CS200 – Програмчлалын үндэс

Лекц 06

Заагч

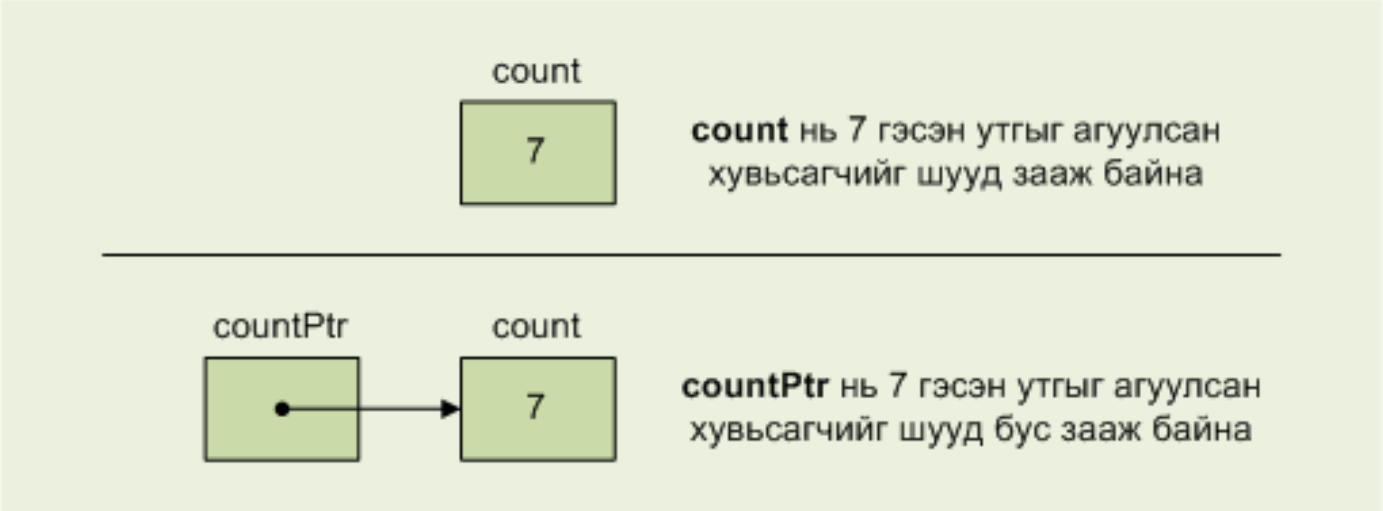
Профессор А.Эрдэнэбаатар

Лекцийн агуулга

- Заагч хувьсагчийг тодорхойлох ба идэвхижүүлэх
- Заагчийн үйлдэл
- Аргументыг функцэд хаягаар дамжуулах
- `const` тодорхойлогчийг заагчид хэрэглэх
- Хаягаар дуудах арга ашигласан бөмбөлөгөн эрэмбэлэлт
- `sizeof` үйлдэл
- Заагчийн илэрхийлэл ба заагчийн арифметик
- Заагч болон массивын уялдаа
- Заагчийн массив
- Функцийн заагч

Заагч хувьсагчийг тодорхойлох ба идэвхижүүлэх

- Хувьсагчийн утга нь санах ойн хаяг байна
- Энгийн хувьсагч бол тодорхой утгыг хадгалдаг (шууд заах)
- Заагч нь тодорхой утгатай хувьсагчийн хаягийг хадгалдагдаг (шууд бус заах)



Заагч хувьсагчийг тодорхойлох ба идэвхижүүлэх

- * -г заагч хувьсагчийн өмнө бичдэг

```
int *myPtr;
```

- `int` төрлийн заагчийг тодорхойлж/зарлаж байна
- Олон заагчийг тодорхойлохдоо * -г хувьсагч бүрийн өмнө бичнэ

```
int *myPtr1, *myPtr2;
```

- Өгөгдлийн аль ч төрөлд заагчийг тодорхойлж болно
- Заагчийн идэвхижүүлэх утга нь 0, NULL эсхүл хаяг байна
  - 0 эсхүл `NULL` – юу ч бишийг заана (0 бол заагчид олгож болох цорын ганц утга)
  - 0 –оор идэвхижүүлэх нь `NULL` –тай адилхан. Гэхдээ NULL нь илүү тохиромжтой
  - `NULL` нь <`stddef.h`> дотор тодорхойлогдсон тэмдэгт тогтмол

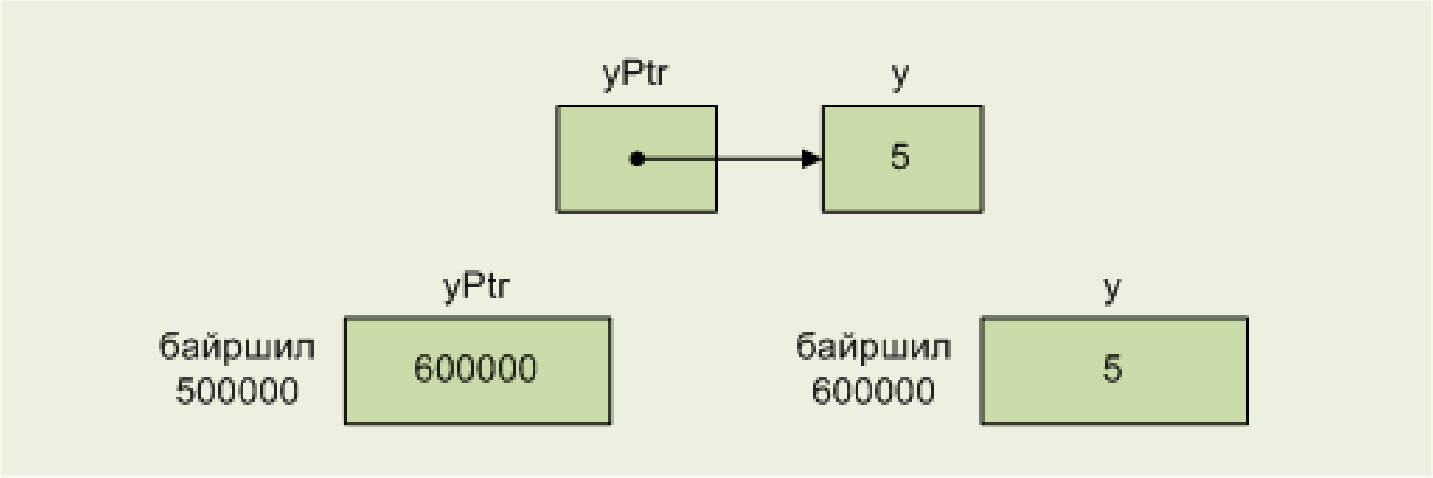
Лекцийн агуулга

- Заагч хувьсагчийг тодорхойлох ба идэвхижүүлэх
- Заагчийн үйлдэл
- Аргументыг функцэд хаягаар дамжуулах
- `const` тодорхойлогчийг заагчид хэрэглэх
- Хаягаар-дуудахыг ашигласан бөмбөлөгөн эрэмбэлэлт
- `sizeof` үйлдэл
- Заагчийн илэрхийлэл ба заагчийн арифметик
- Заагч болон массивын уялдаа
- Заагчийн массив
- Функцийн заагч

