Жамбыл облысы әкімдігі білім басқармасы

«Білім» кәсіби гуманитарлық-техникалық колледжі

(білім беру ұйымының атауы)

**Оқу сабағының жоспары**

(теориялық немесе өндірістік оқыту)

Программалық өнімдердің технологияларын қамтамасыз ету әдістері.

(сабақ тақырыбы)

**Модуль/Пән атауы** Ішкі тапсырмалардың өзара әрекеттесу сызбасын жасап, оларды бір тапсырмаға әзірлеу

**Дайындаған педагог** Нургисаева У.М

**20\_**25**\_ жылғы** «\_17\_\_\_» \_\_ақпан \_\_\_\_

**1. Жалпы мәліметтер**

Курс, оқу жылы, топ 2 курс, 2БҚ-23

Сабақ түрі: Білім, іскерлікті қалыптастыру

**2. Мақсаты, міндеттері:**

**Оқу:** Программалық өнімдердің өмірлік циклінің кезеңдерін, жүйелік архитектураларды, өңдеу әдістерін, сондай-ақ бағдарламалық өнімдерге арналған қауіпсіздік стандарттары мен оңтайландыру әдістерін білу

**Дамыту:** Бағдарламалау тілдерінде код жазу, жүйелерді жобалау, тестілеу мен оңтайландыру, жоба құру.

**Тәрбиелік:** Ұқыптылық пен жауапкершілікті қалыптастыру.

**3. Оқу-жаттығу процесінде білім алушылар меңгеретін күтілетін нәтижелер және кәсіби дағдылар тізбесі:** Студенттер нақты жобалар бойынша жұмыс істеп, түрлі құралдар мен технологиялар арқылы бағдарламалық өнімдерді жасау, тестілеу, бағалау және пайдаланушы талаптарына сәйкестігін тексеру дағдыларын игереді.

**4. Қажетті ресурстар:** ДК немесе ноутбуктер

**Токеймбетов Б.Т., Қасымбеков А.Б.** – Цифрлық құрылғылар және микропроцессорлар негіздері. Алматы: Қазақ университеті, 2018.

[***https://libr.aues.kz/facultet/frts/kaf\_e/2/umm/e\_3.htm***](https://libr.aues.kz/facultet/frts/kaf_e/2/umm/e_3.htm)

***5. Сабақтың барысы: (90 минут)***

**5.1. Ұйымдастыру кезеңі:** *( 3 мин )*

**5.2. Үй жұмысын жан-жақты тексеру:**

***“Миға шабуыл ” әдісі*** *(15 минут)*

**Программалық өнімдердің технологияларын қамтамасыз ету әдістері.**

Программалық өнімдерді технологиялық қамтамасыз ету – бағдарламалық жүйені жобалау, әзірлеу және енгізу кезінде қолданылатын әдістер, құралдар мен стандарттардың жиынтығы. Бұл процесс бірнеше негізгі аспектілерді қамтиды:

1. **Әдіснамалық қамтамасыз ету** – программалық өнімдерді әзірлеудің теориялық және әдістемелік негіздері.
2. **Технологиялық қамтамасыз ету** – бағдарламаны әзірлеу және енгізу технологиялары, құралдары.
3. **Техникалық қамтамасыз ету** – аппараттық ресурстар мен инфрақұрылым.
4. **Ақпараттық қамтамасыз ету** – деректер қоры, құжаттар мен стандарттар.
5. **Құқықтық қамтамасыз ету** – лицензиялау, авторлық құқықтарды қорғау.

Бұл компоненттер программалық өнімнің сапасын, қауіпсіздігін және сенімділігін қамтамасыз ету үшін маңызды рөл атқарады.

