Бағдарламалық жасақтаманың жалпы құру түсініктемесі

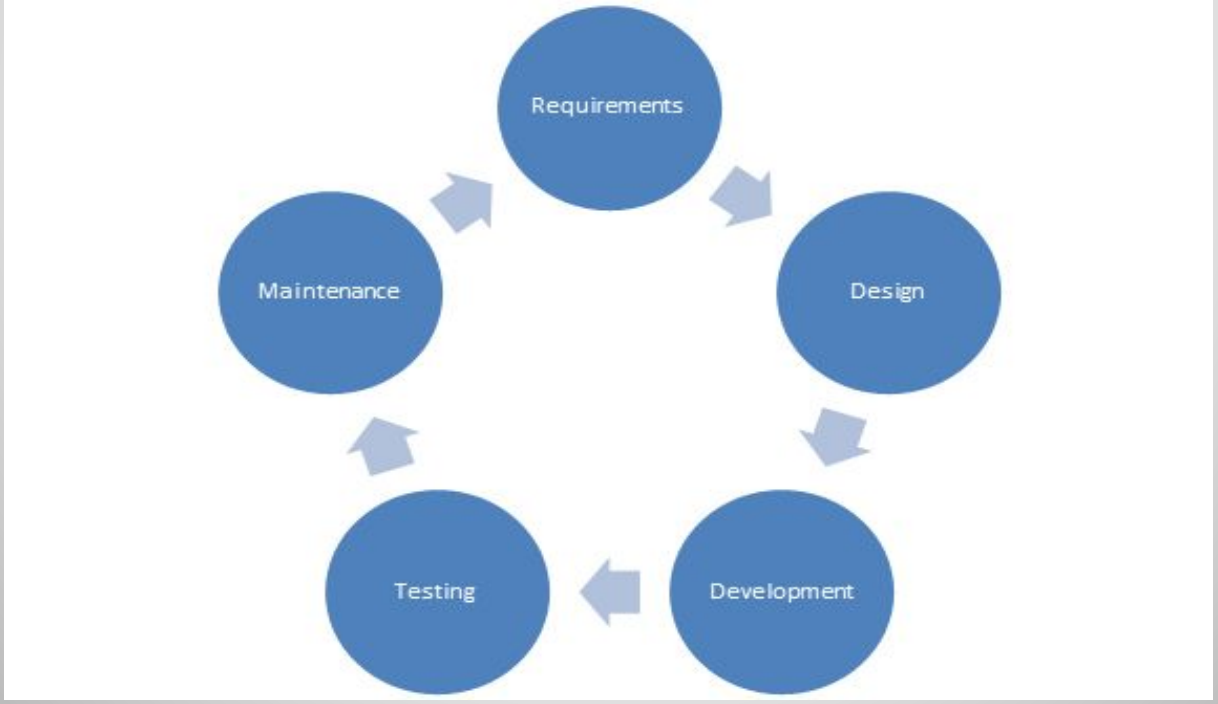
Лекция 13

Объективті домен матрицасы

Дағдылар/түсініктер	MTA емтиханының мақсаты
Қолданбаны түсіну Өмір циклін басқару	Қолданбаның өмірлік циклын басқаруды түсіну (3.1)
Тестілеуді түсіну	Қолданбаның өмірлік циклын басқаруды түсіну (3.1)
Деректер құрылымын түсіну	Алгоритмдер мен деректер құрылымын түсіну (3.3)
Сұрыптау алгоритмдерін түсіну	Алгоритмдер мен деректер құрылымын түсіну (3.3)

Қолданбаның өмірлік циклін басқару (ҚӨБ)

- Бағдарламалық жасақтаманың өмірлік циклін басқару (ҚӨБ) - бұл жаңа бағдарламалық жасақтама өнімінің пайда болуынан бастап өнімнің дайын болғанына дейінгі айналатын әрекеттер жиынтығы.