Заагчийн үйлдэл

- **&** (хаягийн үйлдэл)
  - Үйлдлийн гишүүний хаягийг буцаана

```
int y = 5;
int *yPtr;
yPtr = &y;   /* yPtr gets address of y */
             /* yPtr "points to" y */
```



Заагчийн үйлдэл

- * (шууд бус хандалт)
  - Үйлдлийн гишүүний заасныг (хувьсагчийг) буцаана
  - *yPtr нь y –г буцаана (учир нь yPtr бол y –н заагч)
  - * үйлдлийг утга олгоход ашиглаж болно

```
*yPtr = 7;  /* changes y to 7 */
```

  - Шууд бус хандалтын заагч нь (* үйлдлийн гишүүн) нь тогтмол биш **lvalue** байна
- * болон & нь эсрэг үйлдлүүд юм. Өөрөөр хэлбэл нэг нь нөгөөгийн гаргалгааг үгүйсгэдэг

Заагчийн үйлдэл

```
1.  /* Ex_42 Using the & and * operators */
2.  #include <stdio.h>
3.
4.  int main( void )
5.  {
6.     int a;          /* a is an integer */
7.     int *aPtr;      /* aPtr is a pointer to an integer */
8.
9.     a = 7;
10.    aPtr = &a;      /* aPtr set to address of a */
11.
12.    printf( "The address of a is %p"
13.            "\nThe value of aPtr is %p", &a, aPtr );
14.
15.    printf( "\n\nThe value of a is %d"
16.            "\nThe value of *aPtr is %d", a, *aPtr );
17.
18.    printf( "\n\nShowing that * and & are complements of "
19.            "each other\n&*aPtr = %p"
20.            "\n*&aPtr = %p\n", &*aPtr, *&aPtr );
21.
22.    return 0; /* indicates successful termination */
23.
24. } /* end main */
```

Хэрвээ **aPtr** нь a-н заагч бол **&a** болон **aPtr** нь ижил утгатай

a болон ***aPtr** нь ижил утгатай

&a болон ***&aPtr** нь ижил утгатай

Заагчийн үйлдэл

```
The address of a is 0012FF7C
The value of aPtr is 0012FF7C

The value of a is 7
The value of *aptr is 7

Showing that * and & are complements of each other
&*aPtr = 0012FF7C
*&aPtr = 0012FF7C
```

Үйлдлүүдийн ахлах чанар

Үйлдэл	Биелэгдэх чиглэл	Төрөл
`[] ()`	Зүүнээс баруун	Хамгийн ахлах
`+ - ++ -- ! * &` (төрөл)	Баруунаас зүүн	Ганц гишүүнт
`* / %`	Зүүнээс баруун	Үржигдэх
`+ -`	Зүүнээс баруун	Нэмэгдэх
`< <= > >=`	Зүүнээс баруун	Харьцаа
`== !=`	Зүүнээс баруун	Тэнцэтгэл
`&&`	Зүүнээс баруун	Логик AND
`||`	Зүүнээс баруун	Логик OR
`?:`	Баруунаас зүүн	Нөхцөлт
`= += -+ *= /= %=`	Баруунаас зүүн	Олгох
`,`	Зүүнээс баруун	Таслал

Лекцийн агуулга

- Заагч хувьсагчийг тодорхойлох ба идэвхижүүлэх
- Заагчийн үйлдэл
- Аргументыг функцэд хаягаар дамжуулах
- `const` тодорхойлогчийг заагчид хэрэглэх
- Хаягаар-дуудахыг ашигласан бөмбөлөгөн эрэмбэлэлт
- `sizeof` үйлдэл
- Заагчийн илэрхийлэл ба заагчийн арифметик
- Заагч болон массивын уялдаа
- Заагчийн массив
- Функцийн заагч

Аргументыг функцэд хаягаар дамжуулах

- Заагч аргумент ашиглаж хаягаар дуудах
  - Аргутентийн хаягийг **&** үйлдэл ашиглаж дамжуулна
  - Санах ой дахь бодит байршлыг өөрчлөх боломж олгодог
  - Массивыг дамжуулахад **&** хэрэггүй, учир нь түүний нэр өөрөө заагч
- ***** үйлдэл
  - Функцийн дотор орлосон нэрээр хувьсагчийг ашигладаг

```
void timestwo( int *number)
{
   *number = 2 * ( *number )
}
```

  - ***number** нь дамжуулсан хувьсагчийн орлосон нэр

Утгаар дуудах

```
1.  /* Ex_43 Cube a variable using call-by-value */
2.  #include <stdio.h>
3.
4.  int cubeByValue( int n ); /* prototype */
5.
6.  int main( void )
7.  {
8.     int number = 5; /* initialize number */
9.
10.    printf( "The original value of number is %d", number );
11.
12.    /* pass number by value to cubeByValue */
13.    number = cubeByValue( number );
14.
15.    printf( "\nThe new value of number is %d\n", number );
16.
17.    return 0; /* indicates successful termination */
18.
19. } /* end main */
20.
21. /* calculate and return cube of integer argument */
22. int cubeByValue( int n )
23. {
24.    return n * n * n;    /* cube local variable n and return result */
25.
26. } /* end function cubeByValue */
The original value of number is 5
The new value of number is 125
```

Заагч аргументтэй хаягаар дуудах

```
1.  /* Ex_44 Cube a variable using call-by-reference with a pointer argument */
2.
3.  #include <stdio.h>
4.
5.  void cubeByReference( int *nPtr ); /* prototype */
6.
7.  int main( void )
8.  {
9.     int number = 5; /* initialize number */
10.
11.    printf( "The original value of number is %d", number );
12.
13.    /* pass address of number to cubeByReference */
14.    cubeByReference( &number );
15.
16.    printf( "\nThe new value of number is %d\n", number );
17.
18.    return 0; /* indicates successful termination */
19.
20. } /* end main */
21.
22. /* calculate cube of *nPtr; modifies variable number in main */
23. void cubeByReference( int *nPtr )
24. {
25.    *nPtr = *nPtr * *nPtr * *nPtr;  /* cube *nPtr */
26. } /* end function cubeByReference */
The original value of number is 5
The new value of number is 125
```

