CS200 – Програмчлалын үндэс

Лекц 09

Өгөгдлийн бүтцүүд

Профессор А.Эрдэнэбаатар

Лекцийн агуулга

- Өгагдлийн обьектод динамик санах ой хуваарьлаж, чөлөөлөх
  - `malloc`
  - `free`
- Бүтэц үүсгэж ажиллах
  - Холбоост жагсаалт
  - Стек
  - Дараалал
  - Хоёртын мод

Оршил

- Динамик өгөгдлийн бүтцүүд
  - Програмын ажиллагааны явцад ихэсч, багасдаг өгөгдлийн бүтцүүд
  - Холбоост жагсаалт
    - Оруулалт, устгалтыг хаана ч хийж болдог
  - Стек
    - Зөвхөн стекийн оройноос оруулж, устгаж болдог
  - Дараалал
    - Дарааллын араас оруулж, нүүрнээс нь устгадаг
  - Хоёртын мод
    - Өгөгдлийн давхардлыг үр дүнтэй арилгаж, өндөр хурдтай хайх, эрэмбэлэх боломж олгодог

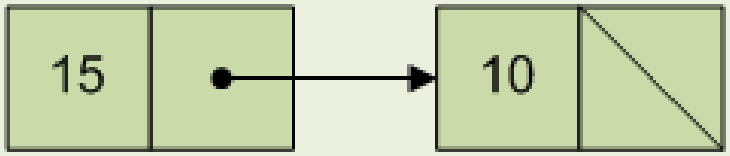
Өөртөө-хандагч бүтцүүд

- Өөртөө-хандагч бүтцүүд
  - Ижил төрлийн бүтцийн заагчийг агуулсан бүтэц
  - Бүтцүүдийг холбох замаар жагсаалт, дараалал, стек, мод зэрэг хэрэгтэй өгөгдлийн бүтцийг үүсгэж болно
  - **NULL** заагчаар төгсөнө

```
struct node {
   int data;
   struct node *nextPtr;
}
```

  - **nextPtr**
    - **node** төрлийн обьектын заагч
    - Холбоосын үүрэг гүйцэтгэдэг
      - Ө.х. нэг зангилааг нөгөөтэй холбоно

Өөртөө-хандагч бүтцүүд хоорондоо холбогддог



Лекцийн агуулга

- Өгагдлийн обьектод динамик санах ой хуваарьлаж, чөлөөлөх
  - `malloc`
  - `free`
- Бүтэц үүсгэж ажиллах
  - Холбоост жагсаалт
  - Стек
  - Дараалал
  - Хоёртын мод

Санах ойн динамик хуваарилалт

- Санах ойн динамик хуваавиралт
  - Гүйцэтгэлийн явцад ойг олж авч, чөлөөлнө
- **malloc**
  - Хуваарилах байтын тоог авна
    - Обьектын хэмжээг тодорхойлохдоо sizeof –г ашиглана
  - **void *** заагчийг буцаана
    - **viod *** заагчид ямар ч төрлийн заагчийг олгож болно
    - Хэрэв ой хүрэлцээгүй бол **NULL** буцна
  - Жишээ

```
newPtr = malloc( sizeof( struct node ) );
```

  - **free**
    - **malloc** –н хуваариласан ойг чөлөөлнө
    - Заагчийг аргумент болгож авна
    - Санамж
      - **malloc** –р хуваарилагдаагүй ойг чөлөөлөх нь алдаа болно
      - Чөлөөлсөн ойд хандах нь алдаа болно

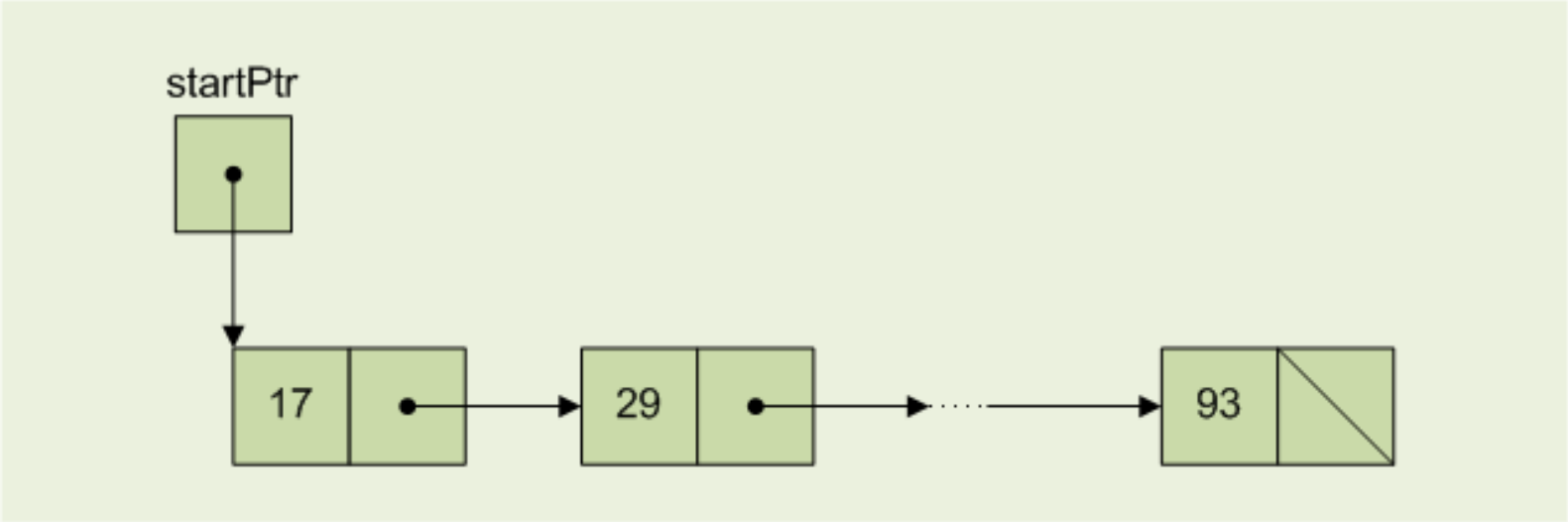
Лекцийн агуулга

- Өгагдлийн обьектод динамик санах ой хуваарьлаж, чөлөөлөх
  - `malloc`
  - `free`
- Бүтэц үүсгэж ажиллах
  - Холбоост жагсаалт
  - Стек
  - Дараалал
  - Хоёртын мод

Холбоост жагсаалт

- Холбоост жагсаалт
  - Зангилаа гэж нэрлэгддэг өөртөө-хандагч обьектуудын шулуун цуглуулга
  - Заагч холбоосоор холбогдоно
  - Жагсаалтын эхний зангилааний заагчаар хандана
  - Тухайн зангилааний гишүүн-заагчаар дараачийн занилаануудад хандана
  - Сүүлийн зангилааний заагч нь `NULL` байх ба энэ нь төгсгөлийг илэрхийлнэ
- Дараах зүйлд холбоост жагсаалтыг массивын оронд ашиглана
  - Өгөгдлийн элементийн тоо тодорхойгүй
  - Жагсаалт хурдан эрэмбэлэгдсэн байх шаардлагатай
- Холбоост жагсаалтын төрөл
  - Дан холбоост жагсаалт
  - Давхар холбоост жагсаалт
  - Цагираг холбоост жагсаалт

Холбоост жагсаалтын график дүрслэл



Холбоост жагсаалтын жишээ

```
1.  /* Ex_96 Operating and maintaining a list */
2.  #include <stdio.h>
3.  #include <stdlib.h>
4.
5.  /* self-referential structure */
6.  struct listNode {
7.     char data; /* each listNode contains a character */
8.     struct listNode *nextPtr; /* pointer to next node*/
9.  }; /* end structure listNode */
10.
11. typedef struct listNode ListNode; /* synonym for struct listNode */
12. typedef ListNode *ListNodePtr; /* synonym for ListNode* */
13.
14. /* prototypes */
15. void insert( ListNodePtr *sPtr, char value );
16. char delete( ListNodePtr *sPtr, char value );
17. int isEmpty( ListNodePtr sPtr );
18. void printList( ListNodePtr currentPtr );
19. void instructions( void );
20.
21. int main( void )
22. {
23.    ListNodePtr startPtr = NULL; /* initially there are no nodes */
24.    int choice; /* user's choice */
25.    char item;  /* char entered by user */
26.
27.    instructions(); /* display the menu */
28.    printf( "? " );
29.    scanf( "%d", &choice );
```