Талаптар

- Талаптарды талдау - бұл жаңа бағдарламалық жасақтама жүйесіне арналған егжей-тегжейлі талаптарды анықтау процесі.
- Бизнес-аналитик - бизнес қажеттіліктерін талдап, талаптарды орындайтын разработчиктерге айналдыру үшін жауап береді.

Дизайн

- Дизайн әрекеті бағдарламалық жасақтаманы қалай жүзеге асырудың жоспарларын, үлгілерін және архитектурасын құру үшін қолданылады.
- Қатысушылар:
  - Архитектор
  - Пайдаланушы тәжірибесіндегі дизайнер

Даму

- Бағдарламалық жасақтаманы әзірлеу қызметі бағдарламалық қамтама кодын, мәліметтер базасын және басқа да тиісті мазмұнды құру арқылы дизайнды іске асыруды қамтиды.
- Қатысушылар:
  - Разработчиктер
  - Мәліметтер базасының администраторы (МБА)
  - Техникалық жазушылар
  - Мазмұн әзірлеушілер

Тестілеу

- Тестілеу дайын өнімнің сапасын қамтамасыз ету үшін қолданылады.
- Талаптар құжатында көрсетілген жүйелік күтулер мен жүйенің нақты әрекеті арасындағы мүмкін болатын алшақтықты анықтайды.
- Қатысушылар:
  - Тестерлер

Тестілеуді түсіну

- Бағдарламалық жасақтаманы тестілеу бұл бағдарламалық жасақтаманың оның талаптарына сәйкестігін тексеру процесі.
- Бағдарламалық жасақтаманы тестілеу тек ақауларды табуға көмектеседі - ол кемшіліктердің болмауына кепілдік бере алмайды.
- Өнімді дамыту циклiндегі кемшіліктерді ертерек табу әлдеқайда тиімді.

Тестілеу әдістері

- Қара қорап тесті
  - Тек кіру және шығу параметрлеріне назар аудару.
  - Ішкі жүйелік жұмыс туралы кез-келген білім тестілеу үшін пайдаланылмайды.
  - Бағдарламалық жасақтама оның барлық талаптарын қанағаттандыратындығына көз жеткізу үшін қолданылады.
- Ақ қорап тесті
  - Тестерлер жүйені тестілеу кезінде жүйенің ішкі білімдерін қолданады.
  - Әр әдіс немесе функция тиісті сынақ жағдайларында жұмыс істейтініне көз жеткізу үшін қолданылады.

Тест деңгейлері

- Модульдік тестілеу
  - Код бірлігінің функционалдығын тексереді.
- Интеграциялық тестілеу
  - Бағдарламалық жасақтама компоненттері арасындағы интерфейсті бағалайды.
- Жүйелік тестілеу
  - Бағдарламалық жасақтаманы жалпы тестілеу.
- Регрессиялық тестілеу
  - Әрбір жаңа түзету бұрын жұмыс істеп тұрған бағдарламаны бұзбайтынына көз жеткізеді.

