CS200 – Програмчлалын үндэс

Лекц 04

Си функц

Профессор А.Эрдэнэбаатар

Лекцийн агуулга

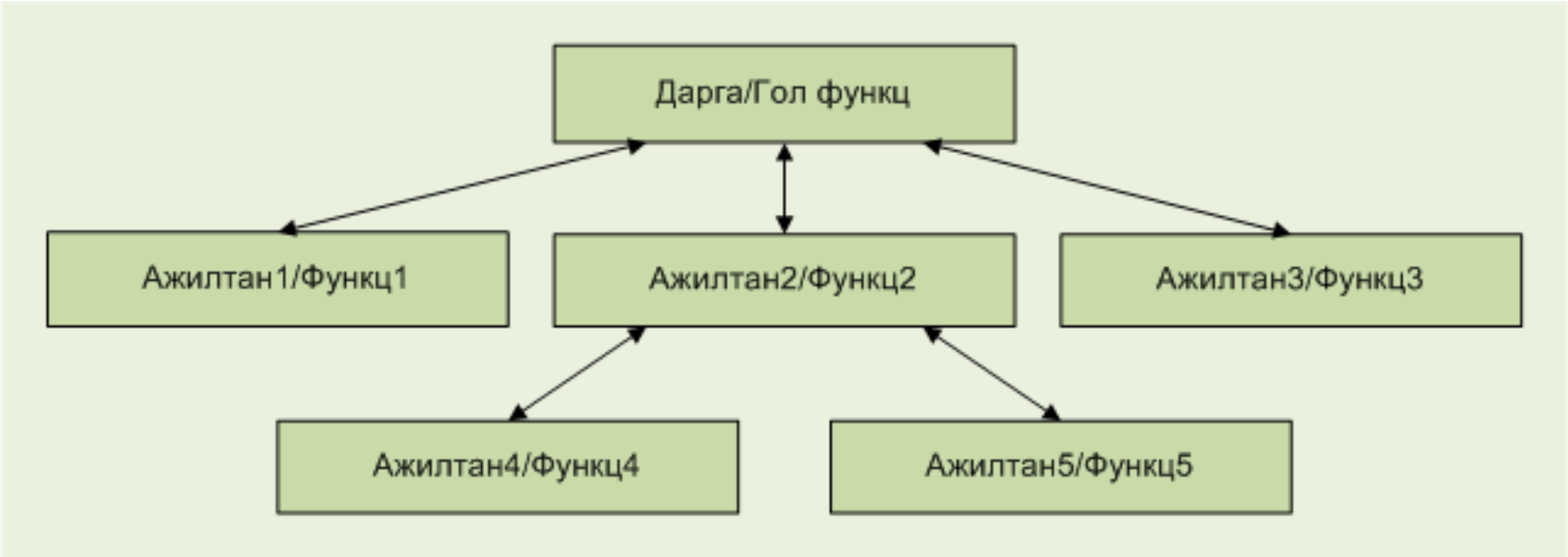
- Функц гэх жижиг модуль – блокоор програм бүтээх
- Си стандарт сан дахь зарим математик функцүүд
- Шинээр функц яаж бүтээх вэ?
- Функцүүдийн хооронд мэдээлэл дамжуулах механизм
- Рекурсив (өөрийгөө дууддаг) функцийг яаж бичих вэ?
- Дүгнэлт

Си хэлний програмын модуль

- Функц
  - Функц бол Си хэлний модуль
  - Програм нь хэрэглэгчийн зохиосон болон сангийн функцүүдийн нэгдлээс бүтдэг (Си хэлний стандарт санд маш олон тооны функц байдаг)
- Функцийг дуудах
  - Функцийг дуудахад
    - Функцийн нэр, аргумент (өгөгдлийг) өгнө
    - Функц нь ямар нэг үйл, боловсруулалтыг хийнэ
    - Функц утга (үр дүн) буцаана
- Функц дуудахын төсөөтэй жишээ:
  - Дарга ажилтандаа ажил гүйцэтгэх даалгавар өгнө
    - Ажилтан мэдээлэл авч, ажлыг гүйцэтгээд үр дүнг буцаана
    - Мэдээллийн далдлалт: дарга гүйцэтгэлийн нарийн ширийнийг мэдэхгүй

Дарга-Ажилтан/Функцийн шаталсан харьцаа

Сум:
Доошоо – Аргумент
Дээшээ – Буцах утга



Лекцийн агуулга

- Функц гэх жижиг модуль – блокоор програм бүтээх
- Си стандарт сан дахь зарим математик функцүүд
- Шинээр функц яаж бүтээх вэ?
- Функцүүдийн хооронд мэдээлэл дамжуулах механизм
- Рекурсив (өөрийгөө дууддаг) функцийг яаж бичих вэ?
- Дүгнэлт

Сангийн математик функцүүд

- Сангийн математик функцүүд
  - Нийтлэг математик тооцооллыг гүйцэтгэдэг
  - Толгой файл: `#include <math.h>`
- Функцийг дуудах загвар

FunctionName (argument);

  - Олон аргументтэй бол таслалаар зааглана

printf(“%.2f”, sqrt(900.0));

  - Аргументээс квадрат язгуур гаргах `sqrt` функцийг дуудаж байна
  - Бүх математик функцийн буцах утга `double` төрөл байна
  - Аргумент нь тогтмол, хувьсагч, илэрхийлэл байж болно