Функцийн загвар заагч аргументтэй байна

Функц cubeByReference –д хаяг дамжуулж байна. Энэ нь заач хувьсагчийн утга байж болно

Энэ функцэд *nPtr бол тоо, иймд энэ оператор өөрөө тооны утгыг өөрчилж байна

Шинжилгээ: Утгаар дуудах

Алхам 1: main функц cubeByValue –г дуудахын өмнө

```
int main( void ){
  int number = 5;
  number = cubeByValue( number );
}
```

number: 5

```
int cubeByValue( int n )
{
  return n * n * n;
}
```

n: тодорхойгүй

Алхам 2: cubeByValue дуудлага авсны дараа

```
int main( void ){
  int number = 5;
  number = cubeByValue( number );
}
```

number: 5

```
int cubeByValue( int n )
{
  return n * n * n;
}
```

n: 5

Алхам 3: cubeByValue функц параметрын кубыг бодосны дараа, main функц рүү буцахын өмнө

```
int main( void ){
  int number = 5;
  number = cubeByValue( number );
}
```

number: 5

```
int cubeByValue( int n )
{
  return n * n * n;   // 125
}
```

n: 5

Алхам 4: cubeByValue функц main функц рүү буцсаны дараа, number –д утга олгохын өмнө

```
int main( void ){
  int number = 5;
  number = cubeByValue( number );   // 125
}
```

number: 5

```
int cubeByValue( int n )
{
  return n * n * n;
}
```

n: тодорхойгүй

Алхам 5: main функц number –д утга олгосны дараа

```
int main( void ){
  int number = 5;
  number = cubeByValue( number );   // 125
}
```

number: 125

```
int cubeByValue( int n )
{
  return n * n * n;
}
```

n: тодорхойгүй

Шинжилгээ: Хаягаар дуудах

Алхам 1: main функц cubeByReference –г дуудахын өмнө

```
int main( void )
{
  int number = 5;
  cubeByReference( &number );
}
```

number
5

```
int cubeByReference( int *nPtr )
{
  *nPtr = *nPtr * *nPtr * *nPtr;
}
```

nPtr
тодорхойгүй

Алхам 2: cubeByReference дуудлага авсны дараа, *nPtr –н кубыг бодохын өмнө

```
int main( void )
{
  int number = 5;
  cubeByReference( &number );
}
```

number
5

```
int cubeByReference( int *nPtr )
{
  *nPtr = *nPtr * *nPtr * *nPtr;
}
```

Дуудлага энэ заагчийг тогтооно

nPtr

Алхам 3: *nPtr –н кубыг бодосны дараа, main функц рүү буцахын өмнө

```
int main( void )
{
  int number = 5;
  cubeByReference( &number );
}
```

number
125

```
int cubeByReference( int *nPtr )
{                    125
  *nPtr = *nPtr * *nPtr * *nPtr;
}
```

Дуудагдсан функц дуудагчийн хувьсагчийг өөрчилнө

nPtr

Лекцийн агуулга

- Заагч хувьсагчийг тодорхойлох ба идэвхижүүлэх
- Заагчийн үйлдэл
- Аргументыг функцэд хаягаар дамжуулах
- `const` тодорхойлогчийг заагчид хэрэглэх
- Хаягаар-дуудахыг ашигласан бөмбөлөгөн эрэмбэлэлт
- `sizeof` үйлдэл
- Заагчийн илэрхийлэл ба заагчийн арифметик
- Заагч болон массивын уялдаа
- Заагчийн массив
- Функцийн заагч

const тодорхойлогчийг заагчид хэрэглэх

- `const` тодорхойлогч
  - Хувьсагчийн утга өөрчлөгдөхгүй
  - Функцэд хувьсагчийг өөрчлөх шаардлагагүй үед ашиглана
  - `const` хувьсагчийг өөрчлөх гэж оролдвол алдаа гарна
- `const` заагч
  - Санах ойн тогтмол байршлыг заана
  - Тодорхойлох үедээ идэвхижүүлэх ёстой

```
const int *myPtr = &x;
```

    - **`const int`** төрлийн энгийн заагч

```
int *const myPtr = &x;
```

    - **`int *const`** төрөл : **`int`** –н тогтмол заагч. **`x`** –г өөрчилж болно, харин **`myPtr`** –г болохгүй

```
const int *conts Ptr = &x;
```

    - **`const int`** төрлийн тогтмол заагч

Тогтмол биш өгөгдлийн тогтмол биш заагч

```
1.  /* Ex_45 Converting lowercase letters to uppercase letters
2.     using a non-constant pointer to non-constant data */
3.
4.  #include <stdio.h>
5.  #include <ctype.h>
6.
7.  void convertToUppercase( char *sPtr ); /* prototype */
8.
9.  int main( void )
10. {
11.    char string[] = "characters and $32.98"; /* initialize char array */
12.
13.    printf( "The string before conversion is: %s", string );
14.    convertToUppercase( string );
15.    printf( "\nThe string after conversion is: %s\n", string );
16.
17.    return 0; /* indicates successful termination */
18.
19. } /* end main */
20.
```

sPtr болон *sPtr –н аль алиныг өөрчилж болно

Тогтмол биш өгөгдлийн тогтмол биш заагч...

```
21. /* convert string to uppercase letters */
22. void convertToUppercase( char *sPtr )
23. {
24.    while ( *sPtr != '\0' ) { /* current character is not '\0' */
25.
26.       if ( islower( *sPtr ) ) {    /* if character is lowercase, */
27.          *sPtr = toupper( *sPtr ); /* convert to uppercase */
28.       } /* end if */
29.
30.       ++sPtr;  /* move sPtr to the next character */
31.    } /* end while */
32.
33. } /* end function convertToUppercase */
The string before conversion is: characters and $32.98
The string after conversion is: CHARACTERS AND $32.98
```

sPtr болон ***sPtr** –н аль алиныг нь **convertToUppercase** функц өөрчилж байна

Тогтмол өгөгдлийн тогтмол биш заагч

```
1.  /* Ex_46 Printing a string one character at a time using
2.     a non-constant pointer to constant data */
3.
4.  #include <stdio.h>
5.
6.  void printCharacters( const char *sPtr );
7.
8.  int main( void )
9.  {
10.    /* initialize char array */
11.    char string[] = "print characters of a string";
12.
13.    printf( "The string is:\n" );
14.    printCharacters( string );
15.    printf( "\n" );
16.
17.    return 0; /* indicates successful termination */
18.
19. } /* end main */
20.
```

