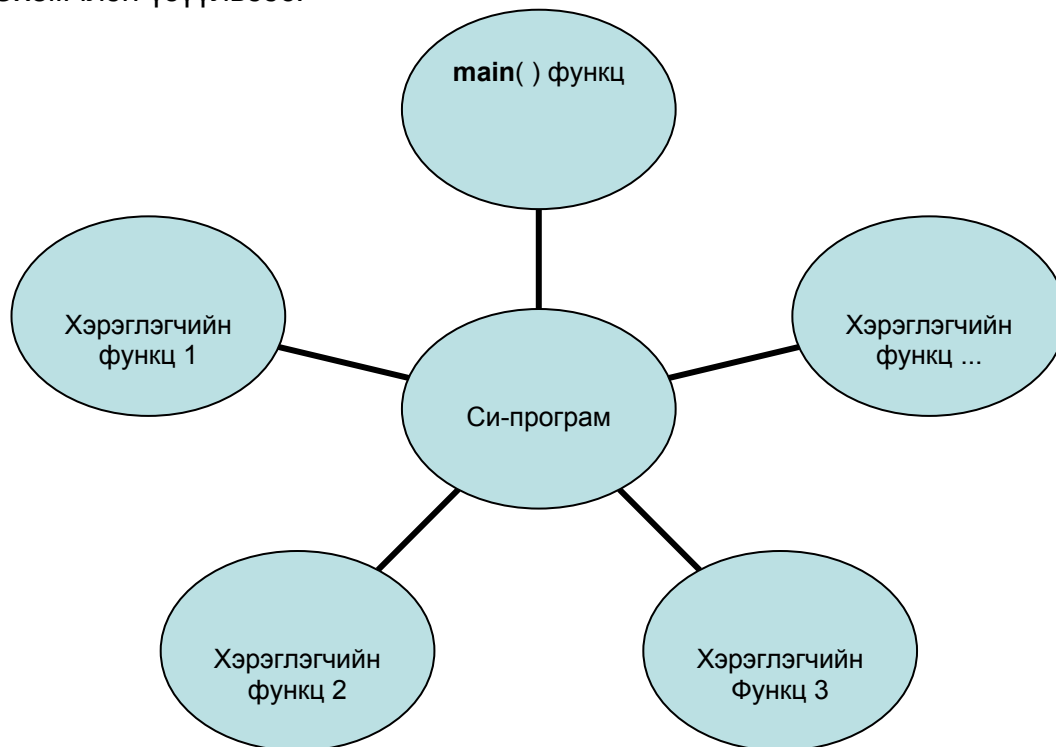
Predefined функцын жишээ бол стандарт функц юм: **printf()**, **scanf()**, **getchar()** г.м. Ийм функц нь хөрвүүлэгчтэй хамт ирдэг. Хэрэглэгч ийм функцыг, ямар кодтой, ямар бүтэцтэй гэдгийг нь нэг их анзааралгүйгээр зөвхөн юу хийдэг вэ гэдгийг нь мэдэж байгаад л

шууд нэрээр нь дуудаж ашигладаг. Тэгвэл хэрэглэгчийн функцыг бол бид өөрсдөө зохионо., ө.х. кодыг нь бичнэ, бас дуудаж ашиглана.

Програмыг функцуудэд хуваах нь олон ач холбогдолтой. Ингэснээр ж.нь програмыг оператор, өгөгдлүүдийн харилцан үйлчлэл мэтээр авч үзэх төвшнөөс нэг шат ахиж функцуудын харилцан үйлчлэл мэтээр авч үзэж болно. Үүний дүнд програмын бүтэц энгийн, ойлгомжтой болж ирдэг.

Түүнээс гадна програмын бүрдэл хэсгүүдийн давхарлал арилж, хэмжээ нь цомхон болно. Учир нь нэг функцыг ганц удаа зохиогоод л дараа нь түүнийгээ програмын аль хэрэгтэй газарт зөвхөн нэрээр нь дуудаж ашиглана. Яг стандарт функцын адилаар.

Схемчлэн үзүүлвээс:



Үндсэн програмын үүргийг **main( )** функц гүйцэтгэнэ. Нэг програмд нэг л **main( )** функц байх ёстой.