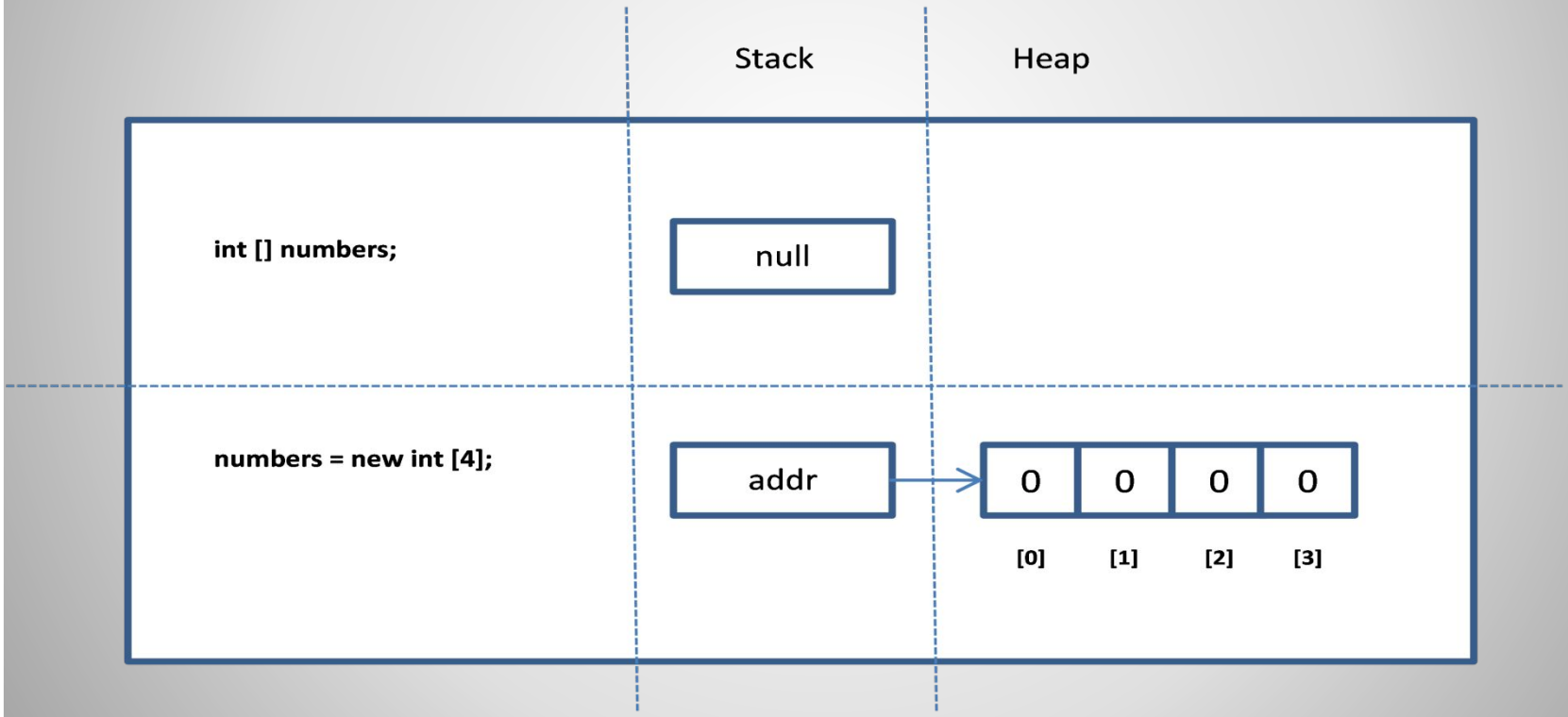
Деректер құрылымын түсіну

- Деректер құрылымы дегеніміз - бұл ақпаратты компьютердің жадында сақтау және сақтау әдістері.
- Деректер құрылымын түсіну дегеніміз
  - Сақтау үлгісін түсіну,
  - Деректер құрылымын құру, оған қол жеткізу және оны басқару үшін қандай әдістер қолданылатынын түсіну.
- Жалпы мәліметтер құрылымы:
  - Массивтер
  - Кезектер
  - Стектер
  - Байланыстырылған тізімдер

Массивтер

- Массив дегеніміз - бір типтегі заттар жиынтығы.
- Массивтің элементтері жадыда аралас сақталады.
- Массивтің өлшемі алдын-ала анықталған және бекітілген.
- Кез-келген массив элементіне индекстің көмегімен тікелей қатынасуға болады.
- C # массивінің индексі нөлге негізделген.