Заагч хувьсагч **sPtr** –г өөрчилж болно. Харин түүний зааж байгаа өгөгдөл ***sPtr** -г өөрчилж болохгүй

Тогтмол өгөгдлийн тогтмол биш заагч...

```
21. /* sPtr cannot modify the character to which it points,
22.    i.e., sPtr is a "read-only" pointer */
23. void printCharacters( const char *sPtr )
24. {
25.    /* loop through entire string */
26.    for ( ; *sPtr != '\0'; sPtr++ ) {   /* no initialization */
27.       printf( "%c", *sPtr );
28.    } /* end for */
29.
30. } /* end function printCharacters */
The string is:
Print characters of a string
```

`sPtr` –г `printCharacters` функц өөрчилж байна

Заагчийн алдаатай хэрэглээ

```
1.  /* Ex_47 Attempting to modify data through a
2.     non-constant pointer to constant data. */
3.  #include <stdio.h>
4.
5.  void f( const int *xPtr ); /* prototype */
6.
7.  int main( void )
8.  {
9.     int y;         /* define y */
10.
11.    f( &y );       /* f attempts illegal modification */
12.
13.    return 0;      /* indicates successful termination */
14.
15. } /* end main */
16.
17. /* xPtr cannot be used to modify the
18.    value of the variable to which it points */
19. void f( const int *xPtr )
20. {
21.    *xPtr = 100;   /* error: cannot modify a const object */
22. } /* end function f */
```

Заагч хувьсагч **xPtr** –г өөрчилж болно. Харин түүний зааж байгаа өгөгдөл ***xPtr** -г өөрчилж болохгүй

***xPtr** нь const тодорхойлогчтой тул түүний утгыг өөрчлөх гэж оролдоход алдаа гарна

Хөрвүүлэлтийн алдаа!

Заагчийн алдаатай хэрэглээ

```
1.  /* Ex_48 Attempting to modify a constant pointer to non-constant data */
2.  #include <stdio.h>
3.
4.  int main( void )
5.  {
6.     int x; /* define x */
7.     int y; /* define y */
8.
9.     /* ptr is a constant pointer to an integer that can be modified
10.       through ptr, but ptr always points to the same memory location */
11.    int * const ptr = &x;
12.
13.    *ptr = 7; /* allowed: *ptr is not const */
14.    ptr = &y; /* error: ptr is const; cannot assign new address */
15.
16.    return 0; /* indicates successful termination */
17.
18. } /* end main */
```

Заагч **ptr** –г өөрчилж болохгүй. Харин түүний зааж байгаа өгөгдөл ***ptr** -г өөрчилж болно

Хөрвүүлэлтийн алдаа!

Заагчийн алдаатай хэрэглээ

```
1.  /* Ex_49 Attempting to modify a constant pointer to constant data. */
2.  #include <stdio.h>
3.
4.  int main( void )
5.  {
6.     int x = 5; /* initialize x */
7.     int y;     /* define y */
8.
9.     /* ptr is a constant pointer to a constant integer. ptr always
10.       points to the same location; the integer at that location
11.       cannot be modified */
12.    const int *const ptr = &x;
13.
14.    printf( "%d\n", *ptr );
15.
16.    *ptr = 7; /* error: *ptr is const; cannot assign new value */
17.    ptr = &y; /* error: ptr is const; cannot assign new address */
18.
19.    return 0; /* indicates successful termination */
20.
21. } /* end main */
```

Заагч **sPtr** болон түүний зааж байгаа өгөгдөл ***sPtr** –ын аль алиныг өөрчилж болохгүй

Хөрвүүлэлтийн алдаа!

Лекцийн агуулга

- Заагч хувьсагчийг тодорхойлох ба идэвхижүүлэх
- Заагчийн үйлдэл
- Аргументыг функцэд хаягаар дамжуулах
- `const` тодорхойлогчийг заагчид хэрэглэх
- Хаягаар-дуудахыг ашигласан бөмбөлөгөн эрэмбэлэлт
- `sizeof` үйлдэл
- Заагчийн илэрхийлэл ба заагчийн арифметик
- Заагч болон массивын уялдаа
- Заагчийн массив
- Функцийн заагч

Хаягаар-дуудах арга ашигласан бөмбөлөгөн эрэмбэлэлт

- Бөмбөлөгөн эрэмбэлэлтэнд заагч ашиглая
  - Хоёр элементийг солих
  - Солигч **`swap`** функц нь массивын элементүүдийн хаягийг (& ашиглан) авах ёстой. Өгөгдмөлөөр массивын элемент Утгаар-дуудагддаг.
  - **`swap`** функц нь заагч болон **`*`** үйлдэл ашиглан элементүүдийг сольж чадна
- Псевдокод

Массивыг идэвхижүүлэх

өгөгдлийг анхны дарааллаар хэвлэх

Бөмбөлөгөн эрэмбэлэлтийн функцийг дуудах

эрэмбэлэгдсэн массивыг хэвлэх

Бөмбөлөгөн эрэмбэлэлтийн функцийг тодорхойлох

Бөмбөлөгөн эрэмбэлэлт

```
1.  /* Ex_50 This program puts values into an array, sorts the values into
2.     ascending order, and prints the resulting array. */
3.  #include <stdio.h>
4.  #define SIZE 10
5.
6.  void bubbleSort( int * const array, const int size ); /* prototype */
7.
8.  int main( void )
9.  {
10.    /* initialize array a */
11.    int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
12.
13.    int i; /* counter */
14.
15.    printf( "Data items in original order\n" );
16.
17.    /* loop through array a */
18.    for ( i = 0; i < SIZE; i++ ) {
19.       printf( "%4d", a[ i ] );
20.    } /* end for */
21.
22.    bubbleSort( a, SIZE ); /* sort the array */
23.
24.    printf( "\nData items in ascending order\n" );
25.
26.    /* loop through array a */
27.    for ( i = 0; i < SIZE; i++ ) {
28.       printf( "%4d", a[ i ] );
29.    } /* end for */
```

Бөмбөлөгөн эрэмбэлэлт...

```
30.
31.    printf( "\n" );
32.
33.    return 0; /* indicates successful termination */
34.
35. } /* end main */
36.
37. /* sort an array of integers using bubble sort algorithm */
38. void bubbleSort( int * const array, const int size )
39. {
40.    void swap( int *element1Ptr, int *element2Ptr ); /* prototype */
41.    int pass; /* pass counter */
42.    int j;    /* comparison counter */
43.
44.    /* loop to control passes */
45.    for ( pass = 0; pass < size - 1; pass++ ) {
46.
47.       /* loop to control comparisons during each pass */
48.       for ( j = 0; j < size - 1; j++ ) {
49.
50.          /* swap adjacent elements if they are out of order */
51.          if ( array[ j ] > array[ j + 1 ] ) {
52.             swap( &array[ j ], &array[ j + 1 ] );
53.          } /* end if */
54.
55.       } /* end inner for */
56.
57.    } /* end outer for */
58.
59. } /* end function bubbleSort */
```