Жагсаалтын зангилаа бүр өгөгдлийн элемент болон дараачийн зангилааны заагчийг агуулна

```
30.
31.    /* loop while user does not choose 3 */
32.    while ( choice != 3 ) {
33.
34.       switch ( choice ) {
35.
36.          case 1:
37.             printf( "Enter a character: " );
38.             scanf( "\n%c", &item );
39.             insert( &startPtr, item ); /* insert item in list */
40.             printList( startPtr );
41.             break;
42.
43.          case 2:
44.
45.             /* if list is not empty */
46.             if ( !isEmpty( startPtr ) ) {
47.                printf( "Enter character to be deleted: " );
48.                scanf( "\n%c", &item );
49.
50.                /* if character is found, remove it */
51.                if ( delete( &startPtr, item ) ) { /* remove item */
52.                   printf( "%c deleted.\n", item );
53.                   printList( startPtr );
54.                } /* end if */
55.                else {
56.                   printf( "%c not found.\n\n", item );
57.                } /* end else */
58.
59.             } /* end if */
```

insert функц өгөгдлийг жагсаалтанд оруулна

delete функц өгөгдлийг жагсаалтнаас устгана

Холбоост жагсаалтын жишээ...

```
60.             else {
61.                printf( "List is empty.\n\n" );
62.             } /* end else */
63.
64.             break;
65.
66.          default:
67.             printf( "Invalid choice.\n\n" );
68.             instructions();
69.             break;
70.
71.       } /* end switch */
72.
73.       printf( "? " );
74.       scanf( "%d", &choice );
75.    } /* end while */
76.
77.    printf( "End of run.\n" );
78.
79.    return 0; /* indicates successful termination */
80.
81. } /* end main */
82.
```

```
83. /* display program instructions to user */
84. void instructions( void )
85. {
86.    printf( "Enter your choice:\n"
87.       "   1 to insert an element into the list.\n"
88.       "   2 to delete an element from the list.\n"
89.       "   3 to end.\n" );
90. } /* end function instructions */
91.
92. /* Insert a new value into the list in sorted order */
93. void insert( ListNodePtr *sPtr, char value )
94. {
95.    ListNodePtr newPtr;      /* pointer to new node */
96.    ListNodePtr previousPtr; /* pointer to previous node in list */
97.    ListNodePtr currentPtr;  /* pointer to current node in list */
98.
99.    newPtr = malloc( sizeof( ListNode ) ); /* create node */
100.
101.   if ( newPtr != NULL ) { /* is space available */
102.      newPtr->data = value; /* place value in node */
103.      newPtr->nextPtr = NULL; /* node does not link to another node */
104.
105.      previousPtr = NULL;
106.      currentPtr = *sPtr;
107.
108.      /* loop to find the correct location in the list */
109.      while ( currentPtr != NULL && value > currentPtr->data ) {
110.         previousPtr = currentPtr;          /* walk to ...   */
111.         currentPtr = currentPtr->nextPtr;  /* ... next node */
112.      } /* end while */
```

Зангилааг жагсаалтанд оруулахын тулд эхлээд түүнд санах ой хуваарилах ёстой

while цикл жагсаалт дахь шинэ зангилааны байрыг хайна

Холбоост жагсаалтын жишээ...

```
113.
114.       /* insert new node at beginning of list */
115.       if ( previousPtr == NULL ) {
116.          newPtr->nextPtr = *sPtr;
117.          *sPtr = newPtr;
118.       } /* end if */
119.       else { /* insert new node between previousPtr and currentPtr */
120.          previousPtr->nextPtr = newPtr;
121.          newPtr->nextPtr = currentPtr;
122.       } /* end else */
123.
124.    } /* end if */
125.    else {
126.       printf( "%c not inserted. No memory available.\n", value );
127.    } /* end else */
128.
129. } /* end function insert */
130.
131. /* Delete a list element */
132. char delete( ListNodePtr *sPtr, char value )
133. {
134.    ListNodePtr previousPtr; /* pointer to previous node in list */
135.    ListNodePtr currentPtr;  /* pointer to current node in list */
136.    ListNodePtr tempPtr;     /* temporary node pointer */
137.
```

Хэрэв жагсаалтанд зангилаа бйхгүй бол шинэ зангилаа “эхлэл” –н зангилаа болно

Эсрэг тохиолдолд заагчдыг өөрчлөх замаар шинэ зангилааг өөр хоёр зангилааны дунд оруулна

```
138.    /* delete first node */
139.    if ( value == ( *sPtr )->data ) {
140.       tempPtr = *sPtr; /* hold onto node being removed */
141.       *sPtr = ( *sPtr )->nextPtr; /* de-thread the node */
142.       free( tempPtr ); /* free the de-threaded node */
143.       return value;
144.    } /* end if */
145.    else {
146.       previousPtr = *sPtr;
147.       currentPtr = ( *sPtr )->nextPtr;
148.
149.       /* loop to find the correct location in the list */
150.       while ( currentPtr != NULL && currentPtr->data != value ) {
151.          previousPtr = currentPtr;         /* walk to ...   */
152.          currentPtr = currentPtr->nextPtr; /* ... next node */
153.       } /* end while */
154.
155.       /* delete node at currentPtr */
156.       if ( currentPtr != NULL ) {
157.          tempPtr = currentPtr;
158.          previousPtr->nextPtr = currentPtr->nextPtr;
159.          free( tempPtr );
160.          return value;
161.       } /* end if */
162.
163.    } /* end else */
164.
165.    return '\0';
166.
167. } /* end function delete */
```

while цикл жагсаалт дахь зангилааны байрыг хайна

Нэгэнт зангилааг олсон бол заагчийг өөрчилж түүнийг устгаад, зангилааны санах ойг чөлөөлнө

```
168.
169./* Return 1 if the list is empty, 0 otherwise */
170.int isEmpty( ListNodePtr sPtr )
171.{
172.   return sPtr == NULL;
173.
174.} /* end function isEmpty */
175.
176./* Print the list */
177.void printList( ListNodePtr currentPtr )
178.{
179.
180.   /* if list is empty */
181.   if ( currentPtr == NULL ) {
182.      printf( "List is empty.\n\n" );
183.   } /* end if */
184.   else {
185.      printf( "The list is:\n" );
186.
187.      /* while not the end of the list */
188.      while ( currentPtr != NULL ) {
189.         printf( "%c --> ", currentPtr->data );
190.         currentPtr = currentPtr->nextPtr;
191.      } /* end while */
192.
193.      printf( "NULL\n\n" );
194.   } /* end else */
195.
196.} /* end function printList */
```

Эхний зангилаа NULL бол жагсаалтанд зангилаа алга (хоосон жагсаалт)

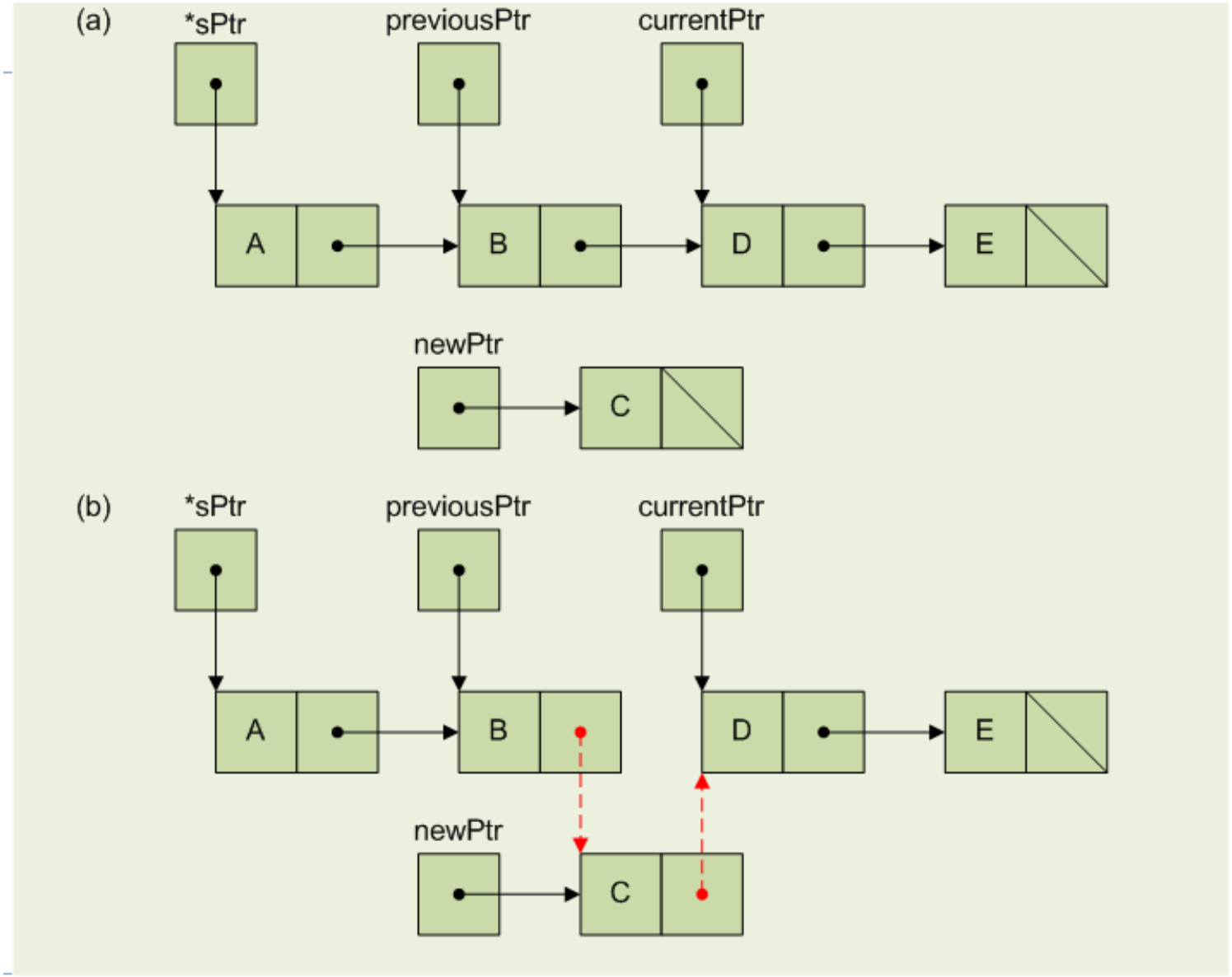
Холбоост жагсаалтын жишээ...

```
Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to end.
? 1
Enter a character: B
The list is:
B --> NULL

? 1
Enter a character: A
The list is:
A --> B --> NULL

? 2
Enter character to be deleted: D
D not found.

? 2
Enter character to be deleted: B
B deleted.
The list is:
A --> NULL

? 2
Enter character to be deleted: A
A deleted.
List is empty.
```