Массив - ішкі көрінісі



Массив - Жалпы операциялар

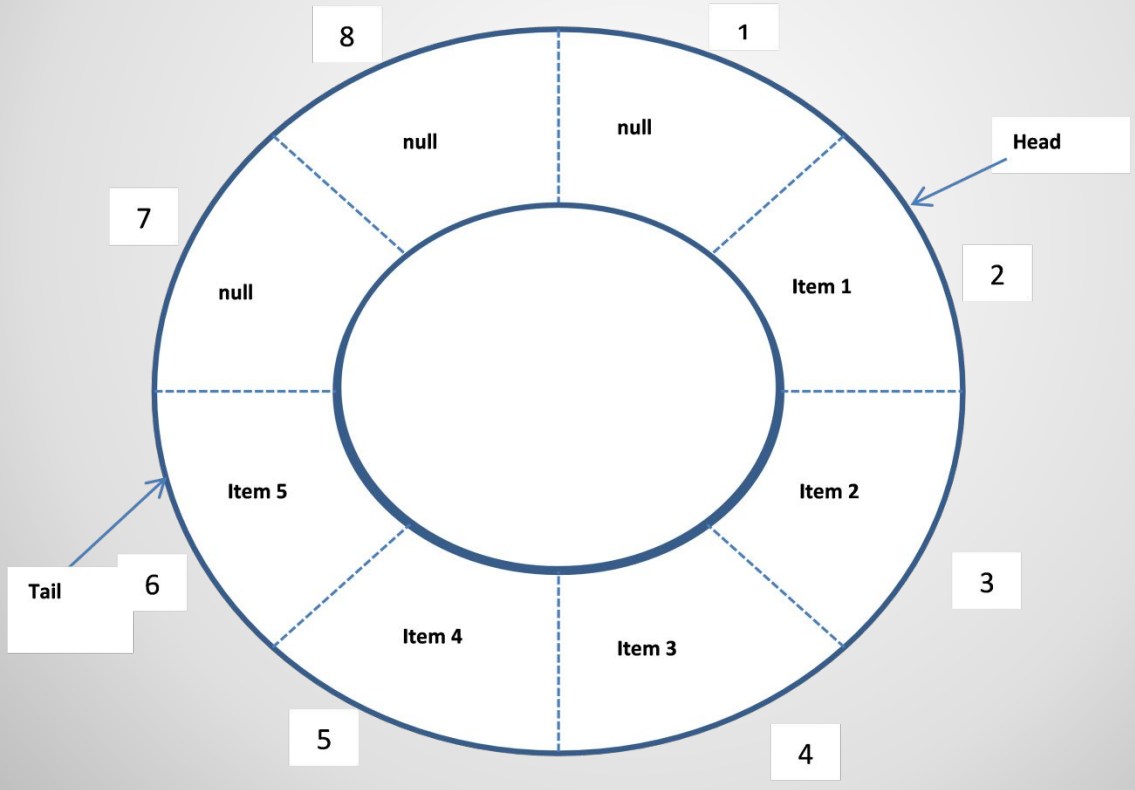
- Массивтер келесі операцияларды қолдайды:
  - Үлестіру
  - Қол жеткізу
- Келесі код массивтің төртінші элементіне 10 мәнін тағайындайды, содан кейін оның екі есесі calc айнымалысына меншіктеледі:

```
number[3] = 10;
int calc = number[3] * 2;
```

Кезектер

- Жинаққа бірінші қосылған элементтер бірінші алынып тасталынады.
- Бірінші кірген бірінші шығады (FIFO).
- Кезек - бұл гетерогенді мәліметтер құрылымы.
- Кезектің сыйымдылығы - кезекте тұра алатын элементтер саны.
- Кезекке элементтер қосылғанда, сыйымдылық автоматты түрде артады.

Кезектер - ішкі көрінісі



Кезектер - жалпы операциялар

- **Enqueue**: Кезектің соңына элемент қосады.
- **Dequeue**: кезек басында тұрған элементті жояды.
- **Peek**: ағымдағы элементті кезектен шығармай-ақ алу.
- **Contains**: белгілі бір элементтің кезекте тұрғанын немесе жоқтығын анықтайды.

Стектер

- Жинаққа соңғы қосылған элемент бірінші бор алынып тасталады.
- Соңғ кірген бірінші шығады (LIFO).
- Стек - бұл гетерогенді мәліметтер құрылымы.
- Стектің сыйымдылығы - кезекте тұра алатын элементтер саны.
- Стекке элементтер қосылғанда, сыйымдылық автоматты түрде артады.