Өргөн хэрэглэгддэг математик функцүүд

Функц	Тодохойлолт	Жишээ
`sqrt( x )`	Кватрат язгуур	`sqrt( 900.0) –> 30.0`
`exp( x )`	Экспоненциаль функц e^x	`exp( 2.0) –> 7.389056`
`log( x )`	e суурьт натураль логарифм	`log( 2.718282 ) –> 1.0`
`log10( x )`	10 суурьт натураль логарифм	`log10( 100.0 ) –> 2.0`
`fabs( x )`	Абсолют утга	`fabs(-5.0 ) –> 5.0`
`cell( x )`	x –с бага биш хамгийн бага бүхэл болгож нарийвчлах	`ceil( -9.8 ) –> -9.0` `ceil( 9.2 ) –> 10.0`
`floor( x )`	x –с их биш хамгийн их бүхэл болгож нарийвчлах	`floor( -9.8 ) –> -9.0` `floor( 9.2 ) –> 9.0`
`pow( x, y )`	Илтгэгч функц x^y	`pow( 2, 7 ) –> 128.0`
`fmod( x, y )`	x/y –н хөвөгч таслалтай үлдэгдэл	`fmod(13.657, 2.333) –> 1.992`
`sin( x )`	Тригнометрийн функц - синус	`sin( 30.0 ) –> 0.5`
`cos( x )`	Тригнометрийн функц – косинус	`cos( 60.0 ) –> 0.5`
`tan( x )`	Тригнометрийн функц - тангенс	`tan( 45.0 ) –> 1.0`

Лекцийн агуулга

- Функц гэх жижиг модуль – блокоор програм бүтээх
- Си стандарт сан дахь зарим математик функцүүд
- Шинээр функц яаж бүтээх вэ?
- Функцүүдийн хооронд мэдээлэл дамжуулах механизм
- Рекурсив (өөрийгөө дууддаг) функцийг яаж бичих вэ?
- Дүгнэлт

Функц

- Функц
  - Програмыг модульчилах
  - Бүх хувьсагчийг функцийн дотор локаль хувьсагч болгон зарлана. Зөвхөн функц дотор л харагдана
  - Параметр - локаль хувьсагчаар дамжин функцүүдийн хооронд мэдэлэл солилцогдоно
- Функцийн ашиг тус
  - “Хуваан захирах” гэсэн програм хөгжүүлэлтийн аргыг хэрэгжүүлэх
  - Програмыг дахин ашиглах. Байгаа функцийг шинэ програмын блок болгон ашиглах, доторх нарийн бүтцийг далдалж, хийсвэржүүлэх (сангийн функцүүд)
  - Кодын давталтыг арилгах

Зөвлөмж

- Олон функцтэй програмд `main` функц нь програмын багцалсан ажлуудлыг гүйцэтгэдэг функцүүдийг дуудсан маягтай байдаг.
- Функц бүр сайн тодорхойлогдсон нэг л асуудлыг шийдэх ба функцийн нэр нь юу хийж байгааг нь илэрхийлэх ёстой.

Функцийн тодорхойлолт

- Функц дараах загвартай байна

```
буцах-утгын-төрөл  функцийн-нэр ( параметрийн-жагсаалт  )
  {
    зарлалт ба операторууд
  }
```

  - Функцийн нэр: дурын зөвшөөрөгдсөн идентификатор
  - Буцах утгын төрөл: үр дүнгийн өгөгдлийн төрөл (өгөгдмөл нь **`int`**). Буцах утгагүй бол **`void`** төрөл байна
  - Параметрийн жагсаалт: таслалаар зааглагдсан жагсаалтаар параметрүүдийг зарладаг. Параметр тус бүрийн төрлийг заах ёстой ба байхгүй бол int гэж үзнэ
  - Зарлал ба операторууд: функцийн их бие(блок). Хувьсагчдийг блокын дотор тодорхойлж (үүрлэж)болно. Функцийг өөр функцийн дотор тодорхойлж болохгүй
  - Буцалтыг удирдах. Юу ч буцаахгүй бол **`return;`** эсхүл баруун **`}`** хаалт хүрээд функцээс буцна. Юм буцааж байвал **`return expression;`** операторoор буцна

```c
/* Ex_17 Creating and using a programmer-defined function */
#include <stdio.h>

int square( int y ); /* function prototype */

/* function main begins program execution */
int main( void )
{
   int x; /* counter */

   /* loop 10 times and calculate and output square of x each time */
   for ( x = 1; x <= 10; x++ ) {
      printf( "%d  ", square( x ) ); /* function call */
   } /* end for */

   printf( "\n" );

   return 0; /* indicates successful termination */

} /* end main */

/* square function definition returns square of parameter */
int square( int y ) /* y is a copy of argument to function */
{
   return y * y; /* returns square of y as an int */

} /* end function square */
```

```
1   4   9   16   25   36   49   64   81   100
```

Функцийн загвар нь функц өөрөө дараа тодорхойлогдоно гэдгийг заана

square функцийг дуудаж байна

Функцийн тодорхойлолт

Функцийн загвар

- Функцийн загвар
  - Функцийн нэр
  - Параметрууд – гаднаас авах
  - Буцах төрөл – функцээс буцах өгөгдлийн төрөл
  - Функцийн тодорхойлолт нь програмд түүнийг ашигласнаас хойно гарч ирэх тохиолдолд л загварыг хэрэглэдэг
  - Функцийн загварын жишээ:

    ```
    int maximum( int x, int y, int z );
    ```

    - Функц 3 бүхэл утга авч, бүхэл утга буцааж байна
  - Дэвшүүлэх, хувиргах дүрэм
    - Өгөгдлийг доод түвшний төрөлд хувиргах нь алдаанд хүргэж болзошгүй