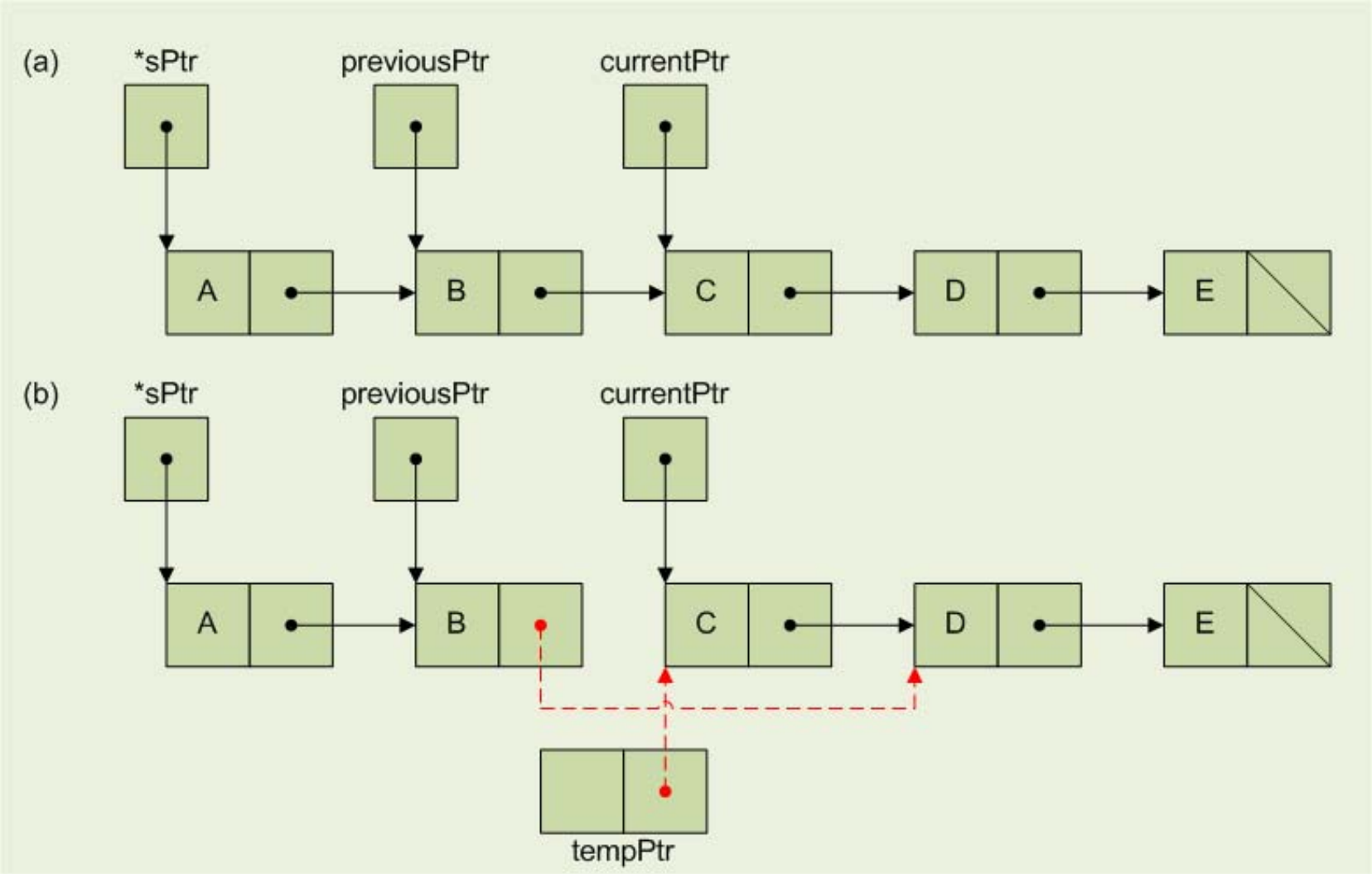
Холбоост жагсаалтын жишээ...

```
? 4
Invalid choice.

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to end.
? 3
End of run.

? 1
Enter a character: C
The list is:
C --> NULL

? 2
Enter character to be deleted: C
C deleted:
List is empty.

? 4
Invalid choice.

Enter your choice:
   1 to insert an element into the list.
   2 to delete an element from the list.
   3 to end.
? 3
End of run.
```

Жагсаалтанд дарааллаар зангилаа оруулах



Жагсаалтнаас зангилаа устгах



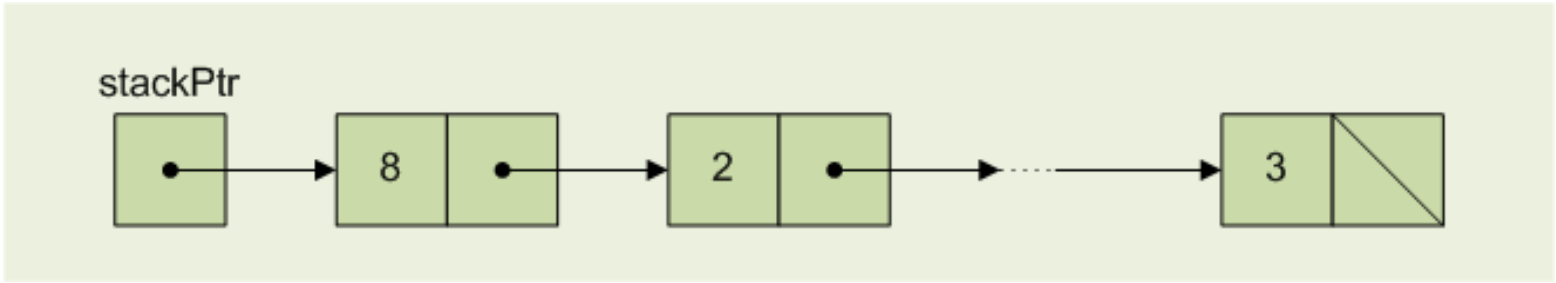
Лекцийн агуулга

- Өгагдлийн обьектод динамик санах ой хуваарьлаж, чөлөөлөх
  - `malloc`
  - `free`
- Бүтэц үүсгэж ажиллах
  - Холбоост жагсаалт
  - Стек
  - Дараалал
  - Хоёртын мод

Стек

- Стек
  - Шинэ зангилааг зөвхөн оройноос нэмэж, устгаж болно
  - Давхарлаж хураасан тавагтай төстэй
  - LIFO – Last-in, First-out (Сүүлд-ор, Түрүүлж-гар)
  - Стекийн ёроолыг `NULL` руу заасан холбоос илтгэдэг
  - Холбоост жагсаалтын шахсан хувилбар
- push
  - Стекийн оройд шинэ зангилаа нэмнэ
- pop
  - Оройгоос зангилаа устгана
  - Татсан утгыг хадгална
  - Таталт амжилттай болсон бол `true` буцна

Стекийн график дүрслэл



Стекийн жишээ

```
1.  /* Ex_97 dynamic stack program */
2.  #include <stdio.h>
3.  #include <stdlib.h>
4.
5.  /* self-referential structure */
6.  struct stackNode {
7.     int data;                  /* define data as an int */
8.     struct stackNode *nextPtr; /* stackNode pointer */
9.  }; /* end structure stackNode */
10.
11. typedef struct stackNode StackNode; /* synonym for struct stackNode */
12. typedef StackNode *StackNodePtr; /* synonym for StackNode* */
13.
14. /* prototypes */
15. void push( StackNodePtr *topPtr, int info );
16. int pop( StackNodePtr *topPtr );
17. int isEmpty( StackNodePtr topPtr );
18. void printStack( StackNodePtr currentPtr );
19. void instructions( void );
20.
21. /* function main begins program execution */
22. int main( void )
23. {
24.    StackNodePtr stackPtr = NULL; /* points to stack top */
25.    int choice; /* user's menu choice */
26.    int value;  /* int input by user */
27.
28.    instructions(); /* display the menu */
29.    printf( "? " );
```