Стектер - ішкі көрінісі

- Стек кезек тәрізді бейнеленуі мүмкін, тек стектің соңы жоғарғы жағы деп атайды, ал басын төменгі жағы деп атайды.
- Жаңа элементтер әрқашан жинақтың соңына қосылады; бұл болған кезде, ең жоғарғысы боп жаңадан қосылған элементті көрсете бастайды.
- Элементтер стекте жоғарғы жағынан алынып тасталады, ал келесі жоғарғы элемент жинақтағы келесі элементті көрсету үшін реттеледі.

Стек - жалпы операциялар

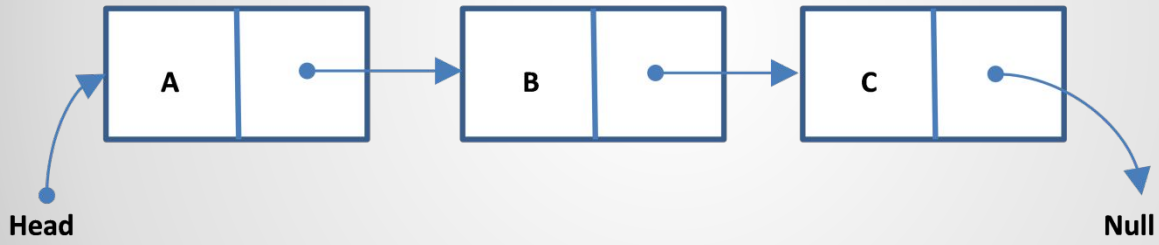
- **Push:** стектің соңына элемент қосу.
- **Pop:** Жоғарыдағы элементті өшіру.
- **Peek:** Жоғарыда тұрған элементті жоймай стектен алу.
- **Contains:** Берілген айнымалы стекта бар ма анықтау.

Байланыстырылған тізімдер(Linked List)

- Байланыстырылған тізім дегеніміз - әрбір түйінде тізбектегі келесі түйінге сілтеме болатындай етіп орналастырылған түйіндер жиынтығы.
- Байланыстырылған тізімдегі әрбір түйін екі ақпараттан тұрады:
  - түйінге сәйкес келетін мәліметтер
  - келесі түйінге сілтеме