Бөмбөлөгөн эрэмбэлэлт...

```
60.
61. /* swap values at memory locations to which element1Ptr and
62.    element2Ptr point */
63. void swap( int *element1Ptr, int *element2Ptr )
64. {
65.    int hold = *element1Ptr;
66.    *element1Ptr = *element2Ptr;
67.    *element2Ptr = hold;
68. } /* end function swap */
Data items in original order
   2    6    4    8   10   12   89   68   45   37
Data items in ascending order
   2    6    4    8   10   12   37   45   68   89
```

swap функц хоёр заагчийн заасан бүхэл утгуудыг сольно

Лекцийн агуулга

- Заагч хувьсагчийг тодорхойлох ба идэвхижүүлэх
- Заагчийн үйлдэл
- Аргументыг функцэд хаягаар дамжуулах
- `const` тодорхойлогчийг заагчид хэрэглэх
- Хаягаар-дуудахыг ашигласан бөмбөлөгөн эрэмбэлэлт
- `sizeof` үйлдэл
- Заагчийн илэрхийлэл ба заагчийн арифметик
- Заагч болон массивын уялдаа
- Заагчийн массив
- Функцийн заагч

`sizeof` үйлдэл

- **`sizeof`**
  - Үйлдлийн гишүүний дүрслэлийн хэмжээг байтаар илэрхийлж буцаана
  - Массивын хувьд: 1 элементийн хэмжээ X элементийн тоо
  - Хэрэв **`sizeof( int )`** нь 4 -тэй тэнцүү бол

```
int myArray[ 10 ];
printf( "%d", sizeof( myArray ) );
```

    - 40 –г хэвлэнэ
- **`sizeof`** –г дараах зүйлд ашиглаж болно
  - Хувьсагчийн нэр
  - Төрлийн нэр
  - Тогтмол утга

sizeof үйлдэл

```
1.  /* Ex_51 Sizeof operator when used on an array name
2.     returns the number of bytes in the array. */
3.  #include <stdio.h>
4.
5.  size_t getSize( float *ptr ); /* prototype */
6.
7.  int main( void )
8.  {
9.     float array[ 20 ]; /* create array */
10.
11.    printf( "The number of bytes in the array is %d"
12.            "\nThe number of bytes returned by getSize is %d\n",
13.            sizeof( array ), getSize( array ) );
14.
15.    return 0; /* indicates successful termination */
16.
17. } /* end main */
18.
19. /* return size of ptr */
20. size_t getSize( float *ptr )
21. {
22.    return sizeof( ptr );
23.
24. } /* end function getSize */
```

`float` нь санах ойд 4 байт эзэлдэг тул 20 утга 80 байт болно

```c
/* Ex_52 Demonstrating the sizeof operator */
#include <stdio.h>

int main( void )
{
   char c;
   short s;
   int i;
   long l;
   float f;
   double d;
   long double ld;
   int array[ 20 ];  /* create array of 20 int elements */
   int *ptr = array; /* create pointer to array */

   printf( "     sizeof c = %d\tsizeof(char)  = %d"
           "\n     sizeof s = %d\tsizeof(short) = %d"
           "\n     sizeof i = %d\tsizeof(int) = %d"
           "\n     sizeof l = %d\tsizeof(long) = %d"
           "\n     sizeof f = %d\tsizeof(float) = %d"
           "\n     sizeof d = %d\tsizeof(double) = %d"
           "\n    sizeof ld = %d\tsizeof(long double) = %d"
           "\n sizeof array = %d"
           "\n   sizeof ptr = %d\n",
           sizeof c, sizeof( char ), sizeof s, sizeof( short ), sizeof i,
           sizeof( int ), sizeof l, sizeof( long ), sizeof f,
           sizeof( float ), sizeof d, sizeof( double ), sizeof ld,
           sizeof( long double ), sizeof array, sizeof ptr );

   return 0; /* indicates successful termination */

} /* end main */
```

Лекцийн агуулга

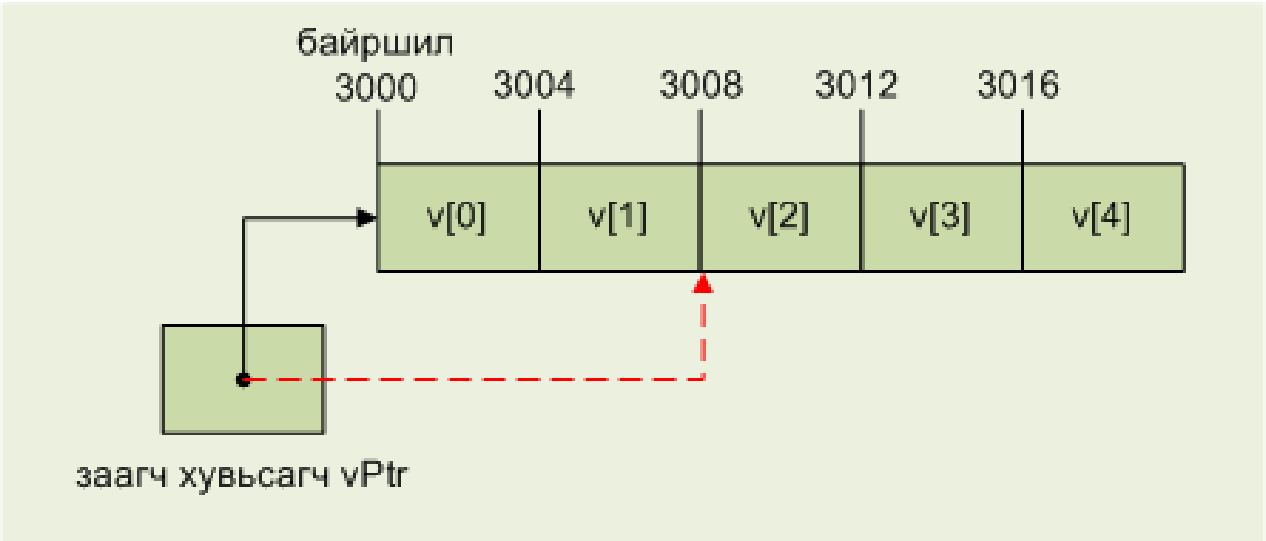
- Заагч хувьсагчийг тодорхойлох ба идэвхижүүлэх
- Заагчийн үйлдэл
- Аргументыг функцэд хаягаар дамжуулах
- `const` тодорхойлогчийг заагчид хэрэглэх
- Хаягаар-дуудахыг ашигласан бөмбөлөгөн эрэмбэлэлт
- `sizeof` үйлдэл
- Заагчийн илэрхийлэл ба заагчийн арифметик
- Заагч болон массивын уялдаа
- Заагчийн массив
- Функцийн заагч