Стекийн зангилаа бүр өгөгдлийн элемент болон дараачийн зангилааны заагчийг агуулна

```
30.    scanf( "%d", &choice );
31.
32.    /* while user does not enter 3 */
33.    while ( choice != 3 ) {
34.
35.       switch ( choice ) {
36.
37.          /* push value onto stack */
38.          case 1:
39.             printf( "Enter an integer: " );
40.             scanf( "%d", &value );
41.             push( &stackPtr, value );
42.             printStack( stackPtr );
43.             break;
44.
45.          /* pop value off stack */
46.          case 2:
47.
48.             /* if stack is not empty */
49.             if ( !isEmpty( stackPtr ) ) {
50.                printf( "The popped value is %d.\n", pop( &stackPtr ) );
51.             } /* end if */
52.
53.             printStack( stackPtr );
54.             break;
55.
56.          default:
57.             printf( "Invalid choice.\n\n" );
58.             instructions();
59.             break;
```

```
60.
61.       } /* end switch */
62.
63.       printf( "? " );
64.       scanf( "%d", &choice );
65.    } /* end while */
66.
67.    printf( "End of run.\n" );
68.
69.    return 0; /* indicates successful termination */
70.
71. } /* end main */
72.
73. /* display program instructions to user */
74. void instructions( void )
75. {
76.    printf( "Enter choice:\n"
77.       "1 to push a value on the stack\n"
78.       "2 to pop a value off the stack\n"
79.       "3 to end program\n" );
80. } /* end function instructions */
81.
82. /* Insert a node at the stack top */
83. void push( StackNodePtr *topPtr, int info )
84. {
85.    StackNodePtr newPtr; /* pointer to new node */
86.
87.    newPtr = malloc( sizeof( StackNode ) );
88.
```

Стект зангилаа оруулахын тулд эхлээд түүнд санах ой хуваарилах ёстой

Стекийн жишээ...

```
89. /* insert the node at stack top */
90.    if ( newPtr != NULL ) {
91.       newPtr->data = info;
92.       newPtr->nextPtr = *topPtr;
93.       *topPtr = newPtr;
94.    } /* end if */
95.    else { /* no space available */
96.       printf( "%d not inserted. No memory available.\n", info );
97.    } /* end else */
98.
99. } /* end function push */
100.
101./* Remove a node from the stack top */
102.int pop( StackNodePtr *topPtr )
103.{
104.   StackNodePtr tempPtr; /* temporary node pointer */
105.   int popValue; /* node value */
106.
107.   tempPtr = *topPtr;
108.   popValue = ( *topPtr )->data;
109.   *topPtr = ( *topPtr )->nextPtr;
110.   free( tempPtr );
111.
112.   return popValue;
113.
114.} /* end function pop */
115.
```

Стект зангилааг үргэлж оройноос нь оруулдаг тул зангилааны байрлал хайх шаардлагагүй

Оруулсан зангилаа шинэ орой болно

Стект зангилааг үргэлж оройноос нь устгадаг тул зангилааны байрлал хайх шаардлагагүй

Хоёр дахь зангилаа шинэ орой болно

Татсан зангилааны санах ойг чөлөөлнө

Стекийн жишээ...

```
116. /* Print the stack */
117. void printStack( StackNodePtr currentPtr )
118. {
119.
120.    /* if stack is empty */
121.    if ( currentPtr == NULL ) {
122.       printf( "The stack is empty.\n\n" );
123.    } /* end if */
124.    else {
125.       printf( "The stack is:\n" );
126.
127.       /* while not the end of the stack */
128.       while ( currentPtr != NULL ) {
129.          printf( "%d --> ", currentPtr->data );
130.          currentPtr = currentPtr->nextPtr;
131.       } /* end while */
132.
133.       printf( "NULL\n\n" );
134.    } /* end else */
135.
136. } /* end function printList */
137.
138. /* Return 1 if the stack is empty, 0 otherwise */
139. int isEmpty( StackNodePtr topPtr )
140. {
141.    return topPtr == NULL;
142.
143. } /* end function isEmpty */
```

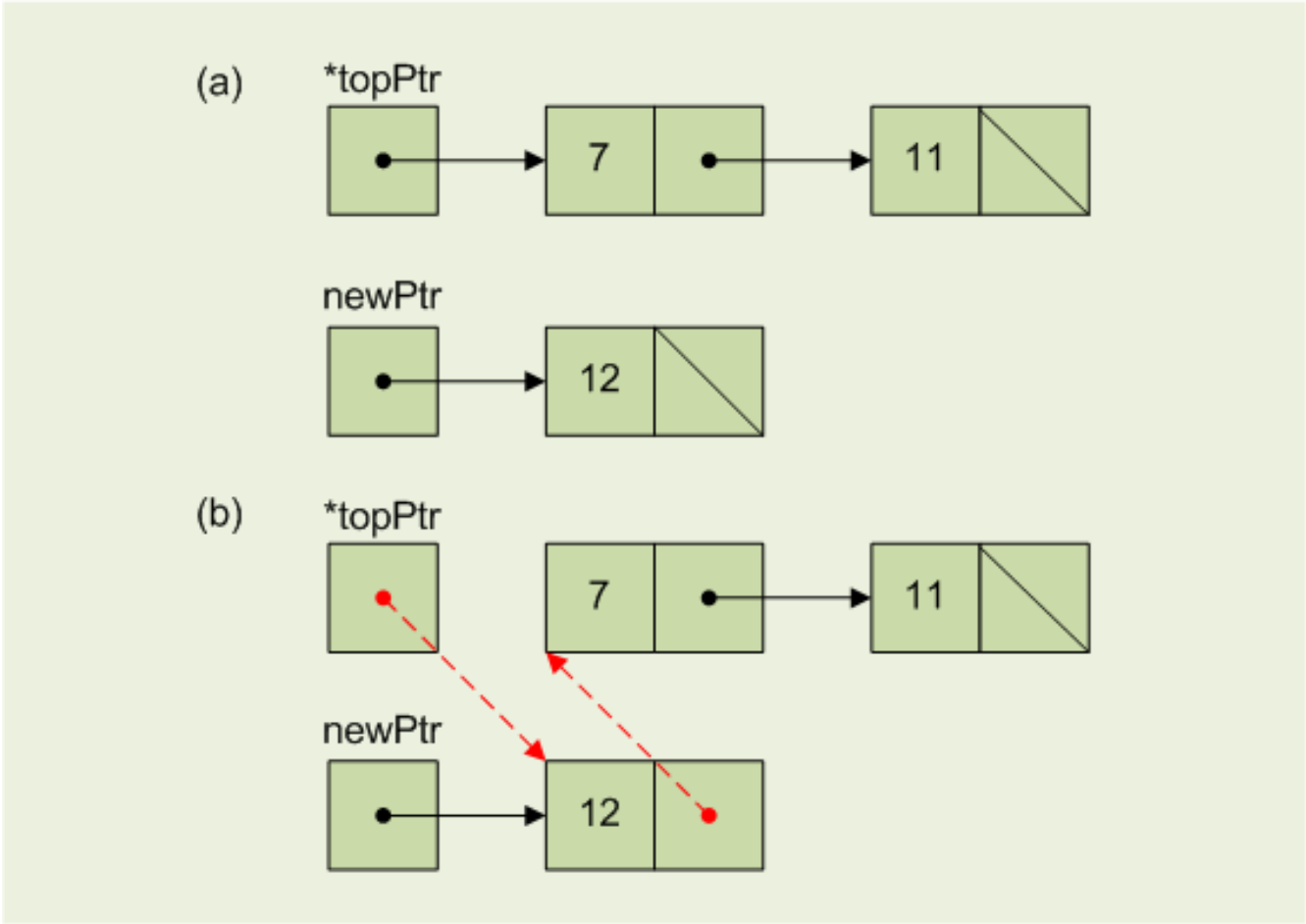
Стекийн жишээ...

```
Enter choice:
1 to push a value on the stack
2 to pop a value off the stack
3 to end program
?1
Enter an integer: 5
The stack is:
5 --> NULL

? 1
Enter an integer: 6
The stack is:
6 --> 5 --> NULL

? 1
Enter an integer: 4
The stack is:
4 --> 6 --> 5 --> NULL

? 2
The popped value is: 4.
The stack is:
6 --> 5 --> NULL
```