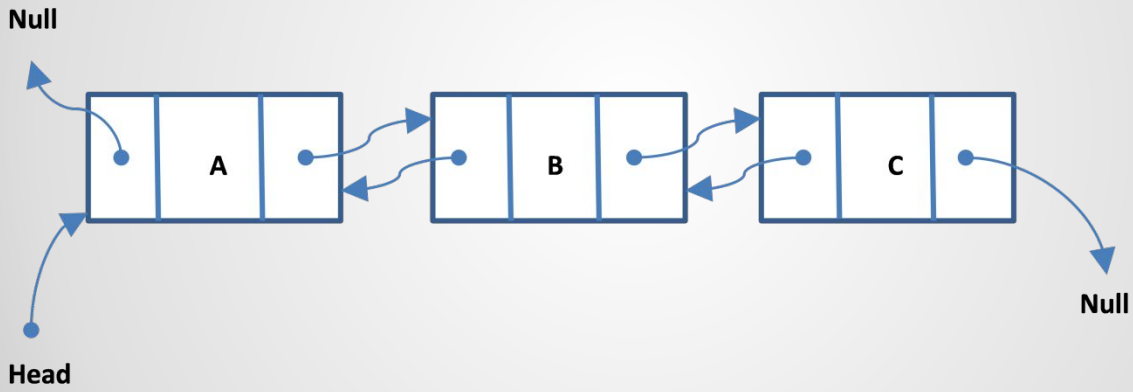
Linked Lists – ішкі көрінісі

- Бір жақты байланысқан тізім



Linked Lists – ішкі көрінісі

- Екі жақты байланысқан тізім

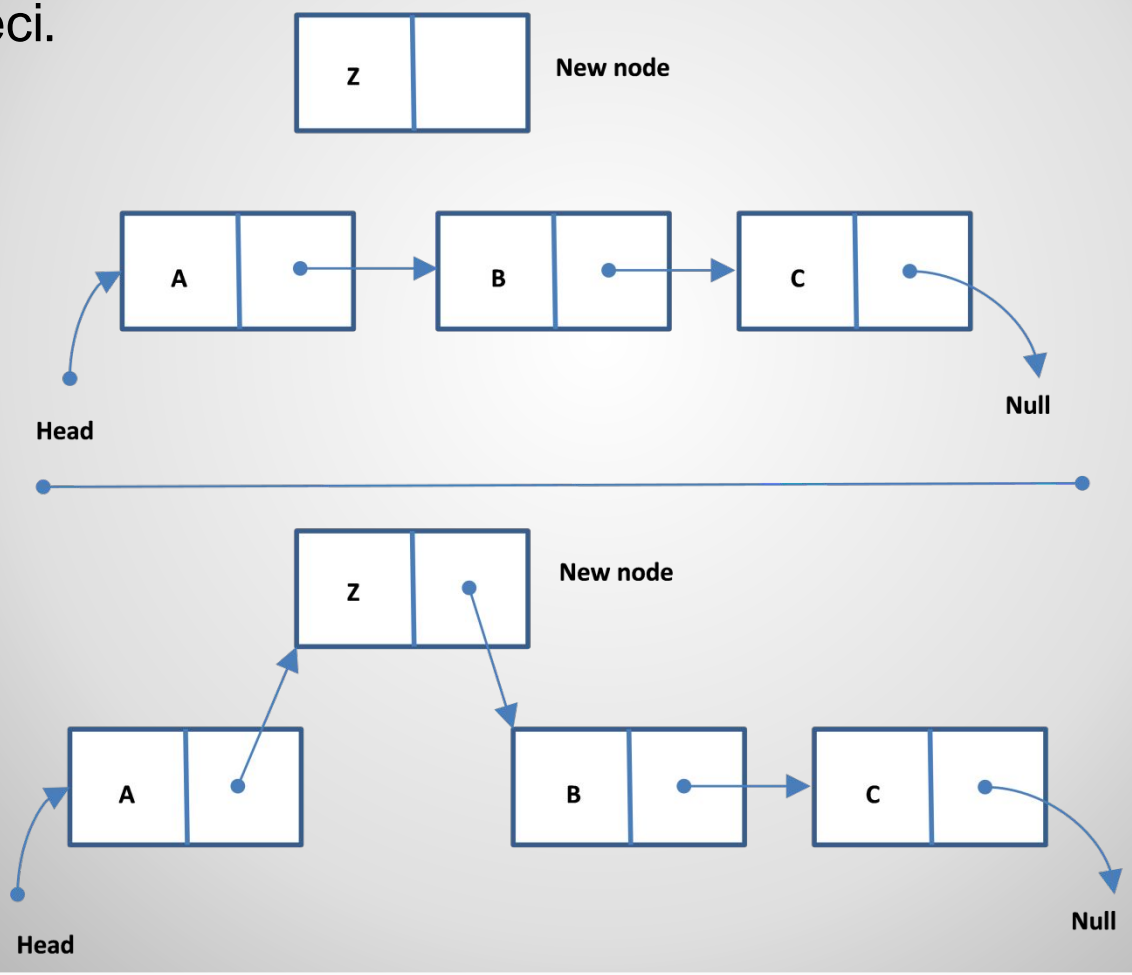


Linked Lists – жалпы операциялар

- **Add:** байланысқан тізімге түйін қосу.
- **Remove:** Берілген түйінді өшіру.
- **Find:** Берілген айнымалыны байланысқан тізімнің түйінен табу.

Linked Lists – қосу әрекетін визуализациялау

- Байланыстырылған тізімге элемент қосу - сілтемелерді өзгерту мәселесі.