Заагчийн илэрхийлэл ба заагчийн арифметик

- Заагч дээр арифметик үйлдэл гүйцэтгэж болно
  - Заагчийг нэмэгдүүлэх/хорогдуулах (`++ --`)
  - Заагч дээр бүхэл тоо нэмэх (`+ += - -=`)
  - Нэг заагчаас нөгөө заагчийг хасаж болно
  - Үйлдлүүдийг массив дээр л хийгээгүй бол утга муутай

Заагчийн илэрхийлэл ба заагчийн арифметик

- Бүхэл тоо 4 байт эзэлдэг машин дээр 5 элементтэй бүхэл массив байна гэж бодьё
  - **`vPtr`** нь эхний элемент болох **`v[0]`** –г заана (жишээ нь **`vPtr = 3000`**)
  - **`vPtr += 2;`** илэрхийллийн үр дүнд **`vPtr`** нь 3008 гэсэн хаягийг заах болно. (заагчийг 2 –оор нэмэгдүүлж байгаа юм шиг харагдавч, бүхэл тоо 4 байт эзэлдэг тул 3008 –г заана)



Заагчийн илэрхийлэл ба заагчийн арифметик

- Заагчаас заагч хасах
  - Нэг элементээс нөгөө элементийн хоорондох элементийн тоог буцаана Хэрвээ
    `vPtr2 = v[2];`
    `vPtr0 = v[0];` бол
  - `vPtr2 - vPtr0` илэрхийллээс 2 гэсэн үр дүнд гарна
- Заагчийн харьцуулалт (< == >)
  - Аль заагч нь массивын өндөр дугаартай элемент гэдгийг олох
  - Мөн заагч 0 –тэй тэнцүү эсэхийг шалгах

Заагчийн илэрхийлэл ба заагчийн арифметик

- Заагчид ижил төрөл бол нэгд нь нөгөөг олгож болно
  - Хэрвээ өөр төрөл бол төрлийн хувиргалт хийх ёстой
  - Онцгой тохиолдол: `void` төрлийн заагч (`void *`)
    - Ерөнхий заагч, аль ч төрлийг илэрхийлнэ
    - Заагчийг `void` төрлийн заагч болгоход төрлийн хувиргалт шаардлагагүй
    - `void` төрлийн заагчаар утга өөрчилж болохгүй

Лекцийн агуулга

- Заагч хувьсагчийг тодорхойлох ба идэвхижүүлэх
- Заагчийн үйлдэл
- Аргументыг функцэд хаягаар дамжуулах
- `const` тодорхойлогчийг заагчид хэрэглэх
- Хаягаар-дуудахыг ашигласан бөмбөлөгөн эрэмбэлэлт
- `sizeof` үйлдэл
- Заагчийн илэрхийлэл ба заагчийн арифметик
- Заагч болон массивын уялдаа
- Заагчийн массив
- Функцийн заагч

Заагч болон массивын уялдаа

- Заагч болон массив ойрхон холбоотой
  - Массивын нэр нь тогтмол заагчтай төстэй
  - Заагчаар массивын индексийн үйлдэл хийж болно
- `b[5]` массив болон `bPtr` заагчийг тодорхойльё
  - Нэгийг нь нөгөөтэй нь тэнцүү болгохын тулд:
    ```
    bPtr = b;
    ```
  - Массивын нэр нь үнэндээ түүний эхний элементийн хаяг байдаг
    ```
    bPtr = &b[0];
    ```
  - Энэ оператор `bPtr` –т `b` –н эхний элементийн хаягийг олгож байна
- Элемент `b[3]`
  - *Заагчийн шилжилт*: `*(bPtr+3)` эсхүл *Заагчийн индексжилт*: `bPtr[3]` бичлэгээр хандаж болно.
  - Массив өөрөө заагчийн арифметикт орж болно: `*(b+3)`

Заагч ба массив

```
1.  /* Ex_53 Using subscripting and pointer notations with arrays */
2.
3.  #include <stdio.h>
4.
5.  int main( void )
6.  {
7.     int b[] = { 10, 20, 30, 40 }; /* initialize array b */
8.     int *bPtr = b;                /* set bPtr to point to array b */
9.     int i;                        /* counter */
10.    int offset;                   /* counter */
11.
12.    /* output array b using array subscript notation */
13.    printf( "Array b printed with:\nArray subscript notation\n" );
14.
15.    /* loop through array b */
16.    for ( i = 0; i < 4; i++ ) {
17.       printf( "b[ %d ] = %d\n", i, b[ i ] );
18.    } /* end for */
19.
20.    /* output array b using array name and pointer/offset notation */
21.    printf( "\nPointer/offset notation where\n"
22.            "the pointer is the array name\n" );
23.
24.    /* loop through array b */
25.    for ( offset = 0; offset < 4; offset++ ) {
26.       printf( "*( b + %d ) = %d\n", offset, *( b + offset ) );
27.    } /* end for */
28.
```

Массивын индекстэй бичлэг

Заагчийн шилжилттэй бичлэг

Заагч ба массив...

```
29.    /* output array b using bPtr and array subscript notation */
30.    printf( "\nPointer subscript notation\n" );
31.
32.    /* loop through array b */
33.    for ( i = 0; i < 4; i++ ) {
34.       printf( "bPtr[ %d ] = %d\n", i, bPtr[ i ] );
35.    } /* end for */
36.
37.    /* output array b using bPtr and pointer/offset notation */
38.    printf( "\nPointer/offset notation\n" );
39.
40.    /* loop through array b */
41.    for ( offset = 0; offset < 4; offset++ ) {
42.       printf( "*( bPtr + %d ) = %d\n", offset, *( bPtr + offset ) );
43.    } /* end for */
44.
45.    return 0; /* indicates successful termination */
46.
47. } /* end main */
Array b printed with:
Array subscription notation
b[ 0 ] = 10
b[ 1 ] = 20
b[ 2 ] = 30
b[ 3 ] = 40
```

Заагчийн индекстэй бичлэг

Заагчийн шилжилттэй бичлэг

Заагч ба массив…

```
Pointer/offset notation where
the pointer is the array name
*( b + 0 ) = 10
*( b + 1 ) = 20
*( b + 2 ) = 30
*( b + 3 ) = 40

Pointer subscript notation
bPtr[ 0 ] = 10
bPtr[ 1 ] = 20
bPtr[ 2 ] = 30
bPtr[ 3 ] = 40

Pointer/offset notation
*( bPtr + 0 ) = 10
*( bPtr + 1 ) = 20
*( bPtr + 2 ) = 30
*( bPtr + 3 ) = 40
```