Өгөгдлийн төрлийн дэвшүүлэх эрэмбэ

дээд

доод

Өгөгдлийн төрөл	Printf –н хувиргалт	Scanf –н хувиргалт
`long double`	`%lf`	`%lf`
`double`	`%f`	`%f`
`float`	`%f`	`%f`
`unsigned long int`	`%lu`	`%lu`
`long int`	`%ld`	`%ld`
`unsigned int`	`%u`	`%u`
`int`	`%d`	`%d`
`unsigned short`	`%hu`	`%hu`
`short`	`%hd`	`%hd`
`char`	`%c`	`%c`

Функцийн жишээ

```
1.  /* Finding the maximum of three integers */
2.  #include <stdio.h>
3.
4.  int maximum( int x, int y, int z ); /* function prototype */
5.
6.  /* function main begins program execution */
7.  int main( void )
8.  {
9.     int number1; /* first integer */
10.    int number2; /* second integer */
11.    int number3; /* third integer */
12.
13.    printf( "Enter three integers: " );
14.    scanf( "%d%d%d", &number1, &number2, &number3 );
15.
16.    /* number1, number2 and number3 are arguments
17.       to the maximum function call */
18.    printf( "Maximum is: %d\n", maximum( number1, number2, number3 ) );
19.
20.    return 0; /* indicates successful termination */
21.
22. } /* end main */
23.
```

Функцийн загвар

Функцийн дуудалт

Функцийн жишээ...

```
24. /* Function maximum definition */
25. /* x, y and z are parameters */
26. int maximum( int x, int y, int z )
27. {
28.    int max = x;      /* assume x is largest */
29.
30.    if ( y > max ) { /* if y is larger than max, assign y to max */
31.       max = y;
32.    } /* end if */
33.
34.    if ( z > max ) { /* if z is larger than max, assign z to max */
35.       max = z;
36.    } /* end if */
37.
38.    return max;       /* max is largest value */
39.
40. } /* end function maximum */
Enter three integers: 22 85 17
Maximum is: 85

Enter three integers: 85 22 17
Maximum is: 85

Enter three integers: 22 17 85
Maximum is: 85
```

Функцийн тодорхойлолт

Анхаарах зүйл

- Функцийн загвар нь хөрвүүлэгчид дараах зүйлийг хэлж өгдөг:
  - Функцээс буцах өгөгдлийн төрөл
  - Авах параметрийн тоо, төрөл, дараалал
  - Функцийн тодорхойлолт нь програмд түүнийг ашигласнаас хойно гарч ирэх тохиолдолд л загварыг хэрэглэдэг
  - Өгөгдмөлөөр хөрвүүлэгч функцийг int төрлийн буцах утгатай гэдгээс өөр юм мэддэггүй
  - Функцийн загвар нь **`int`** –с өөр, функцийн тодорхойлолтонд буцах төрлийг орхигдуулсан бол дүрмийн алдаа болно
  - Буцах утгын төрлийг мартах нь тодорхойгүй алдаанд хүргэж болзошгүй утгыг буцаадаг. Си хэлний стандартаар энэ орхигдуулалтын үр дүнг тодорхойгүй гэсэн байдаг
  - **`void`** төрлийн буцах утгатай функцээс утга буцаах нь дүрмийн алдаа болно.
    - Функцийн загвар, толгой болон дуудалт нь параметрийн тоо, төрөл, дараалал, буцах утгын төрөлтэй тохирох ёстой

Толгой файл

- Толгой файл
  - Сангийн функцүүдийн загварыг агуулдаг
  - **<stdio.h>**, **<math.h>** г.м.
  - **#include <*filename*>** гэж ачаална

    #include <math.h>
- Толгой файл үүсгэх
  - Функцүүдтэй файл үүсгэ
  - **filename.h** гэж хадгал
  - Бусад файлд ачаалахдаа

    #include "filename.h"
  - Ингэж функцийг дахин ашиглана

Стандарт сангийн зарим толгой файл

Толгой файл	Тайлбар
`<assert.h>`	Програмыг зүгшрүүлэхэд тустай макро, мэдээлэл
`<ctype.h>`	Тэмдэгтийн шинжийг шалгах, хувиргах функц
`<errno.h>`	Алдааны нөхцлийг тайлагнах хэрэгтэй макро
`<float.h>`	Системийн хөвөгч таслалтай тооны хэмжээний хязгаарлалт
`<limits.h>`	Системийн нэгдсэн хэмжээний хязгаарлалт
`<locale.h>`	Програм ажиллах орчиндоо хэрхэн өөрчлөгдөхийг заасан функц болон мэдээлэл
`<math.h>`	Математик функц
`<setjmp.h>`	Функцийг дуудах, буцах ердийн дарааллыг алгасах функц
`<signal.h>`	Програмын ажиллагаанд гарч болох нөхцлийг удирдах функц, макро
`<stdarg.h>`	Функцийн аргументийн тоо, төрөл тодорхой бус тохиолдолтой холбоотой макро
`<stddef.h>`	С хэлэнд тодорхой тооцоолол гүйцэтгэхэд ашигладаг нийтлэг өгөгдлийн төрөл
`<stdio.h>`	Оролт/Гаралтын функц, мэдээлэл
`<stdlib.h>`	Тоо – текстийн хөрвүүлэлт, ойн хуваарилалт, санамсаргүй тоо зэрэг функц
`<string.h>`	Текст мөрийг боловсруулах функц
`<time.h>`	Цаг хугацаатай холбоотой функц