**2. Программалық өнімдерді технологиялық қамтамасыз етудің негізгі әдістері**

Программалық өнімді тиімді әзірлеу үшін әртүрлі әдістер қолданылады. Олардың ішіндегі негізгі бағыттар:

**2.1. Бағдарламалық қамтамасыз етуді әзірлеу әдістері**

Программалық өнімдерді әзірлеу барысында әртүрлі модельдер қолданылады:

* **Каскадтық модель (Waterfall Model)** – программалық өнім кезең-кезеңімен жасалады (талаптарды анықтау → жобалау → жүзеге асыру → тестілеу → енгізу → қызмет көрсету).
* **Спиральдық модель (Spiral Model)** – тәуекелдерді басқаруға бағытталған, итерациялық даму процесі бар модель.
* **Agile (Agile Development)** – программалық өнімді итеративті түрде жасауға мүмкіндік береді, икемді және жылдам даму процесін қамтамасыз етеді.
* **DevOps (Development and Operations)** – программалық өнімнің үздіксіз интеграциясы мен жеткізілуін қамтамасыз ететін әдіс.

Әр әдістің өзіндік ерекшеліктері мен артықшылықтары бар, сондықтан оларды жобаның ерекшеліктеріне қарай таңдайды.

**2.2. Код сапасын қамтамасыз ету әдістері**

Программалық өнімнің сенімділігін арттыру үшін код сапасын бақылау әдістері қолданылады:

* **Кодты шолу (Code Review)** – әріптестермен кодты тексеру арқылы қателерді анықтау және түзету.
* **Автоматтандырылған тестілеу (Automated Testing)** – тест сценарийлері арқылы бағдарламаның тұрақтылығын тексеру.
* **CI/CD (Continuous Integration / Continuous Deployment)** – программалық өнімді үздіксіз тестілеу және жаңартуды қамтамасыз ететін әдіс.
* **Статикалық код анализі** – кодтың сапасын талдау үшін арнайы құралдарды пайдалану (мысалы, SonarQube, ESLint, Pylint).

**2.3. Программалық өнімдердің сенімділігі мен қауіпсіздігін қамтамасыз ету**

Программалық өнімді сенімді және қауіпсіз ету үшін келесі әдістер қолданылады:

* **Шифрлеу және аутентификация** – мәліметтердің қауіпсіздігін қамтамасыз ету үшін деректерді шифрлеу, кіруді бақылау жүйелерін енгізу.
* **Киберқауіпсіздік әдістері** – SQL-инъекциялардан, XSS (Cross-Site Scripting), CSRF (Cross-Site Request Forgery) шабуылдарынан қорғау.
* **Қолданушы рөлдерін басқару** – әрбір қолданушыға қажетті рұқсат деңгейін белгілеу.
* **Жүйенің төзімділігін арттыру** – резервтік көшірмелер жасау, жүйенің жұмысын қалпына келтіру стратегияларын әзірлеу

**2.4. Бағдарламалық өнімдерді тестілеу әдістері**

Тестілеу программалық өнімнің ақауларын анықтауға және оның сапасын арттыруға көмектеседі. Тестілеу әдістері:

* **Қолмен тестілеу (Manual Testing)** – тестілеушілер программалық өнімді өз қолдарымен тексереді.
* **Автоматтандырылған тестілеу (Automated Testing)** – Selenium, JUnit, TestNG сияқты құралдарды пайдалану арқылы орындалады.
* **Функционалдық тестілеу** – программаның негізгі мүмкіндіктерін тексеру.
* **Жүктеме тестілеу (Load Testing)** – жүйенің жоғары жүктемеге қаншалықты төзімді екенін анықтау.
* **Қауіпсіздік тестілеуі** – программаның осал тұстарын анықтау үшін жүргізіледі.

Тестілеу программалық өнімді әзірлеудің ажырамас бөлігі болып табылады және оны тұрақты түрде жүргізу қажет.

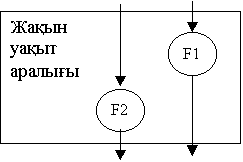
**3. Программалық өнімдерді қамтамасыз етудің заманауи технологиялары**

Қазіргі таңда программалық өнімдердің технологиялық қамтамасыз етілуін жақсарту үшін келесі заманауи әдістер мен құралдар қолданылады:

* **Cloud Computing (Бұлттық есептеулер)** – программалық өнімдерді бұлттық серверлер арқылы әзірлеу және тарату (AWS, Azure, Google Cloud).
* **Microservices Architecture** – жүйені шағын қызметтерге бөлу арқылы икемділік пен тұрақтылықты қамтамасыз ету.
* **Containerization (Docker, Kubernetes)** – бағдарламалық өнімдерді оңай орналастыру және масштабтау.
* **AI & Machine Learning** – жасанды интеллектті программалық өнімдердің қауіпсіздігі мен функционалдығын жақсарту үшін қолдану.