Заагч ба массив

```
1.  /* Ex_54 Copying a string using array notation and pointer notation. */
2.  #include <stdio.h>
3.
4.  void copy1( char *s1, const char *s2 ); /* prototype */
5.  void copy2( char *s1, const char *s2 ); /* prototype */
6.
7.  int main( void )
8.  {
9.     char string1[ 10 ];           /* create array string1 */
10.    char *string2 = "Hello";      /* create a pointer to a string */
11.    char string3[ 10 ];           /* create array string3 */
12.    char string4[] = "Good Bye"; /* create a pointer to a string */
13.
14.    copy1( string1, string2 );
15.    printf( "string1 = %s\n", string1 );
16.
17.    copy2( string3, string4 );
18.    printf( "string3 = %s\n", string3 );
19.
20.    return 0; /* indicates successful termination */
21.
22. } /* end main */
23.
```

Заагч ба массив...

```
24. /* copy s2 to s1 using array notation */
25. void copy1( char *s1, const char *s2 )
26. {
27.    int i; /* counter */
28.
29.    /* loop through strings */
30.    for ( i = 0; ( s1[ i ] = s2[ i ] ) != '\0'; i++ ) {
31.       ;    /* do nothing in body */
32.    } /* end for */
33.
34. } /* end function copy1 */
35.
36. /* copy s2 to s1 using pointer notation */
37. void copy2( char *s1, const char *s2 )
38. {
39.    /* loop through strings */
40.    for ( ; ( *s1 = *s2 ) != '\0'; s1++, s2++ ) {
41.       ;    /* do nothing in body */
42.    } /* end for */
43.
44. } /* end function copy2 */
String1 = Hello
String3 = Good Bye
```

for циклийн нөхцөлд жинхэнэ үйл явагдаж байна

Лекцийн агуулга

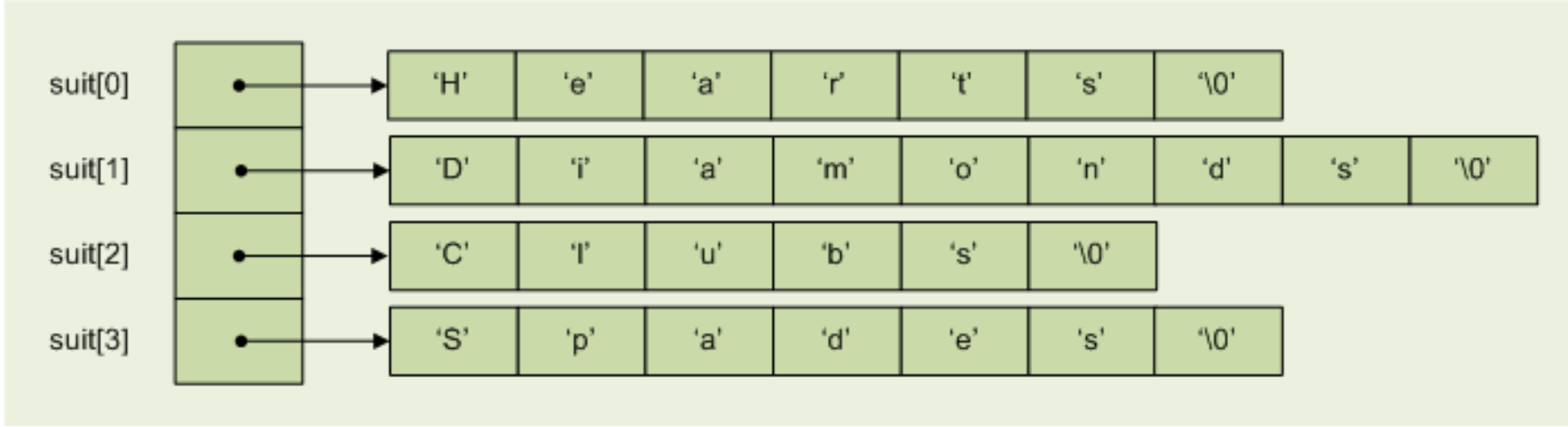
- Заагч хувьсагчийг тодорхойлох ба идэвхижүүлэх
- Заагчийн үйлдэл
- Аргументыг функцэд хаягаар дамжуулах
- `const` тодорхойлогчийг заагчид хэрэглэх
- Хаягаар-дуудахыг ашигласан бөмбөлөгөн эрэмбэлэлт
- `sizeof` үйлдэл
- Заагчийн илэрхийлэл ба заагчийн арифметик
- Заагч болон массивын уялдаа
- Заагчийн массив
- Функцийн заагч

Заагчийн массив

- Массив нь заагчуудыг агуулж болно
- Жишээ: мөрүүдийн массив

```
char *suit[4] = {"Hearts", "Diamonds", "Clubs",
  "Spades"};
```

  - Мөр нь эхний тэмдэгтийг заадаг
  - **`Char *`** - массив **`suit`** –ийн элемент бүр тэмдэгтийн заагч
  - Мөр өөрөө **`suit`** массивт хадгалагдахгүй, харин түүний заагч хадгалагдана



  - Массив **`suit`** тогтсон хэмжээтэй бол, мөрүүд өөр өөр хэмжээтэй байж болно

Заагчийн массив: Ром тооны эквивалент

```
1.  /* Ex_55 roman numeral conversion with array of pointers */
2.  #include <stdio.h>
3.
4.  void main( void )
5.  {
6.    int decimal_number = 101, a = 0, b = 0;
7.    const char *x[11] = {"", "x", "xx", "xxx", "xl", "l", "lx", "lxx",
8.    "lxxx", "xc", "c"};
9.    const char *y[10] = {"", "i", "ii", "iii", "iv", "v", "vi", "vii",
10.   "viii", "ix"};
11.
12.   while ((decimal_number > 100) || (decimal_number < 0)) {
13.     printf("Enter the decimal numbers in the range 1 to 100:\n");
14.     scanf("%d", &decimal_number);
15.   }
16.   a = decimal_number/10;
17.   b = decimal_number%10;
18.   printf("The equivalent roman is %s%s\n", x[a], y[b]);
19. } /* end main */
20.
```