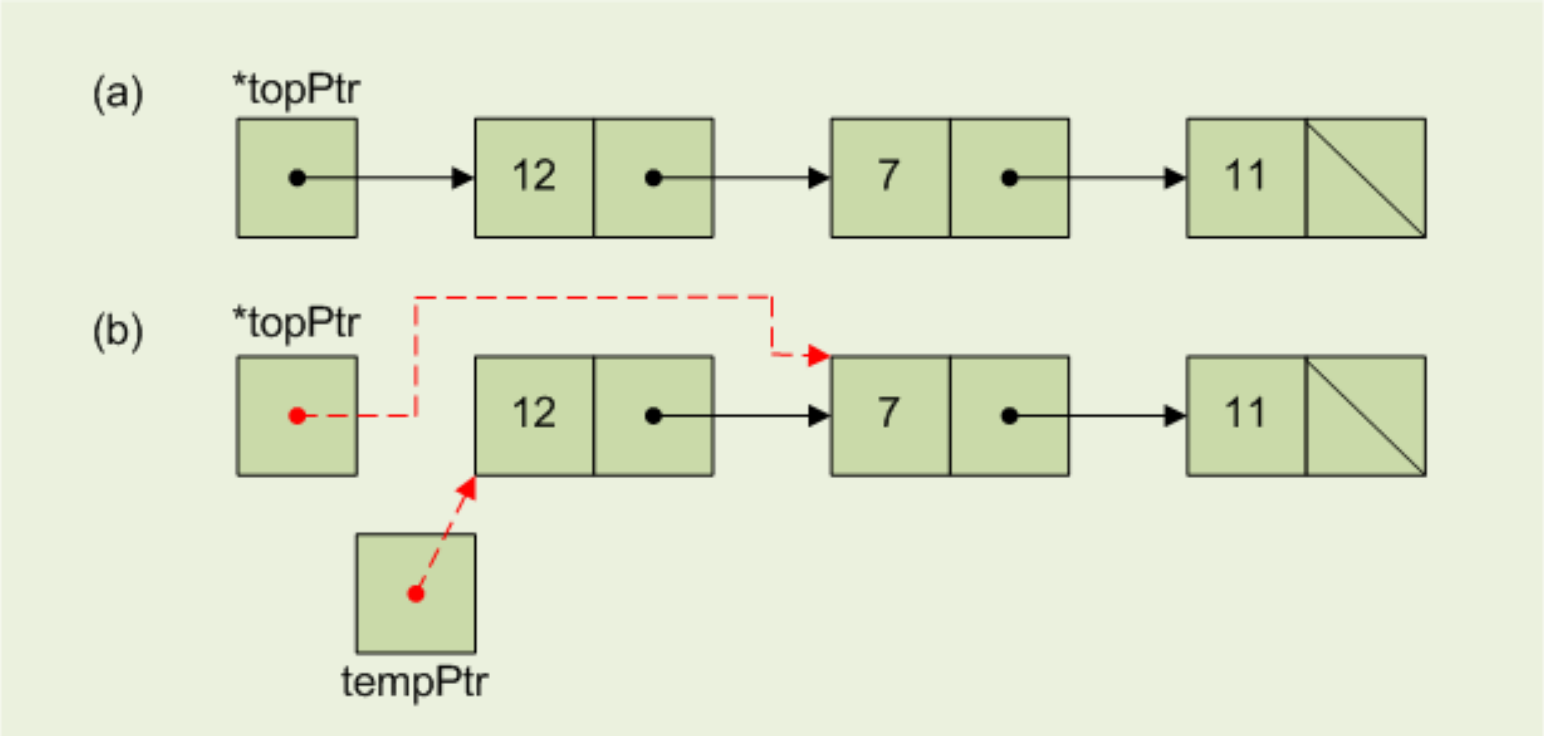
Стекийн жишээ...

```
? 2
The popped value is: 6.
The stack is:
5 --> NULL

? 2
The popped value is: 5.
The stack is empty.

? 2
The stack is empty.

? 4
Invalid choice.

Enter choice:
1 to push a value on the stack
2 to pop a value off the stack
3 to end program

? 3
End of run.
```

push үйлдэл



pop үйлдэл



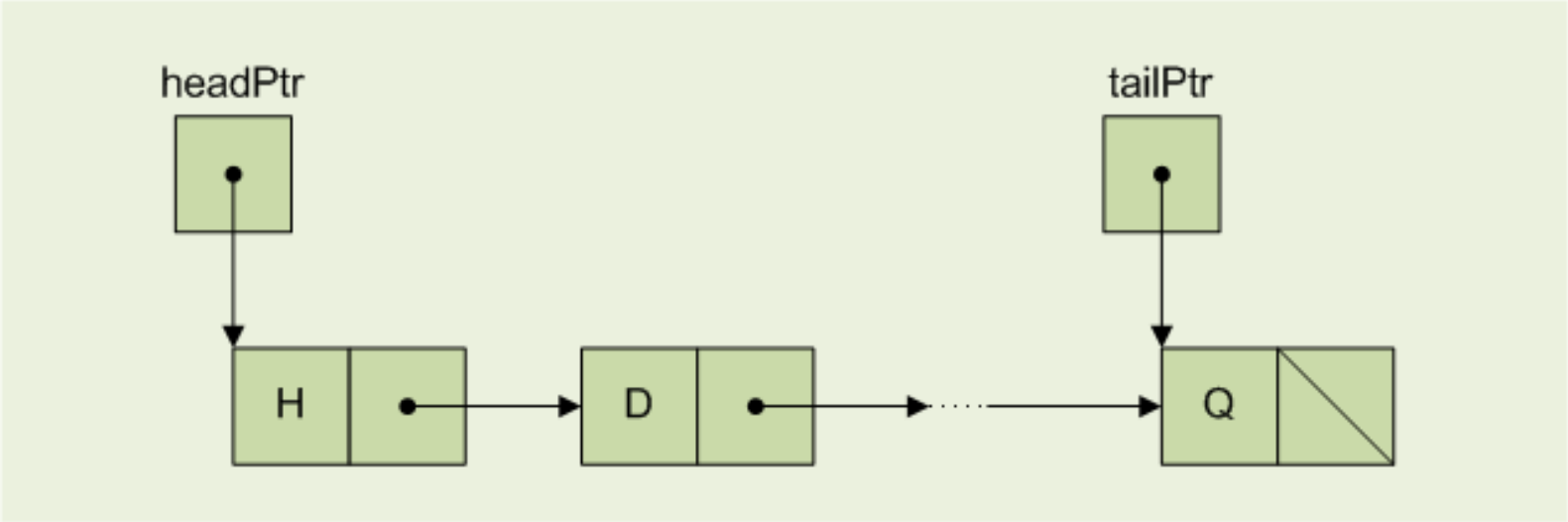
Лекцийн агуулга

- Өгагдлийн обьектод динамик санах ой хуваарьлаж, чөлөөлөх
  - `malloc`
  - `free`
- Бүтэц үүсгэж ажиллах
  - Холбоост жагсаалт
  - Стек
  - Дараалал
  - Хоёртын мод

Дараалал

- Дараалал
  - Дэлгүүрийн төлбөр төлөх дараалалтай төстэй
  - FIFO – First-in, First-out (Түрүүлж-ор, Түрүүлж-гар)
  - Зангилааг зөвхөн толгойноос нь устгана
  - Зангилааг зөвхөн сүүлнээс нь оруулна
- Оруулах ба гаргах үйлдлүүд
  - enqueue(оруулах) ба dequeue(устгах)

Дарааллын график дүрслэл



Дарааллын жишээ

```
1.  /* Ex_98 Operating and maintaining a queue */
2.
3.  #include <stdio.h>
4.  #include <stdlib.h>
5.
6.  /* self-referential structure */
7.  struct queueNode {
8.     char data;                  /* define data as a char */
9.     struct queueNode *nextPtr; /* queueNode pointer */
10. }; /* end structure queueNode */
11.
12. typedef struct queueNode QueueNode;
13. typedef QueueNode *QueueNodePtr;
14.
15. /* function prototypes */
16. void printQueue( QueueNodePtr currentPtr );
17. int isEmpty( QueueNodePtr headPtr );
18. char dequeue( QueueNodePtr *headPtr, QueueNodePtr *tailPtr );
19. void enqueue( QueueNodePtr *headPtr, QueueNodePtr *tailPtr,
20.                char value );
21. void instructions( void );
22.
23. /* function main begins program execution */
24. int main( void )
25. {
26.    QueueNodePtr headPtr = NULL; /* initialize headPtr */
27.    QueueNodePtr tailPtr = NULL; /* initialize tailPtr */
28.    int choice;                  /* user's menu choice */
29.    char item;                   /* char input by user */
```

Дарааллын зангилаа бүр өгөгдлийн элемент болон дараачийн зангилааны заагчийг агуулна

Холбоост жагсаалт, стекээс ялгаатай нь дараалал тогойгоос гадна сүүлээ хянадагийг санаарай

Дарааллын жишээ...

```
30.
31.    instructions(); /* display the menu */
32.    printf( "? " );
33.    scanf( "%d", &choice );
34.
35.    /* while user does not enter 3 */
36.    while ( choice != 3 ) {
37.
38.       switch( choice ) {
39.
40.          /* enqueue value */
41.          case 1:
42.             printf( "Enter a character: " );
43.             scanf( "\n%c", &item );
44.             enqueue( &headPtr, &tailPtr, item );
45.             printQueue( headPtr );
46.             break;
47.
48.          /* dequeue value */
49.          case 2:
50.
51.             /* if queue is not empty */
52.             if ( !isEmpty( headPtr ) ) {
53.                item = dequeue( &headPtr, &tailPtr );
54.                printf( "%c has been dequeued.\n", item );
55.             } /* end if */
56.
57.             printQueue( headPtr );
58.             break;
```

Дарааллын жишээ...

```
59.
60.          default:
61.             printf( "Invalid choice.\n\n" );
62.             instructions();
63.             break;
64.
65.       } /* end switch */
66.
67.       printf( "? " );
68.       scanf( "%d", &choice );
69.    } /* end while */
70.
71.    printf( "End of run.\n" );
72.
73.    return 0; /* indicates successful termination */
74.
75. } /* end main */
76.
77. /* display program instructions to user */
78. void instructions( void )
79. {
80.    printf ( "Enter your choice:\n"
81.             "   1 to add an item to the queue\n"
82.             "   2 to remove an item from the queue\n"
83.             "   3 to end\n" );
84. } /* end function instructions */
```