Лекцийн агуулга

- Функц гэх жижиг модуль – блокоор програм бүтээх
- Си стандарт сан дахь зарим математик функцүүд
- Шинээр функц яаж бүтээх вэ?
- Функцүүдийн хооронд мэдээлэл дамжуулах механизм
- Рекурсив (өөрийгөө дууддаг) функцийг яаж бичих вэ?
- Дүгнэлт

Функцийг утгаар болон хаягаар дуудах

- Утгаар дуудах
  - Аргументын хуулбарыг функц рүү дамжуулдаг
  - Функц доторх өөрчлөлт анхдагчид нөлөөлөхгүй
  - Функц аргументыг өөрчлөх шаардлагагүй үед ашигла
    - Санамсаргүй өөрчлөлтөөс сэргийлнэ
- Хаягаар дуудах
  - Анхдагч аргументыг өөрийг нь дамжуулдаг
  - Функц доторх өөрчлөлт анхдагчийг өөрчилнө
  - Зөв гэдэгт “итгэлтэй” функцэд л зөвхөн ашигла
- Одоохондоо утгаар дуудахад анхаарлаа хандуулья!

Жишээ: Санамсаргүй тоо гаргах

- **`rand`** функц
  - **`<stdlib.h>`** -г ачаалах
  - **`0 – RAND_MAX`**(доод тал нь 32767) –н хооронд “санамсаргүй” тоо буцаана
  - Псевдо санамсаргүй
    - “Санамсаргүй” тооны тогтсон цуваа
    - Функцийг дуудах бүрт ижил цуваа
- Масштаб
  - [1, n] хооронд санамсаргүй тоо гаргах

    `1 + ( rand() % n )`
    - **`( rand() % n )`** нь [1, n-1] хооронд утга өгнө
    - Жишээ: **`1 + ( rand() % 6 )`**

```c
/* Ex_19 Shifted, scaled integers produced by 1 + rand() % 6 */
#include <stdio.h>
#include <stdlib.h>

/* function main begins program execution */
int main( void )
{
   int i; /* counter */

   /* loop 20 times */
   for ( i = 1; i <= 20; i++ ) {

      /* pick random number from 1 to 6 and output it */
      printf( "%10d", 1 + ( rand() % 6 ) );

      /* if counter is divisible by 5, begin new line of output */
      if ( i % 5 == 0 ) {
         printf( "\n" );
      } /* end if */

   } /* end for */

   return 0; /* indicates successful termination */

} /* end main */
```

1 – 6 хооронд санамсаргүй тоо гаргаж байна

```
     6          6          5          5          6
     5          1          1          5          3
     6          6          2          4          2
     6          2          3          4          1
```

```c
/* Ex_20 Roll a six-sided die 6000 times */
#include <stdio.h>
#include <stdlib.h>

/* function main begins program execution */
int main( void )
{
   int frequency1 = 0; /* rolled 1 counter */
   int frequency2 = 0; /* rolled 2 counter */
   int frequency3 = 0; /* rolled 3 counter */
   int frequency4 = 0; /* rolled 4 counter */
   int frequency5 = 0; /* rolled 5 counter */
   int frequency6 = 0; /* rolled 6 counter */

   int roll; /* roll counter, value 1 to 6000 */
   int face; /* represents one roll of the die, value 1 to 6 */

   /* loop 6000 times and summarize results */
   for ( roll = 1; roll <= 6000; roll++ ) {
      face = 1 + rand() % 6; /* random number from 1 to 6 */

      /* determine face value and increment appropriate counter */
      switch ( face ) {

         case 1: /* rolled 1 */
            ++frequency1;
            break;
```

Функцийг утгаар дуудах жишээ

```
28.
29.          case 2: /* rolled 2 */
30.             ++frequency2;
31.             break;
32.
33.          case 3: /* rolled 3 */
34.             ++frequency3;
35.             break;
36.
37.          case 4: /* rolled 4 */
38.             ++frequency4;
39.             break;
40.
41.          case 5: /* rolled 5 */
42.             ++frequency5;
43.             break;
44.
45.          case 6: /* rolled 6 */
46.             ++frequency6;
47.             break; /* optional */
48.       } /* end switch */
49.
50.    } /* end for */
51.
```

Функцийг утгаар дуудах жишээ

```
52. /* display results in tabular format */
53.    printf( "%s%13s\n", "Face", "Frequency" );
54.    printf( "   1%13d\n", frequency1 );
55.    printf( "   2%13d\n", frequency2 );
56.    printf( "   3%13d\n", frequency3 );
57.    printf( "   4%13d\n", frequency4 );
58.    printf( "   5%13d\n", frequency5 );
59.    printf( "   6%13d\n", frequency6 );
60.
61.    return 0; /* indicates successful termination */
62.
63. } /* end main */
Face    Frequency
   1         1003
   2         1017
   3          983
   4          994
   5         1004
   6          999
```

Жишээ: Санамсаргүй тоо гаргах

- **`srand`** функц
  - **`<stdlib.h>`** -г ачаалах
  - Бүхэл утга авч түүний “санамсаргүй” тоон цуваан дахь байршилд үсэрч очно

```
srand( seed );
```

  - `srand( time(NULL) ); /* load <time.h> */`
    - `time(null);`
      - Time() функц нь хугацааг 1970 оны 1 сарын 1 –ны 0 цаг, 0 минут, 0 секундээс эхлэн тоолсон секундээр илэрхийлдэг
      - Эхлэлийг “санамсаргүй” болгох нэг арга