Функциялар немесе мәліметтердің процедуралық байланысы бір үрдістің бөлігі болып табылады. Егер функциялар модельдік программаны альтернативті бөлігі түрінде біріктірсе, онда  модельдегі функцияларды процедуралық байланысы түрінде қарастыруға болады. Барлық әрекеттер жалпы үрдісте ғана орындалатын болғандықтан кейбір элементтердің әлсіз байланысы процедуралық байланыстың технологиялық мүмкіндігі жоғарыда айтылған бөлімдерге қарағанда кең болады.

|  |
| --- |
|  |
|  | https://libr.aues.kz/facultet/tef/kaf_ik/31/umm/ik_77.files/image063.gif |

Егер функциялар белгілі уақыт аралығында параллель орындалса, онда бұндай функциялар байланысын уақытаралық байланыс деп атайды. Мәліметтердің уақытаралық байланыс түріне белгілі уақыт аралығында мәліметтердің қолданылуын қарастыруға болады. Мәліметтердің форматы әр түрлі болуына байланысты, функциялардың есептеуі мен көптігіне байланысты осы модульдік технологиялық мүмкіндігі жоғарғы бөлімдерге қарағанда кеми түседі.

Мәліметтердің немесе функциялардың бір топқа логикалық бірігуін логикалық байланыс деп атайды. Мысалы: Бір типтегі мәліметтерді немесе мәтіндік ақпараттарды өңдеуге арналған функцияларды қарастыруға болады. Уақытаралық байланысы бар модульдерге қарағанда логикалық байланысы бар модельдер  анағұрлым күрделі, технологияны білу мүмкіндігі кең болады.

Егер элементтер арасындағы байланыс аз немесе жоқ болатын болса, онда бұндай байланыс түрін кездейсоқ байланыс деп атайды. Элементтердің арасындағы байланыс өте аз болғандықтан кездейсоқ байланыстың технологиялық  көрсеткіші ішіндегі ең кемі болып табылады.

**Ресурстар библиотекасы**

Ресурстар библиотекасының екі түрі бар:

а) ішкі;

б) кластар библиотекасы.

Жақын тағайындалған іс-әрекеттердің  функцияларының программалар-ының жиынтығы ресурстар библиотекасын құрайды. Осы библиотекадағы программалардың бір–бірімен байланысы логикалық та, ал программалардың өзіндегі байланыс функционалдық болады. Жақын тағайындалған кластар жиынтығы кластар библиотекасын құрайды. Кластағы элементтер арасындағы байланыс ақпараттық та, ал кластар арасындағы байланыс функционалдық болады.

**Шығатын ( восходящий ) және шықпайтын ( неисходящий )**

Жобаны жобалау, енгізу және ирархиялық сұлба бойынша тестілеу декомпозиция көмегімен алу кездерінде екі тәсілді қолданады:

а) шығатын;

б) шыпайтын.

Шығатын әдіс бойынша алдымен төменгі деңгейлі компоненттерді жобалап енгізеді. Содан кейін ғана келесі деңгейді жобалауға кіріседі. Барлық компоненттер тестіден өткеннен кейін ғана жалпы жобаны жинауға болады. Осы тәсілде төменгі деңгейдегі компоненттерді компоненттер библиотекасына енгізу жиі қолданылады.

Жоғарыдан төмен қарай жобалауды немесе алдымен жоғарғы деңгейді, содан кейін ирархиялық саты бойынша біртіндеп қолданады. Осы үрдіс әрбір деңгейдегі құрылған программаны тестілеу түріндегі тексеруге жіберу үшін төменгі деңгейдегі құрылмаған программаны қолданбау үшін арнайы бітегіш модульдерді қолданады.