Дарааллын жишээ...

```
85.
86. /* insert a node a queue tail */
87. void enqueue( QueueNodePtr *headPtr, QueueNodePtr *tailPtr,
88.                char value )
89. {
90.    QueueNodePtr newPtr; /* pointer to new node */
91.
92.    newPtr = malloc( sizeof( QueueNode ) );
93.
94.    if ( newPtr != NULL ) { /* is space available */
95.       newPtr->data = value;
96.       newPtr->nextPtr = NULL;
97.
98.       /* if empty, insert node at head */
99.       if ( isEmpty( *headPtr ) ) {
100.         *headPtr = newPtr;
101.      } /* end if */
102.      else {
103.         ( *tailPtr )->nextPtr = newPtr;
104.      } /* end else */
105.
106.      *tailPtr = newPtr;
107.   } /* end if */
108.   else {
109.      printf( "%c not inserted. No memory available.\n", value );
110.   } /* end else */
111.
112.} /* end function enqueue */
```

Дараалалд зангилаа оруулахын тулд эхлээд түүнд санах ой хуваарилах ёстой

Дараалалд зангилааг үргэлж сүүлнээс нь оруулдаг тул зангилааны байрлал хайх шаардлагагүй

Хэрэв дараалал хоосон бол оруулан зангилаа шинэ толгой болохын зэрэгцээ шинэ сүүл болно

Оруулсан зангилаа шинэ сүүл болно

Дарааллын жишээ...

```
113.
114./* remove node from queue head */
115.char dequeue( QueueNodePtr *headPtr, QueueNodePtr *tailPtr )
116.{
117.   char value;            /* node value */
118.   QueueNodePtr tempPtr; /* temporary node pointer */
119.
120.   value = ( *headPtr )->data;
121.   tempPtr = *headPtr;
122.   *headPtr = ( *headPtr )->nextPtr;
123.
124.   /* if queue is empty */
125.   if ( *headPtr == NULL ) {
126.      *tailPtr = NULL;
127.   } /* end if */
128.
129.   free( tempPtr );
130.
131.   return value;
132.
133.} /* end function dequeue */
134.
135./* Return 1 if the list is empty, 0 otherwise */
136.int isEmpty( QueueNodePtr headPtr )
137.{
138.   return headPtr == NULL;
139.
140.} /* end function isEmpty */
```

Дараалалд зангилааг үргэлж сүүлнээс нь устгадаг тул зангилааны байрлал хайх шаардлагагүй

Хоёр дахь зангилаа шинэ толгой болно

Хэрэв устгасан зангилаа дарааллын сүүлийнх бол тэр сүүл, мөн толгой болдог. Иймд **`tailPTR`** –г **`NULL`** болгох ёстой

Устгасан зангилааны санах ойг чөлөөлнө

Дарааллын жишээ...

```
141.
142./* Print the queue */
143.void printQueue( QueueNodePtr currentPtr )
144.{
145.
146.   /* if queue is empty */
147.   if ( currentPtr == NULL ) {
148.      printf( "Queue is empty.\n\n" );
149.   } /* end if */
150.   else {
151.      printf( "The queue is:\n" );
152.
153.      /* while not end of queue */
154.      while ( currentPtr != NULL ) {
155.         printf( "%c --> ", currentPtr->data );
156.         currentPtr = currentPtr->nextPtr;
157.      } /* end while */
158.
159.      printf( "NULL\n\n" );
160.   } /* end else */
161.
162.} /* end function printQueue */
```

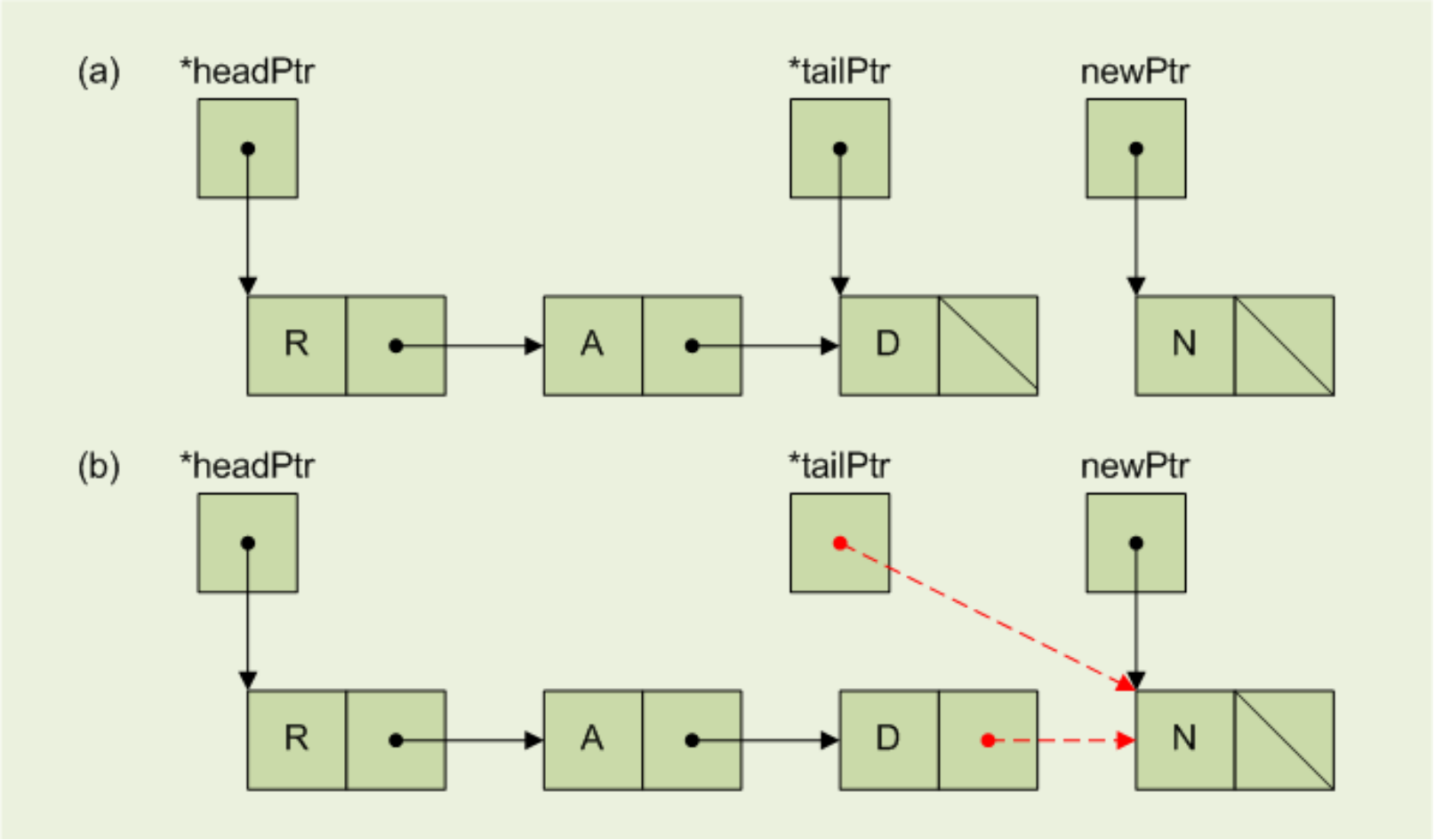
Дарааллын жишээ...

```
Enter your choice:
   1 to add an item to the queue
   2 to remove an item from the queue
   3 to end
? 1
Enter a character: A
The queue is
A --> NULL

? 1
Enter a character: B
The queue is
A --> B --> NULL

? 1
Enter a character: C
The queue is
A --> B --> C --> NULL

? 2
A has been dequeued.
The queue is
B --> C --> NULL
```