Функцийг утгаар дуудах жишээ

```
1.  /* Ex_21 Randomizing die-rolling program */
2.  #include <stdlib.h>
3.  #include <stdio.h>
4.
5.  /* function main begins program execution */
6.  int main( void )
7.  {
8.     int i;           /* counter */
9.     unsigned seed; /* number used to seed random number generator */
10.
11.    printf( "Enter seed: " );
12.    scanf( "%u", &seed ); /* note %u for unsigned */
13.
14.    srand( seed ); /* seed random number generator */
15.
16.    /* loop 10 times */
17.    for ( i = 1; i <= 10; i++ ) {
18.
```

Эхлэлийг санамсаргүй болгож байна

Функцийг утгаар дуудах жишээ

```
19.         /* pick a random number from 1 to 6 and output it */
20.         printf( "%10d", 1 + ( rand() % 6 ) );
21.
22.         /* if counter is divisible by 5, begin a new line of output */
23.         if ( i % 5 == 0 ) {
24.            printf( "\n" );
25.         } /* end if */
26.
27.      } /* end for */
28.
29.      return 0; /* indicates successful termination */
30.
31. } /* end main */
Enter seed: 67
         6         1         4         6         2
         1         6         1         6         4

Enter seed: 867
         2         4         6         1         6
         1         1         3         6         2

Enter seed: 67
         6         1         4         6         2
         1         6         1         6         4
```

Хадгалалтын төрөл

- Хадгалалтын төрлийг тодорхойлогч
  - Амьдрах хугацаа – обьект ойд хир удаан орших
  - Муж – обьект програмын аль хэсэгт харагдах
  - Холболт – идентификаторын ашиглагдах файлыг заах (дараа дэлгэрүүлнэ)
- Автомат хадгалалт
  - Обьект блок дотроо үүсч, устана
  - **`auto`**: локаль хувьсагчийн хувьд өгөгдмөл

    ```
    auto double x, y;
    ```
  - **`register`**: хувьсагчийг өндөр хурдтай регистрт хадгалахыг хичээдэг (зөвхөн автомат хувьсагч л ашигладаг)

    ```
    register int counter = 1;
    ```

Хадгалалтын төрөл

- Статик хадгалалт
  - Хувьсагч нь програмын ажиллагааны туршид оршино
  - Өгөгдмөл утга нь 0 (тэг)
  - **`static`**: функц дотор тодорхойлогдсон локаль хувьсагч. Функц төгссөн ч утга нь хадгалагдана. Гэхдээ зөвхөн өөрийн функцдээ л харагдана
  - **`extern`**: глобаль хувьсагч болон функцийн хувьд өгөгдмөл. Ямар ч функцэд хамаарна

Мужийг тодорхойлох дүрэм

- 4 төрлийн муж байдаг
- Файлын муж
  - Функцийн гадна зарлагдсан идентификаторын муж бүх функцийг хамрана
  - `global` хувьсагч, функцийн тодорхойлолт болон функцийн загварт ашигладаг
- Функцийн муж
  - Функцийн их бие дотроос л хандана
  - Зөвхөн шошгод л ашигладаг (`start:`, `case:` г.м.)

Мужийн дүрэм

- Блокын муж
  - Блок дотор зарлагдсан идентификатор. Хамрах муж нь зарлалтаар эхэлж, баруун ‘}’ хаалтаар төгсөнө
  - Хувьсагч болон функцийн параметрт ашигладаг (функцийн локаль хувьсагч)
  - Хэрвээ дотоод блокт ижил нэртэй хувьсагч байвал гадаад блок нь “далдлагдана”
- Функцийн загварын муж
  - Параметрын жагсаалт дахь идентификаторт ашигладаг

Мужийн жишээ

```
1.  /* Ex_22 A scoping example */
2.  #include <stdio.h>
3.
4.  void useLocal( void );         /* function prototype */
5.  void useStaticLocal( void ); /* function prototype */
6.  void useGlobal( void );       /* function prototype */
7.
8.  int x = 1; /* global variable */
9.
10. /* function main begins program execution */
11. int main( void )
12. {
13.    int x = 5; /* local variable to main */
14.
15.    printf("local x in outer scope of main is %d\n", x );
16.
17.    { /* start new scope */
18.       int x = 7; /* local variable to new scope */
19.
20.       printf( "local x in inner scope of main is %d\n", x );
21.    } /* end new scope */
22.
```

Файлын мужтай глобаль хувьсагч

Блокын мужтай хувьсагч

Блокын мужтай хувьсагч

Мужийн жишээ...

```
23.    printf( "local x in outer scope of main is %d\n", x );
24.
25.    useLocal();      /* useLocal has automatic local x */
26.    useStaticLocal(); /* useStaticLocal has static local x */
27.    useGlobal();      /* useGlobal uses global x */
28.    useLocal();       /* useLocal reinitializes automatic local x */
29.    useStaticLocal(); /* static local x retains its prior value */
30.    useGlobal();      /* global x also retains its value */
31.
32.    printf( "\nlocal x in main is %d\n", x );
33.
34.    return 0; /* indicates successful termination */
35.
36. } /* end main */
37.
38. /* useLocal reinitializes local variable x during each call */
39. void useLocal( void )
40. {
41.    int x = 25;  /* initialized each time useLocal is called */
42.
43.    printf( "\nlocal x in useLocal is %d after entering useLocal\n", x );
44.    x++;
45.    printf( "local x in useLocal is %d before exiting useLocal\n", x );
46. } /* end function useLocal */
```