Лекцийн агуулга

- Заагч хувьсагчийг тодорхойлох ба идэвхижүүлэх
- Заагчийн үйлдэл
- Аргументыг функцэд хаягаар дамжуулах
- `const` тодорхойлогчийг заагчид хэрэглэх
- Хаягаар-дуудахыг ашигласан бөмбөлөгөн эрэмбэлэлт
- `sizeof` үйлдэл
- Заагчийн илэрхийлэл ба заагчийн арифметик
- Заагч болон массивын уялдаа
- Заагчийн массив
- Функцийн заагч

Функцийн заагч

- Функцийн заагч
  - Функцийн хаягийг хадгалдаг
  - Массивын нэр эхний элементийн хаягийг заадагтай төстэй
  - Функцийн нэр бол функцийг тодорхойлсон кодын эхлэлийн хаяг болдог
- Функцийн заагчийг дараах байдлаар ашиглаж болно
  - Функц рүү дамжуулах
  - Массивт хадгалах
  - Өөр функцийн заагчид олгох

Функцийн заагч

- Жишээ: Бөмбөлөгөн эрэмбэлэлт
  - **`buble`** функц функцийн заагчийг авна
    - Энэ туслах функцийг дуудаж, эрэмбэлэлтийг өсөх болон буурах дараалалаар явуулахыг тодорхойлно
  - **`buble`** функци дэх функцийн заагч аргумент:

```
int ( *compare )( int a, int b );
```

    - Энэ бичлэг хоёр бүхэл тоо аваад, бүхэл утга буцаадаг функцийн заагч гэдгийг хэлж байна
  - Хэрвээ хаалтыг орхисон:

```
int *compare( int a, int b );
```

    - байвал хоёр бүхэл тоо аваад бүхлийн заагчийг буцаадаг функцийг тодорхойлох болно

Функцийн заагч

```
1.  /* Ex_56 Multipurpose sorting program using function pointers */
2.  #include <stdio.h>
3.  #define SIZE 10
4.
5.  /* prototypes */
6.  void bubble( int work[], const int size, int (*compare)( int a, int b ) );
7.  int ascending( int a, int b );
8.  int descending( int a, int b );
9.
10. int main( void )
11. {
12.    int order;   /* 1 for ascending order or 2 for descending order */
13.    int counter; /* counter */
14.
15.    /* initialize array a */
16.    int a[ SIZE ] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
17.
18.    printf( "Enter 1 to sort in ascending order,\n"
19.            "Enter 2 to sort in descending order: " );
20.    scanf( "%d",  &order );
21.
22.    printf( "\nData items in original order\n" );
23.
24.    /* output original array */
25.    for ( counter = 0; counter < SIZE; counter++ ) {
26.       printf( "%5d", a[ counter ] );
27.    } /* end for */
28.
```

buble функцийн заагчийг аргумент болно авч байна

Функцийн заагч...

```
29.  /* sort array in ascending order; pass function ascending as an
30.        argument to specify ascending sorting order */
31.     if ( order == 1 ) {
32.        bubble( a, SIZE, ascending );
33.        printf( "\nData items in ascending order\n" );
34.     } /* end if */
35.     else { /* pass function descending */
36.        bubble( a, SIZE, descending );
37.        printf( "\nData items in descending order\n" );
38.     } /* end else */
39.
40.     /* output sorted array */
41.     for ( counter = 0; counter < SIZE; counter++ ) {
42.        printf( "%5d", a[ counter ] );
43.     } /* end for */
44.
45.     printf( "\n" );
46.
47.     return 0; /* indicates successful termination */
48.
49. } /* end main */
50.
```

Хэрэглэгчийн сонголтоос хамаарч **buble** функц **ascending** болон **descending** функцийн аль нэгээр массивыг эрэмбэлнэ

Функцийн заагч…

```
51. /* multipurpose bubble sort; parameter compare is a pointer to
52.    the comparison function that determines sorting order */
53. void bubble( int work[], const int size, int (*compare)( int a, int b ) )
54. {
55.    int pass;  /* pass counter */
56.    int count; /* comparison counter */
57.
58.    void swap( int *element1Ptr, int *element2ptr ); /* prototype */
59.
60.    /* loop to control passes */
61.    for ( pass = 1; pass < size; pass++ ) {
62.
63.       /* loop to control number of comparisons per pass */
64.       for ( count = 0; count < size - 1; count++ ) {
65.
66.          /* if adjacent elements are out of order, swap them */
67.          if ( (*compare)( work[ count ], work[ count + 1 ] ) ) {
68.             swap( &work[ count ], &work[ count + 1 ] );
69.          } /* end if */
70.
71.       } /* end for */
72.
73.    } /* end for */
74.
75. } /* end function bubble */
76.
```

buble функцид дамжуулсан функцийн заагчаас хамаарч “дараалал алдагдсан” элементүүдийг сольж байгааг анхаар

Функцийн заагч...

```
77. /* swap values at memory locations to which element1Ptr and
78.    element2Ptr point */
79. void swap( int *element1Ptr, int *element2Ptr )
80. {
81.    int hold; /* temporary holding variable */
82.
83.    hold = *element1Ptr;
84.    *element1Ptr = *element2Ptr;
85.    *element2Ptr = hold;
86. } /* end function swap */
87.
88. /* determine whether elements are out of order for an ascending
89.    order sort */
90. int ascending( int a, int b )
91. {
92.    return b < a;    /* swap if b is less than a */
93.
94. } /* end function ascending */
95.
96. /* determine whether elements are out of order for a descending
97.    order sort */
98. int descending( int a, int b )
99. {
100.   return b > a;    /* swap if b is greater than a */
101.
102. } /* end function descending */
```

buble функцид дамжуулсан **ascending** програмын энэ цэгийг заана

buble функцид дамжуулсан **descending** програмын энэ цэгийг заана

Функцийн заагч…

```
Enter 1 to sort in ascending order,
Enter 2 to sort in descending order: 1

Data items in original order
    2    6    4    8   10   12   89   68   45   37
Data items in ascending order
    2    4    6    8   10   12   37   45   68   89


Enter 1 to sort in ascending order,
Enter 2 to sort in descending order: 2

Data items in original order
    2    6    4    8   10   12   89   68   45   37
Data items in descending order
   89   68   45   37   12   10    8    6    4    2
```

Дүгнэлт

- Заагч: өгөгдлийн_төрөл * заагч_хувьсагчийн_нэр (`%p`, 16 –тын бүхэл)
- Утгаар-дуудах(өөрчлөгдөхгүй), Хаягаар-дуудах(өөчлөгдөнө)
- **`const`** тодорхойлогч хувьсагчийг өөрчлөгдөхгүй болгодог (4 хувилбар)
- **`sizeof`** хэмжээг байтаар тодорхойлдог
- Заагчийн массив
- Функцийн заагч: функцийн нэр бол хаяг