Сұрыптау алгоритмдерін түсіну

- Сұрыптау алгоритмдері - бұл тізімдегі элементтерді белгілі бір ретпен реттейтін алгоритмдер.
- Сұрыптау алгоритмдерін түсіну мәселені шешудің әртүрлі әдістерін түсінуге, талдауға және салыстыруға көмектеседі.
- көптеген сұрыптау алгоритмдеріне мысал:
  - BubbleSort - көпіршікті сұрыптау
  - QuickSort.- жылдам сұрыптау

BubbleSort

- **BubbleSort** екі элементті салыстыру арқылы жұмыс істейді, тексергенде рет бұзылса ауыстырады. Алгоритм мұны барлық тізім қажетті ретке келгенше жалғастырады.
- **BubbleSort** өз атауын алгоритмнің жұмыс істеу тәсілінен алады: алгоритмде кіші элементтер көпіршік сияқты төбеге шығады.

BubbleSort алгоритмінің демонстрациясы

Step	*Before*	*After*	*Comments*
1	**20**, **30**, 10, 40	**20**, **30**, 10, 40	The algorithm compares the first two elements (20 and 30); because they are in the correct order, no swap is needed.
2	20, **30**, **10**, 40	20, **10**, **30**, 40	The algorithm compares the next two elements (30 and 10); because they are out of order, the elements are swapped.
3	20, 10, **30**, **40**	20, 10, **30**, **40**	The algorithm compares the next two elements (30 and 40); because they are in the correct order, no swap is needed.

BubbleSort алгоритмінің демонстрациясы

Step	*Before*	*After*	*Comments*
1	**20**, **10**, 30, 40	**10**, **20**, 30, 40	The algorithm compares the first two elements (20 and 10); because they are out of order, the elements are swapped.
2	10, **20**, **30**, 40	10, **20**, **30**, 40	The algorithm compares the next two elements (20 and 30); because they are in the correct order, no swap is needed.
3	10, 20, **30**, **40**	10, 20, **30**, **40**	The algorithm compares the next two elements (30 and 40); because they are in the correct order, no swap is needed.

BubbleSort алгоритмінің демонстрациясы

Step	*Before*	*After*	*Comments*
1	**10**, **20**, 30, 40	**10**, **20**, 30, 40	The algorithm compares the first two elements (10 and 20); because they are in the correct order, no swap is needed.
2	10, **20**, **30**, 40	10, **20**, **30**, 40	The algorithm compares the next two elements (20 and 30); because they are in the correct order, no swap is needed.
3	10, 20, **30**, **40**	10, 20, **30**, **40**	The algorithm compares the next two elements (30 and 40); because they are in the correct order, no swap is needed.

Жылдам сұрыптау

- **QuickSort** алгоритмі бөліп алу-және-басқару техникасын мәліметтер сұрыпталып болғанша, бөліп алып реттеуді жасайды.
- **QuickSort BubbleSort алгоритміне қарағанда тезірек жұмыс жасайды**.

QuickSort алгоритмін демонстрациялау

Step	*Data to be sorted*	*Comments*
1	50, 10, **30**, 20, 40	Start with an unsorted list and pick a pivot element—in this case 30.
2	20, **10** | 30 | 50, **40**	Partition the list, with items less than the pivot going to the left list and items greater than the pivot going to the right list. Then, to sort the left list, pick a pivot (here, 10). Similarly, to sort the right list, pick a pivot (here, 40) for that list.
3	- | 10 | 20 | 30 | - | 40 | 50	In the left list, 20 is greater than 10, and in the right list, 50 is greater than 40; therefore, both 20 and 50 go into the right list. This yields lists of only one number, which are all by definition sorted.
4	10, 20, 30, 40, 50	All the small sorted lists are merged to create the final complete sorted list.

Қайталау

- Қосымшаның өмірлік циклын басқару:
  - Талаптар талдау
  - Бағдарламалық жасақтама жасау
  - Тестілеу
  - Шығаруды басқару
- Тестілеу әдістері:
  - “Қара қорап” тесті
  - “Ақ қорап” тесті

Қайталау

- Тест деңгейлері
  - Модульдік тестілеу, интеграциялық тестілеу, жүйелік тестілеу, регрессиялық тестілеу
- Деректер құрылымы
  - Массивтер
  - Кезектер
  - Стектер
  - Linked Lists
- Сұрыптау алгоритмдері: BubbleSort,