Блокын мужтай хувьсагч

Мужийн жишээ...

```
47.
48. /* useStaticLocal initializes static local variable x only the first time
49.    the function is called; value of x is saved between calls to this
50.    function */
51. void useStaticLocal( void )
52. {
53.    /* initialized only first time useStaticLocal is called */
54.    static int x = 50;
55.
56.    printf( "\nlocal static x is %d on entering useStaticLocal\n", x );
57.    x++;
58.    printf( "local static x is %d on exiting useStaticLocal\n", x );
59. } /* end function useStaticLocal */
60.
61. /* function useGlobal modifies global variable x during each call */
62. void useGlobal( void )
63. {
64.    printf( "\nglobal x is %d on entering useGlobal\n", x );
65.    x *= 10;
66.    printf( "global x is %d on exiting useGlobal\n", x );
67. } /* end function useGlobal */
```

Блокын мужтай статик хувьсагч

Глобаль хувьсагч

Мужийн жишээ...

```
local x in outer scope of main is 5
local x in inner scope of main is 7
local x in outer scope of main is 5

local x in useLocal is 25 after entering useLocal
local x in useLocal is 26 before exiting useLocal

local static x is 50 on entering useStaticLocal
local static x is 51 on exiting useStaticLocal

global x is 1 on entering useGlobal
global x is 10 on exiting useGlobal

local x in useLocal is 25 after entering useLocal
local x in useLocal is 26 before exiting useLocal

local static x is 51 on entering useStaticLocal
local static x is 52 on exiting useStaticLocal

global x is 10 on entering useGlobal
global x is 100 on exiting useGlobal

local x in main is 5
```

Лекцийн агуулга

- Функц гэх жижиг модуль – блокоор програм бүтээх
- Си стандарт сан дахь зарим математик функцүүд
- Шинээр функц яаж бүтээх вэ?
- Функцүүдийн хооронд мэдээлэл дамжуулах механизм
- Рекурсив (өөрийгөө дууддаг) функцийг яаж бичих вэ?
- Дүгнэлт