### Функц тодорхойлох (зохиох)

Хэрэглэгчийн функцын ерөнхий бүтэц нь:

```
<төрөл> функцын_нэр(<төрөл> парам1, <төрөл> парам2, ...) //функцын толгой буюу прототип
{
    ...
    return <утга>;
}
```

гэсэн хэлбэртэй байна. Энд **<төрөл>** - функцын буцаах утгын төрөл бөгөөд char, unsigned char, int, unsigned int, float, double г.м.-ийн аль нэг байна.

Утга буцаадаггүй функц гэж бас бий. Ийм функц нь:

```
void функцын_нэр(<төрөл> парам1, <төрөл> парам2, ...) //функцын толгой буюу прототип
{
    ...
}
```

гэсэн хэлбэртэй байна.

Функц ашигласан програмын ерөнхий бүтцийг харуулвал:

```

Прототип1;
Прототип2;
...
main()
{
    ...
}
<төрөл> функц1(<төрөл> парам1, <төрөл> парам2, ...)
{
    ...
}
<төрөл> функц2(<төрөл> парам1, <төрөл> парам2, ...)
{
    ...
}
}

```

гэсэн хэлбэртэй байна.

Жишээ нь өгөгдсөн  $x$ ,  $y$ ,  $N$  тоонуудыг ашиглаад:

1.  $z=y^x$ -ийг олж  $z$ -г гаралтанд өгдөг
2.  $z=x^N+y^N$ -ийг олж  $z$ -г гаралтанд өгдөг
3.  $z=1/x^y$ -ийг олж  $z$ -г гаралтанд өгдөг

програм бичье. Бүх гурван тооцоонд давтагдаж буй үйлдэл бол зэрэг дэвшүүлэх үйлдэл байна. Тиймээс үүнийг дэд бодлого болгон аваад харгалзах дэд програм – функцийг бичнэ.

```

#include<stdio.h>
#include<conio.h>
int zereg(int a, int b);    //zereg() функцийн прототип
main()                    //вндсэн функц
{
    int x, y, N, z;
    float f;
    x=5;
    y=3;
    N=4;
    z=zereg(y,x);        //z=y^x-ийг олж байна
    printf("y^x=%d^%d=%d\n", y, x, z);
    z=zereg(x,N)+zereg(y,N);    //z= x^N+y^N-ийг олж байна
    printf("x^N+y^N=%d^%d+%d^%d=%d\n", x, N, y, N, z);
    f=1.0/zereg(x, y);    //f=1/x^y-ийг олж байна
    printf("1/x^y=1/%d^%d=%f\n", x, y, f);
    getch();
    return 0;
}

int zereg(int a, int b)    //zereg() функц
{                          //функцин бие эндээс эхэлж байна
    int i, c;
    c=1;
    for(i=0; i<b; i++) c=c*a;
    return c;              //функцин буцаах утга
}

```

```

#include<stdio.h>
#include<conio.h>

void zereg(int a, int b, int *c); //zereg() функцын прототип

main()           //үндсэн функц
{
  int x, y, N, z, z1, z2;
  float f;
  x=5;
  y=3;
  N=4;
  zereg(y,x,&z);
  printf("y^x=%d^%d=%d\n", y, x, z);
  zereg(x,N,&z1);
  zereg(y,N,&z2);
  z=z1+z2;
  printf("x^N+y^N=%d^%d+%d^%d=%d\n", x, N, y, N, z);
  zereg(x,y,&z);
  f=1.0/z;
  printf("1/x^y=1/%d^%d=%f\n", x, y, f);
  getch();
  return 0;
}

void zereg(int a, int b, int *c)   //zereg() функц
{                                   //функцын бие эндээс эхэлж байна
  int i, p;
  p=1;
  for(i=0; i<b; i++) p=p*a;
  *c=p;
}

```

### Формал ба бодит параметрууд

Хэрэглэгчийн функцыг зарлахдаа мөн тодорхойлохдоо нэрийнх нь ард хаалтанд дотор бичиж буй аргументуудыг **формал (хуурмаг) параметрууд** гэнэ. Харин тухайн функцыг дуудахдаа нэрнийх нь ард бичиж буй аргументуудыг **бодит параметрууд (actual argument)** гэнэ.

### Рекурс функц

Өөрөө өөрийгөө дууддаг функцыг рекурс функц гэнэ.