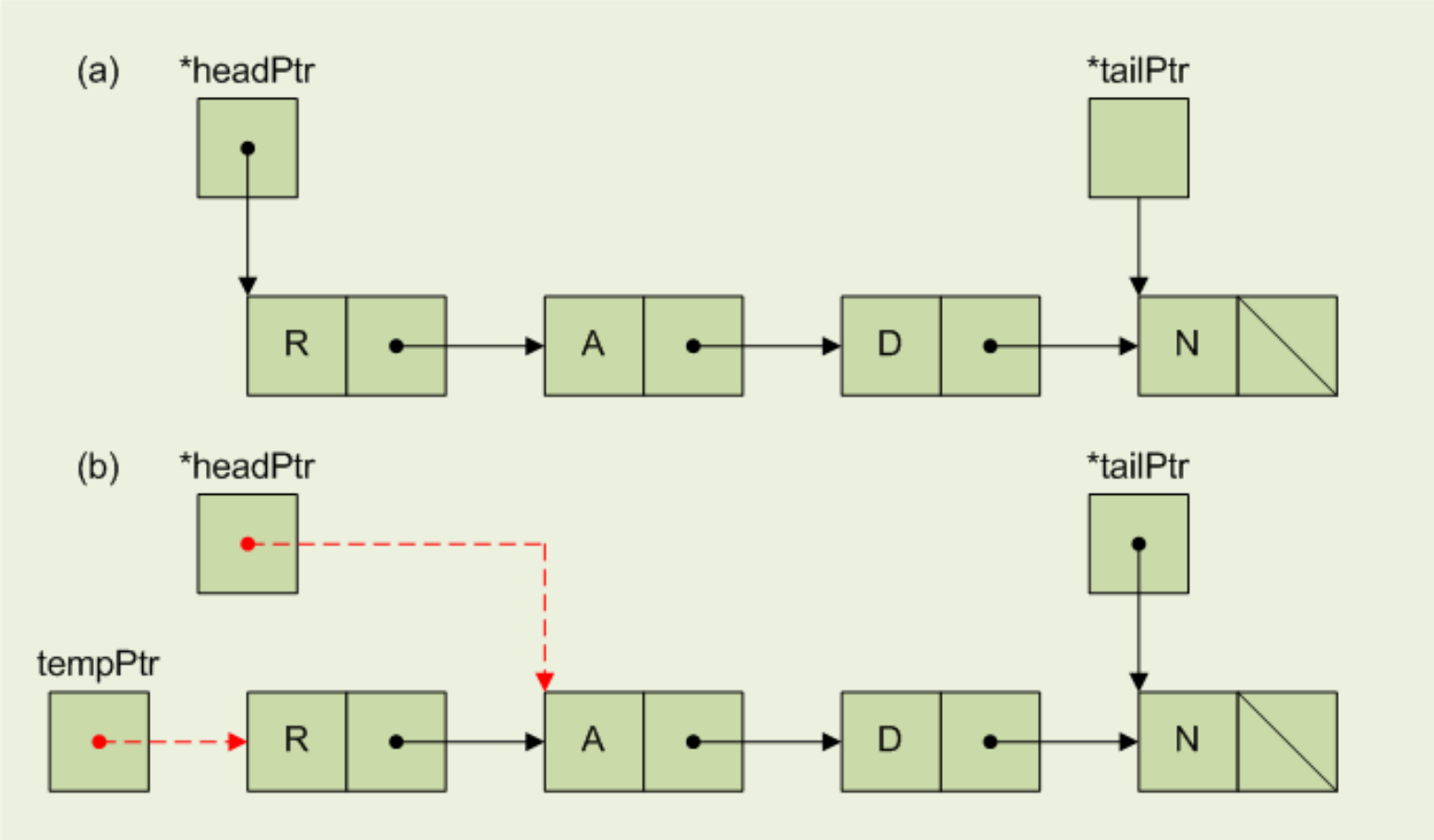
Дарааллын жишээ...

```
? 2
B has been dequeued.
The queue is
C --> NULL

? 2
C has been dequeued.
Queue is empty.

? 2
Queue is empty.

? 4
Invalid choice.

Enter your choice:
   1 to add an item to the queue
   2 to remove an item from the queue
   3 to end
? 3
End of run.
```

enqueue үйлдэл



dequeue үйлдэл



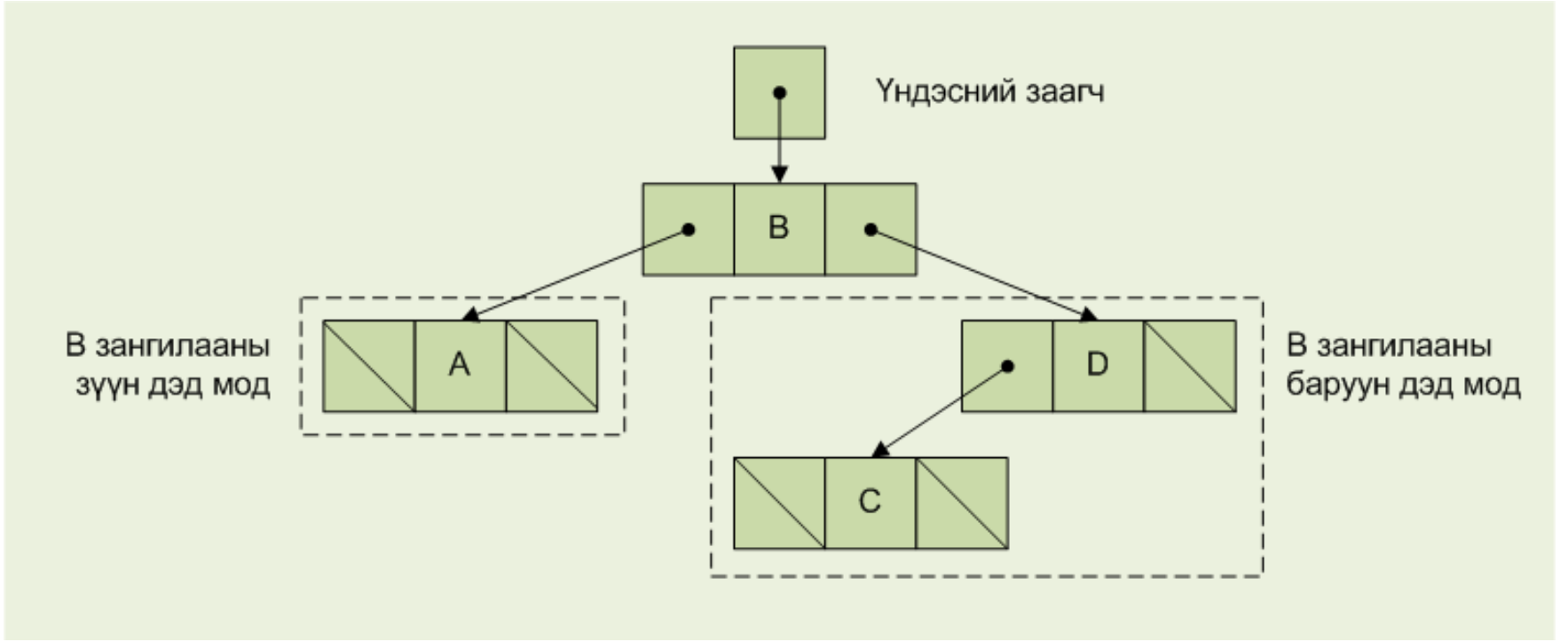
Лекцийн агуулга

- Өгагдлийн обьектод динамик санах ой хуваарьлаж, чөлөөлөх
  - `malloc`
  - `free`
- Бүтэц үүсгэж ажиллах
  - Холбоост жагсаалт
  - Стек
  - Дараалал
  - Хоёртын мод

Мод

- Модны зангилаа нь хоёр буюу түүнээс олон холбоостой байдаг
  - Өмнө үзсэн бүх бүтэц зөвхөн нэг холбоостой байлаа
- Хоёртын мод
  - Бүх зангилаа нь хоёр зангилаатай
    - Тэдгээрийн алин ч NULL байж болно
  - Модны эхний зангилааг үндэс гэнэ
  - Үндэсний (зангилааны) холбоос бүр хүүхдэд ханддаг
  - Хүүхэдгүй зангилааг навч гэнэ

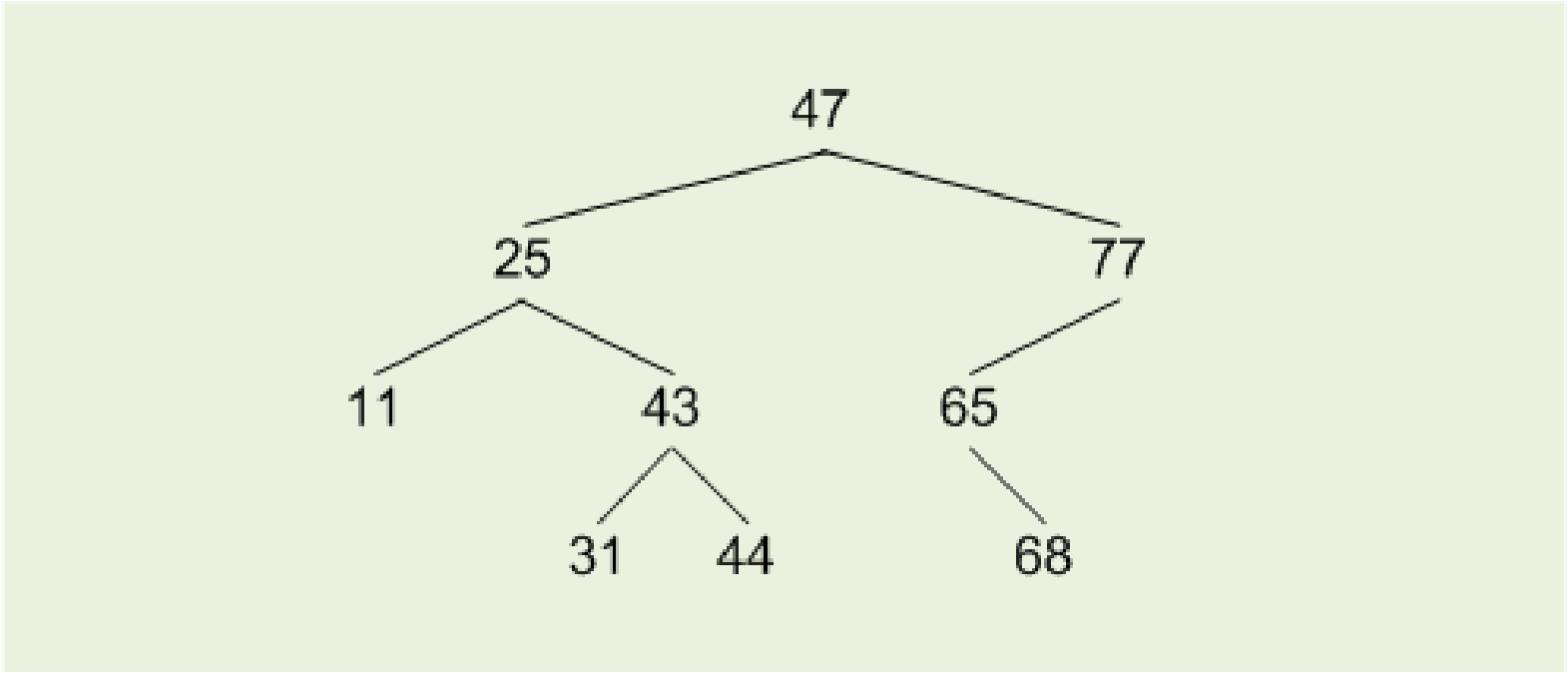
Хоёртын модны график дүрслэл



Мод

- Хоёртын хайлтын мод
  - Зүүн дэд модны утга эцгээсээ бага
  - Баруун дэд модны утга эцгээсээ их
  - Давхардлыг арилгахад тусладаг
  - Тэнцвэржүүлсэн модон дээр хайлт хурдан явагддаг. Максимум log(n) харьцуулалт хийгдэнэ