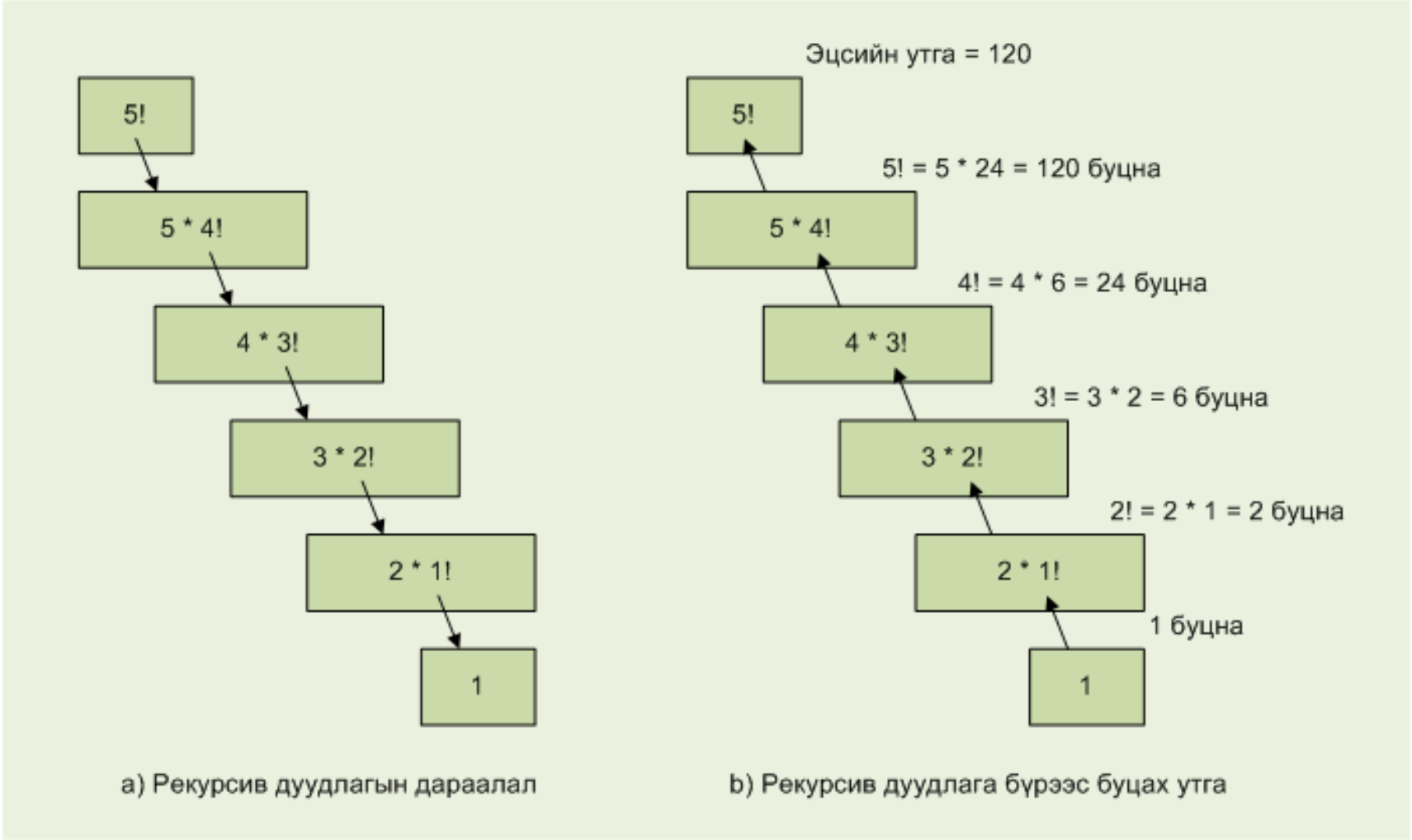
Рекурс

- Рекурсив функц
  - Өөрийгөө дууддаг функц
  - Үндсэн/суурь тохиолдлыг л шийддэг
  - Асуудлыг дараах байдлаар хуваадаг
    - Чадах зүйл
    - Чадахгүй зүйл
      - Анхдагч асуудалтай төсөөтэйг чадахгүй
      - Чадахгүй зүйлээ шийдэхийн тулд функц өөрийн хуулбарыг ажиллуулдаг
  - Эцсийн эцэст суурь тохиолдлыг шийддэг
    - Тэгээд дээшлэх замаар бүх асуудлыг шийддэг

Рекурс

- Жишээ: факториал
  - `5! = 5 * 4 * 3 * 2 * 1`
  - Анхаарч харвал
    - `5! = 5 * 4!`
    - `4! = 4 * 3!` ...
    - Факториалыг рекурсив байдлаар бодож болно
    - Суурь тохиолдлыг шийдэх (`1! = 0! = 1`), дараа нь
      - `2! = 2 * 1! = 2 * 1 = 2;`
      - `3! = 3 * 2! = 3 * 2 = 6;`

5! рекурсив тооцоолол



10! рекурсив тооцоолол

```
1.  /* Ex_23 Recursive factorial function */
2.  #include <stdio.h>
3.
4.  long factorial( long number ); /* function prototype */
5.
6.  /* function main begins program execution */
7.  int main( void )
8.  {
9.     int i; /* counter */
10.
11.    /* loop 11 times; during each iteration, calculate
12.       factorial( i ) and display result */
13.    for ( i = 0; i <= 10; i++ ) {
14.       printf( "%2d! = %ld\n", i, factorial( i ) );
15.    } /* end for */
16.
17.    return 0; /* indicates successful termination */
18.
19. } /* end main */
20.
```

10! рекурсив тооцоолол...

```
21. /* recursive definition of function factorial */
22. long factorial( long number )
23. {
24.    /* base case */
25.    if ( number <= 1 ) {
26.       return 1;
27.    } /* end if */
28.    else { /* recursive step */
29.       return ( number * factorial( number - 1 ) );
30.    } /* end else */
31.
32. } /* end function factorial */
 0! = 1
 1! = 1
 2! = 2
 3! = 6
 4! = 24
 5! = 120
 6! = 720
 7! = 5040
 8! = 40320
 9! = 362880
10! = 3628800
```

Рекурсын жишээ: Фибоначийн цуваа

- Фибоначийн цуваа: 0, 1, 1, 2, 3, 5, 8…
  - Тоо бүр өмнөх хоёр тооны нийлбэр
  - Рекурсив шийдэж болно:
    - `fib( n ) = fib( n - 1 ) + fib( n - 2 )`
    - **`fibonacci`** функцийн код

```
long fibonacci( long n )
{
  if (n == 0 || n == 1 ) /* base case */
    return n;
  else
    return fibonacci(n-1)+fibonacci(n-2);
}
```