Осы әдістен келесі тәсілдерді қолдануға болады:

а) ирархиялық тәсіл. Программаны құруды, деңгейлік режимді қатаң сақтауды ұсынады. Осы тәсілдің негізгі өзекті мәселесі - бітегіштің көптігі;

б) операциялық тәсіл. Модульдерді құруды, программаны жұмыс істеу тәртібінің сұлбасы бойынша іске асыруды ұсынады. Негізгі қиыншылығы  - модульдердің жұмыс істеу реті мәліметтерге байланысты болуы мүмкін. Мәліметтерді шығаруға арналған модульдер ең төменгі деңгей болуына қарамастан күрделі бітегіштерді қолданбау үшін алғашқылардың  бірі болып құрылуы керек.  Осы тәсілді қолдану кезінде көптеген негізгі модульдер жоба соңында құрылатын болғандықтан адам ресурстарын әр деңгейге бөлуге қиындықтар туғызады;

в) Комбинирленген  тәсіл. Комбинирленген тәсіл құруға әсер ететін келес факторлармен санасады:

1)     модульдер жетістігі – барлық  модульдер ішінен керекті модульді берілген ретімен шақыру;

2)     мәліметтерге байланыстылығы – мәліметтерді құратын модульдер өңдеуге арналған модульдерден бұрын ұйымдастыру;

3)      нәтижесін беру мүмкіндігін қамтамасыз ету – мәліметтерді беруге арналған модульдерді өңдеуге арналған модульдерден бұрын ұйымдастыру;

4)     көмекші модульдердің дайындығы – көмекші модульдерді өңдеуге арналған модульдерден бұрын ұйымдастыру;

5)     керекті ресурстардың болуы.

Шықпайтын әдіс келісім бойынша шықпайтын тізбектелген компоненттерді құруда тәртіп сақталмауы мүмкін. Осы әдісті көбінесе нысанды бағытталған программада қолданады.

Шықпайтын әдістің қасиеттері:

а) жобаланатын компоненттердің спецификациясын толық анықтайды және компоненттер арасындағы байланыстарын қамтамасыз етеді;

б) қолданушыға арналған алдын ала анықталған интерфейс көмегімен тұтынушыға демонстрация (көрсете) жасай отырып құрылатын программаға керекті түзетулерді қамтамасыз етеді;

в) тестілеу және комплексті отладка жасау мүмкіндігін қажет етеді.

**Құрылымды және құрылымды емес программалау**

Технологиясы жоғарғы деңгейдегі программа құрудың тәсілдерінің бірі – құрылымды программалау. Құрылымды программалаудың үш түрі бар:

а) сызықты құрылымды;

б) тармақталған құрылымды;

в) циклды құрылымды.

|  |  |  |
| --- | --- | --- |
| **Білдім** | **Білемін** | **Білгім келеді** |
|  |  |  |

**Рефлексия**

**Үй тапсырмасы –** Зертханалық жұмыс №3 орындау және қорғау

**1. Екі санның қосындысын табу:**

a = float(input("Бірінші санды енгізіңіз: "))

b = float(input("Екінші санды енгізіңіз: "))

sum\_result = a + b

print("Қосындысы:", sum\_result)

**2. Есептеуішті қолдану:**

num = float(input("Санды енгізіңіз: "))

# Квадрат және кубты есептеу

square = num \*\* 2

cube = num \*\* 3

# Нәтижелерді шығару

print(f"{num} санының квадраты: {square}")

print(f"{num} санының кубы: {cube}")

**3. Сандарды салыстыру:**

a = float(input("Бірінші санды енгізіңіз: "))

b = float(input("Екінші санды енгізіңіз: "))

# Үлкен санды табу

if a > b:

print(f"{a} сан {b} саннан үлкен.")

elif a < b:

print(f"{b} сан {a} саннан үлкен.")

else:

print("Сандар тең.")

**4. Тізімдегі сандардың орташа мәнін табу:**

# Сандарды енгізу

numbers = input("Сандарды енгізіңіз, араларын бос орынмен бөліңіз: ").split()

# Сандарды тізімге түрлендіру

numbers = [float(num) for num in numbers]

# Орташа мәнді есептеу

average = sum(numbers) / len(numbers)

# Нәтижені шығару

print("Сандардың орташа мәні:", average)