Хоёртын хайлтын мод



Мод

- Модны нэвтрэлт
  - Inorder нэвтрэлт – зангилааны утгуудыг өсөх дарааллаар хэвлэнэ
    1. Зүүн дэд модонд inorder нэвтрэлт хий
    2. Зангилааны утгыг боловсруул (жишээ нь: хэвлэх)
    3. Баруун дэд модонд inorder нэвтрэлт хий
  - Preorder нэвтрэлт
    1. Зангилааны утгыг боловсруул
    2. Зүүн дэд модонд preorder нэвтрэлт хий
    3. Баруун дэд модонд preorder нэвтрэлт хий
  - Postorder нэвтрэлт
    1. Зүүн дэд модонд postorder нэвтрэлт хий
    2. Баруун дэд модонд postorder нэвтрэлт хий
    3. Зангилааны утгыг боловсруул

Модны жишээ

```
1.  /* Ex_99 Create a binary tree and traverse it
2.     preorder, inorder, and postorder */
3.  #include <stdio.h>
4.  #include <stdlib.h>
5.  #include <time.h>
6.
7.  /* self-referential structure */
8.  struct treeNode {
9.     struct treeNode *leftPtr;  /* pointer to left subtree */
10.    int data; /* node value */
11.    struct treeNode *rightPtr; /* pointer to right subtree */
12. }; /* end structure treeNode */
13.
14. typedef struct treeNode TreeNode; /* synonym for struct treeNode */
15. typedef TreeNode *TreeNodePtr; /* synonym for TreeNode* */
16.
17. /* prototypes */
18. void insertNode( TreeNodePtr *treePtr, int value );
19. void inOrder( TreeNodePtr treePtr );
20. void preOrder( TreeNodePtr treePtr );
21. void postOrder( TreeNodePtr treePtr );
22.
23. /* function main begins program execution */
24. int main( void )
25. {
26.    int i; /* counter to loop from 1-10 */
27.    int item; /* variable to hold random values */
28.    TreeNodePtr rootPtr = NULL; /* tree initially empty */
29.
```

Модны зангилаа бүр өгөгдлийн элемент, зүүн болон баруун хүү зангилааны заагчийг агуулна

Модны жишээ…

```
30.    srand( time( NULL ) );
31.    printf( "The numbers being placed in the tree are:\n" );
32.
33.    /* insert random values between 0 and 14 in the tree */
34.    for ( i = 1; i <= 10; i++ ) {
35.       item = rand() % 15;
36.       printf( "%3d", item );
37.       insertNode( &rootPtr, item );
38.    } /* end for */
39.
40.    /* traverse the tree preOrder */
41.    printf( "\n\nThe preOrder traversal is:\n" );
42.    preOrder( rootPtr );
43.
44.    /* traverse the tree inOrder */
45.    printf( "\n\nThe inOrder traversal is:\n" );
46.    inOrder( rootPtr );
47.
48.    /* traverse the tree postOrder */
49.    printf( "\n\nThe postOrder traversal is:\n" );
50.    postOrder( rootPtr );
51.
52.    printf("\n");
53.    return 0; /* indicates successful termination */
54.
55. } /* end main */
```

Модны жишээ…

```
56.
57. /* insert node into tree */
58. void insertNode( TreeNodePtr *treePtr, int value )
59. {
60.
61.    /* if tree is empty */
62.    if ( *treePtr == NULL ) {
63.       *treePtr = malloc( sizeof( TreeNode ) );
64.
65.       /* if memory was allocated then assign data */
66.       if ( *treePtr != NULL ) {
67.          ( *treePtr )->data = value;
68.          ( *treePtr )->leftPtr = NULL;
69.          ( *treePtr )->rightPtr = NULL;
70.       } /* end if */
71.       else {
72.          printf( "%d not inserted. No memory available.\n", value );
73.       } /* end else */
74.
75.    } /* end if */
76.    else { /* tree is not empty */
77.
78.       /* data to insert is less than data in current node */
79.       if ( value < ( *treePtr )->data ) {
80.          insertNode( &( ( *treePtr )->leftPtr ), value );
81.       } /* end if */
```

Зангилааг модонд оруулахын тулд эхлээд түүнд санах ой хуваарилах ёстой

Хэрэв оруулсан зангилааны өгөгдөл тухайн зангилаанаас бага бол програм түүнийг зүүн хүү зангилаанд оруулах гэж оролдоно

```
82.
83.       /* data to insert is greater than data in current node */
84.       else if ( value > ( *treePtr )->data ) {
85.          insertNode( &( ( *treePtr )->rightPtr ), value );
86.       } /* end else if */
87.       else { /* duplicate data value ignored */
88.          printf( "dup" );
89.       } /* end else */
90.
91.    } /* end else */
92.
93. } /* end function insertNode */
94.
95. /* begin inorder traversal of tree */
96. void inOrder( TreeNodePtr treePtr )
97. {
98.
99.    /* if tree is not empty then traverse */
100.   if ( treePtr != NULL ) {
101.      inOrder( treePtr->leftPtr );
102.      printf( "%3d", treePtr->data );
103.      inOrder( treePtr->rightPtr );
104.   } /* end if */
105.
106.} /* end function inOrder */
107.
108./* begin preorder traversal of tree */
109.void preOrder( TreeNodePtr treePtr )
110.{
111.
```

Хэрэв оруулсан зангилааны өгөгдөл тухайн зангилаанаас их бол програм түүнийг баруун хүү зангилаанд оруулах гэж оролдоно

Inorder нэвтрэлт inorder нэтрэлтийг зүүн дэд модон дээр дуудаад, дараа нь зангилааг өөрийг нь хэвлээд, эцэст нь inorder нэтрэлтийг баруун дэд модон дээр дуудна.

Модны жишээ…

```
112./* if tree is not empty then traverse */
113.   if ( treePtr != NULL ) {
114.      printf( "%3d", treePtr->data );
115.      preOrder( treePtr->leftPtr );
116.      preOrder( treePtr->rightPtr );
117.   } /* end if */
118.
119.} /* end function preOrder */
120.
121./* begin postorder traversal of tree */
122.void postOrder( TreeNodePtr treePtr )
123.{
124.
125.   /* if tree is not empty then traverse */
126.   if ( treePtr != NULL ) {
127.      postOrder( treePtr->leftPtr );
128.      postOrder( treePtr->rightPtr );
129.      printf( "%3d", treePtr->data );
130.   } /* end if */
131.
132.} /* end function postOrder */
```

Preorder нэвтрэлт зангилааг өөрийг нь хэвлээд, дараа нь preorder нэтрэлтийг зүүн дэд модон дээр дуудаад, эцэст нь preorder нэтрэлтийг баруун дэд модон дээр дуудна.

Postorder нэвтрэлт postorder нэтрэлтийг зүүн дэд модон дээр дуудаад, дараа нь postorder нэтрэлтийг баруун дэд модон дээр дуудаад, эцэст нь зангилааг өөрийг нь хэвлэнэ

Модны жишээ…

```
The numbers being placed in the tree are:
  6  7  4 12  7dup  2  2dup  5  7dup 11

The preorder traversal is:
  6  4  2  5  7 12 11

The inorder traversal is:
  2  4  5  6  7 11 12

The postorder traversal is:
  2  5  4 11 12  7  6
```

Дүгнэлт

- Санах ойн динамик хаваарилалт
  - `malloc()`
  - `free()`
- Өгөгдлийн 4 бүтэц

Дан холбоост жагсаалт
- Insert: O(1)
- Delete: O(1)
- Search: O(N)

Стек
- Push: O(1)
- Pop: O(1)
- isEmpty: O(1)

Дараалал
- Enqueue: O(1)
- Dequeue: O(1)
- isEmpty: O(1)

Хоёртын хайлтын мод
- Traversal: O(N)
- Insert: O(h)
- Delete: O(h)

h: модны өндөр
Дундаж тохиолдол: O(logN)
Муу тохиолдол: O(N)