Лекция № 12. Бағдарламалық қамтамасыз етудің сенімділігін арттыру әдістері

Лекция мазмұны: Бағдарламалық қамтамасыз етудің сенімділігін арттырудың әртүрлі аспектілері қарастырылады.

Лекция мақсаты: бағдарламалық қамтамасыз ету сенімділігін арттырудың негізгі әдістеріне оқыту.

12.1. Орталық басқару процессорының бағдарламалық қамтамасыз етуінің сенімділігін арттыру

Әзірленген жүйелердің үнемі күрделенуіне, оларға жүктелетін міндеттер ауқымының кеңеюіне, тиісінше, олардың көлемі мен күрделілігінің айтарлықтай артуына байланысты бағдарламалық қамтамасыз етудің сенімділік мәселесі барған сайын маңызды болып отыр. Бағдарламалық қамтамасыз етудің нақты сенімділігі көбінесе аппараттық құралдың сенімділігінен төмен. Микропроцессорлық және компьютерлік жүйелердің жұмысы тек аппараттық және бағдарламалық құралдардың бір мезгілде жұмыс істеуімен мүмкін болады. Бағдарламалық қамтамасыз етудің сенімділігі әдетте әзірлеу кезінде немесе жұмыс кезінде енгізілген бағдарламаларда әртүрлі қателердің болуымен анықталады. Қате деп біз бағдарламаның белгіленген функцияларды орындай алмауын түсінеміз. Қатенің пайда болуы - бұл бағдарламаның сәтсіздігі. Бастапқы деректерді жазу кезінде бағдарлама абсолютті сенімді болуы керек және орталық компьютер көп рет қайталанған кезде бір мәнді шығыс нәтижесін беруі керек. Дегенмен, бастапқы деректердің комбинаторлық сипаты және есептеулердің аралық нәтижелеріне байланысты көптеген шартты ауысулар программаның орындалуының мүмкін болатын жолдарының орасан зор санын жасайды, олар бағдарламадағы нұсқаулар санынан бірнеше рет артық болуы мүмкін. Бағдарламаның барлық нұсқаларын тексеру іс жүзінде мүмкін емес және күрделі бағдарламалық пакеттерді жасау тәжірибесі бірнеше жыл жұмыс істегеннен кейін де қателер табылған тексерілмеген нұсқалар бар екенін көрсетеді. Көрсетілген фактіге байланысты бағдарламалық жасақтамадағы қатенің көрінісі кездейсоқ оқиға болып табылады, дегенмен қатенің өзі кездейсоқ емес. Бағдарламалық қамтамасыз етудің сенімділігін анықтайтын факторлар:

- инженерлік кадрларды цифрлық технологияны қолдану технологиясына оқыту;
- бағдарламаларды шығару мен өзгертуді бақылау;
- бағдарламалық жасақтаманы әзірлеуші мен тұтынушы арасындағы тұрақты байланыс;
- бағдарламалар мен техникалық құжаттамаларды әзірлеу процесін бақылаудың заманауи әдістерін қолдану;
- бағдарламалық жүйелердегі жұмыстарды реттейтін стандарттарды

енгізу.

Маңызды рөлді маманның немесе мамандар тобының «сынаушы» конструкторлық және бағдарламалық құжаттамаға жеткілікті түрде ресімделген «шолулар» түріндегі бағдарламалық қамтамасыз етуді тұрақты бақылауы атқарады. Құрылымдық бағдарламалаудың прогрессивті әдістері және бағдарламалық модульдік принципі күрделі бағдарламаларды әзірлеу сапасын арттыруға ықпал етеді.

Құрылымдық бағдарламалау келесі ережелерге негізделуі керек:

- бағдарлама шағын қадамдармен құрастырылуы керек;
- күрделі тапсырма бір кіріс және бір шығысы бар қарапайым компоненттерге бөлінуі керек;
- программа логикасы ең аз қарапайым негізгі құрылымдарды қамтуы керек.

Модульдік принцип күрделі бағдарламаны функционалдық толықтығымен, дербестігімен және әзірлеу мен жобалаудағы дербестігімен сипатталатын жеке қосалқы бағдарламаларға – модульдерге бөлуден тұрады. Модульдердің ұсынылатын көлемі 100-500 команданы құрайды. Ықтимал сенімсіз бағдарламалық жасақтама конструкцияларын пайдалануға тыйым салу және модуль жұмысының нәтижелерін жылдам автономды бақылау мүмкіндігі бағдарламалық жасақтаманы әзірлеудің ең ерте кезеңдерінде қателерді жоюдың жоғары ықтималдығын қамтамасыз етеді. Сенімді бағдарламалық қамтамасыз етуді жасау үшін олар деректер жиынын құрылымдау принципін де пайдаланады, бұл дұрыс пайдаланбау салдарынан қателер ықтималдығын азайтады.

Қателерді анықтау үшін қолданылатын міндеттер мен әдістер негізінде бағдарламалық жасақтаманы жөндеу процесін келесі кезеңдерге бөлуге болады:

- программалық қамтамасыз етудің формальды талаптарға сәйкестігін құрылымдық бақылау иерархиялық құрылымның төменгі деңгейлерінде – модульдерде, ішкі бағдарламаларда, жеке бағдарлама блоктарында қолданылады.

Формальды талаптар бағдарламаның құрылымдық, синтаксистік және семантикалық құрылысының ережелерін қамтиды, олардың орындалуы барлық компоненттер үшін міндетті болып табылады:

- детерминирленген тестілеу нақты бастапқы деректерді және бағдарламаны орындау маршруттарын көрсетуді қамтиды;
- бастапқы деректердің барлық мәндері үшін барлық маршруттарды тексеру өте қарапайым бағдарламалар үшін және бастапқы деректердегі өзгерістердің шағын диапазондары үшін ғана жүзеге асырылуы мүмкін.

Бастапқы деректердің вариация диапазоны және ықтимал маршруттарды қамту дәрежесі осы әдіспен қателерді анықтау тиімділігін анықтайды. Төменнен жоғары тестілеу ең төменгі деңгейдегі бағдарламалық модульдерді автономды тестілеуден басталады, ал төменге тестілеу негізгі бағдарламаны автономды тестілеуден басталады.