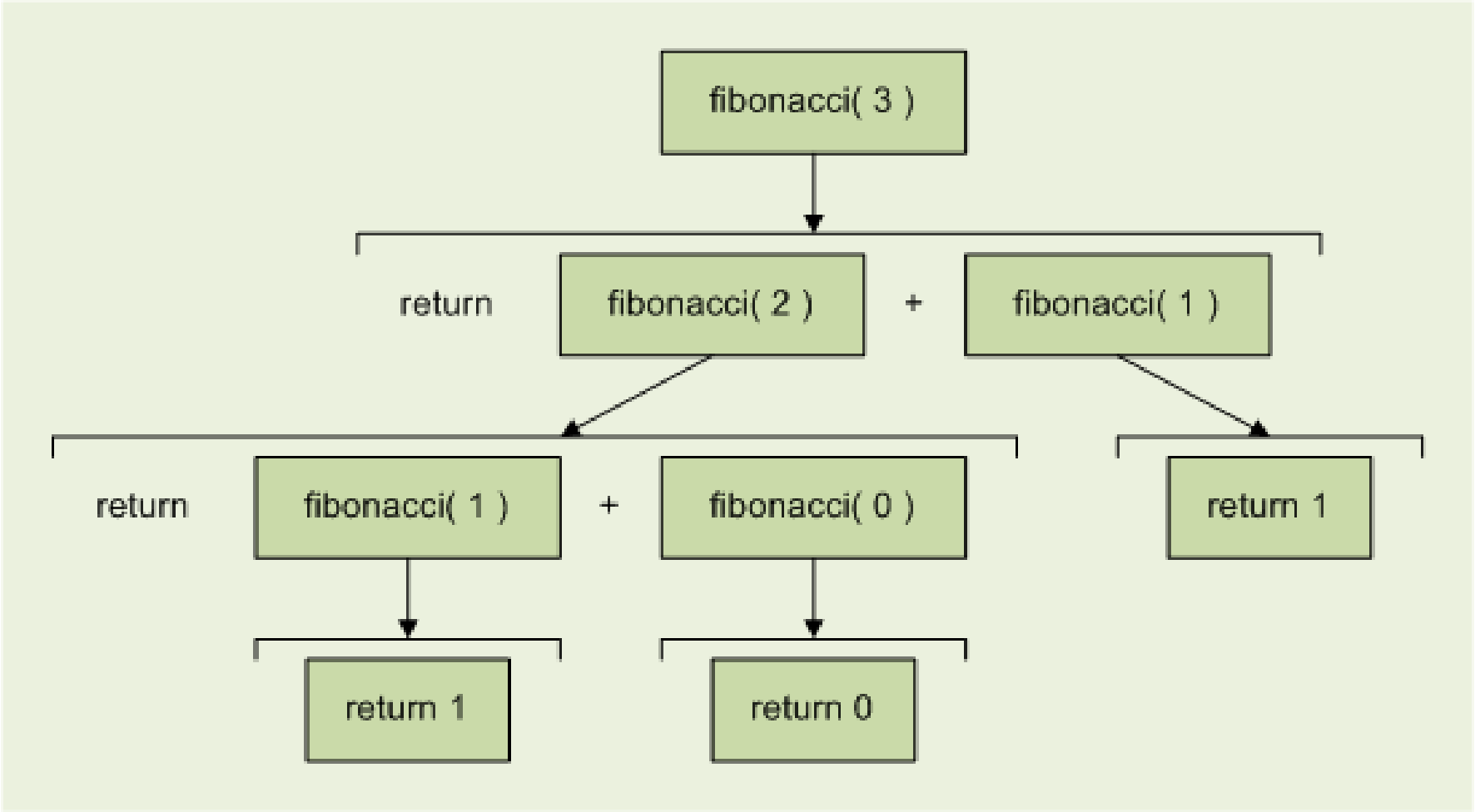
Фибоначийн цуваа

```
1.  /* Ex_24 Recursive fibonacci function */
2.  #include <stdio.h>
3.
4.  long fibonacci( long n ); /* function prototype */
5.
6.  /* function main begins program execution */
7.  int main( void )
8.  {
9.     long result; /* fibonacci value */
10.    long number; /* number input by user */
11.
12.    /* obtain integer from user */
13.    printf( "Enter an integer: " );
14.    scanf( "%ld", &number );
15.
16.    /* calculate fibonacci value for number input by user */
17.    result = fibonacci( number );
18.
19.    /* display result */
20.    printf( "Fibonacci( %ld ) = %ld\n", number, result );
21.
22.    return 0; /* indicates successful termination */
23.
24. } /* end main */
25.
```

Фибоначийн цуваа...

```
26. /* Recursive definition of function fibonacci */
27. long fibonacci( long n )
28. {
29.    /* base case */
30.    if ( n == 0 || n == 1 ) {
31.       return n;
32.    } /* end if */
33.    else { /* recursive step */
34.       return fibonacci( n - 1 ) + fibonacci( n - 2 );
35.    } /* end else */
36.
37. } /* end function fibonacci */
Enter an integer: 0
Fibonacci( 0 ) = 0

Enter an integer: 1
Fibonacci( 1 ) = 1

Enter an integer: 2
Fibonacci( 2 ) = 1

Enter an integer: 3
Fibonacci( 3 ) = 2

Enter an integer: 10
Fibonacci( 10 ) = 55
```

fibonacci(3) рекурсив дуудалтууд



Рекурс ба Алхам

- Давталт
  - Алхам: тодорхой давталт
  - Рекурс: функцийн давтсан дуудалт
- Төгсгөл
  - Алхам: давталтын нөхцөл унах
  - Рекурс: суурь тохиолдлыг таних
- Аль алин нь төгсгөлгүй байж болно
- Тэнцвэржилт
  - Гүйцэтгэлийн хурд(алхам) болон Програм хангамжийн сайн инжинерчилэл(рекурс) –ээс сонгох хэрэгтэй

Лекцийн агуулга

- Функц гэх жижиг модуль – блокоор програм бүтээх
- Си стандарт сан дахь зарим математик функцүүд
- Шинээр функц яаж бүтээх вэ?
- Функцүүдийн хооронд мэдээлэл дамжуулах механизм
- Рекурсив (өөрийгөө дууддаг) функцийг яаж бичих вэ?
- Дүгнэлт

Дүгнэлт

- Функц бол Си програмын үндсэн модуль
- Си хэлэнд **`math`** зэрэг функцийн олон тооны стандарт сан бий
- Функцийг тодорхойлно
- Функцийг ашиглахаас өмнө түүний тодорхойлолт эсхүл загвар бичигдсэн байна
- Загвар болон бусад мэдээллийг толгой файлд заадаг
- Функцийг утгаар болон хаягаар дууддаг
- Санамсаргүй тоог гарган авч болно
- Хадгалалтын хэд хэдэн төрөл бий
- Обьектууд програмд өөрийн амьдрах мужтай
- Асуудлыг рекурсив аргаар шийдэж болно

Дүгнэлт

- Удирдлагын бүтцийн дараалсан гүйцэтгэлийг өөрчлөхөд **`break`**, **`continue`** операторыг ашигладаг
- Олон сонголтыг **`switch`** операторoор хийж болно
- **`do`**...**`while`** бол давталтын өөр нэг оператор
- Логик **`&&`**(AND), **`||`**(OR), **`!`**(NOT) үйлдлүүд нийлмэл нөхцлийг үүсгэдэг
- Тэнцэтгэлийн **`==`**, олгох **`=`** үйлдлүүдийг будилж хэрэглэснээс ноцтой, илрүүлэхэд төвөгтэй алдаа гаргаж болзошгүй