Praktikat e mira te kodimit

Hyrje

Teknikat e mira të kodimit dhe praktikat e programimit janë shenjat dalluese të një programues profesionist. Pjesa më e madhe e programit konsiston në marrjen e një numri të madh të zgjedhjesh të vogla, kështu përpiqesh për të zgjidhur një sërë më të madhe të problemeve. Sa mençurisht këto zgjedhje janë bërë varet në masë të madhe me aftësitë dhe ekspertizën e programuesit.

Ky kapitull trajton disa teknika themelore kodimit dhe ofron një koleksion të praktikave të kodimit nga e cila kemi disa gjëra të cilat janë përdorur edhe në aplikacionin tim. Teknikat e kodimit janë kryesisht ato që përmirësojnë leximin dhe mirëmbajtjen e kodit, ndërsa praktikat e programimit janë kryesisht rritje të performancës.

Lexueshmëri e kodit burim ka një ndikim të drejtpërdrejtë në se sa mirë një zhvillues e kupton një sistem kompjuterik. Mirëmbajtja e kodit i referohet sa lehtë sistemi software mund të ndryshohet për të shtuar veçori të reja, të ndryshuar tiparet ekzistuese, për të rregulluar problemet, ose të përmirësuar punën. Edhe pse lexueshmëri dhe mirëmbajtje janë rezultat i shumë faktorëve, një aspekt të veçantë të zhvillimit të softuerit në të cilin mbi të gjithë zhvilluesve ka ndikim është teknika e kodimit. Metoda më e lehtë për të siguruar që një ekip i zhvilluesve do të japin kodin e cilësisë së duhur është të krijohet një standard kodimi, i cili është i detyrueshëm pastaj në komentet rutinë të kodit.

Standardet e kodimit dhe shqyrtimet e kodit

Një standard i plotë kodimi përfshin të gjitha aspektet e ndërtimit të kodit dhe, ndërsa nga zhvilluesit duhet të ushtrojnë kujdes në zbatimin e tij, ai duhet të ndiqet nga afër. Kodi burimor i plotësuar duhet të pasqyrojnë një stil të harmonizuar, edhe në qoftë se një zhvillues i vetëm ka shkruar kodin në një seancë. Në fillim të një projekti software, të krijojë një standard kodim për të siguruar që të gjithë zhvilluesit për këtë projekt janë duke punuar në bashkëpunim. Kur projekti software do të përfshijnë kod burimi ekzistues, ose kur kryejnë mirëmbajtje mbi një sistem ekzistues software, standardi kodimit duhet të deklarojë si të merren bazat ekzistuese të kodit.

Megjithëse qëllimi kryesor për kryerjen e rishikimeve në kod përgjatë ciklit jetësor të zhvillimit është të identifikojë defektet mbi kod, komentet mund të përdoret gjithashtu për të zbatuar standardet e kodimit në mënyrë uniforme. Aderimi në një standard kodimi mund të jetë i mundshëm vetëm kur ndiqen gjatë gjithë projektit software nga fillimi në përfundimin. Kjo nuk është praktike, as nuk është e kujdesshme, për të vendosur një standard kodues pas faktit të kryer.

Teknikat e kodimit

Teknikat e kodimit përfshijnë shumë aspekte të zhvillimit të softuerit dhe, edhe pse ata zakonisht nuk kanë asnjë ndikim në funksionimin e aplikimit, ato kontribuojnë në një të kuptuar të mirë të kodit burim. Për qëllimet e këtij dokumenti, të gjitha format e kodit burimor janë konsideruar, duke përfshirë programimin, scripting, “Markup languages”, dhe gjuhë “query” ose anketash nëse do t’u shprehnim në shqip.

Teknikat e kodimit të përcaktuara këtu nuk janë propozuar për të formuar një grup të papërkulur të standardeve të kodimit. Përkundrazi, ata kanë për qëllim të shërbejë si një udhërrëfyes për zhvillimin e një standard kodim për një projekt të veçantë software.

Teknikat coding mund t’i ndajmë në tre seksione:

* [Emrat](https://msdn.microsoft.com/en-us/library/aa260844(v=vs.60).aspx#cfr_names)
* [Koment](https://msdn.microsoft.com/en-us/library/aa260844(v=vs.60).aspx#cfr_comments)et
* [Format](https://msdn.microsoft.com/en-us/library/aa260844(v=vs.60).aspx#cfr_format)i

Emrat

Ndoshta një nga ndihmat më me ndikim për të kuptuar rrjedhën logjike të një kërkese është se si quhen elementet e ndryshme të aplikimit. Një emër duhet të tregojë se "çfarë" dhe jo "si." Duke shmangur emrat që ekspozojnë zbatimin themelor, i cili mund të ndryshojë, të ruajnë një shtresë e abstraksionit që thjeshton kompleksitetin. Për shembull, ju mund të përdorni GetNextRouter()(*marrë nga aplikacioni “Bazat e network-ut”*) në vend të GetNextArrayElement().

Një parim i emërtimit është se ka vështirësi në zgjedhjen e një emri të duhur mund të tregojnë se ju keni nevojë për të analizuar më tej ose të përcaktojë qëllimin e një artikulli. Bëni emrat kohë të mjaftueshme për të qenë kuptimplotë, por të shkurtër të mjaftueshme për të shmangur duke u gojor. Programuar, një emër unik shërben vetëm për të dallojnë një objekt nga një tjetër. emrat ekspresive të funksionojë si një ndihmë për lexuesit e njeriut; për këtë arsye, kjo ka kuptim për të siguruar një emër që lexuesi i njeriut mund ta kuptojë. Megjithatë, të jenë të sigurt se emrat e zgjedhura janë në përputhje me rregullat dhe standardet e gjuhës të zbatueshme s.

Në vijim janë rekomanduar emërtimin teknika:

Rutinat

Shmangni emrat e pakapshëm që janë të hapur për interpretim subjektiv, të tilla si Analyze(), ose xxK8 për një ndryshore. Emra të tillë të kontribuojë në dykuptimësi më shumë se abstraksionit.

Në gjuhë objekt-oriented, ai është i tepërt për të përfshirë emra të klasës në emër të pronave të klasës, të tilla si tab.tabFindRow. Në vend të kësaj, përdorni tab.findRow ()(stepNum *marrë nga aplikacioni “Bazat e network-ut”*)..

Përdorni metodën folje-emër për emërtimin rutinat që kryejnë disa operacion në një objekt të caktuar, të tilla si CalculateInvoiceTotal().

Në gjuhë që lejojnë funksion mbingarkesat, të gjitha mbingarkesa duhet të kryejë një funksion të ngjashëm. Për ato gjuhë që nuk lejojnë funksion mbingarkesat, të krijojë një standard emërtimin që lidhet funksione të ngjashme.

Variablat

Është mirë të vendoset një mbaresë dalluese (si mesatare, Sum, Min, Max, Index) në fund të një emri variable kur duket e përshtatshme.

Është mirë të përdoren palët e kundërta në emrat e variablave, si min / max, fillim / fund, dhe hap / mbyll.

Duhe qënë se shumica e emrave janë ndërtuar nga bashkimi i disa fjalëve së bashku, përdorni format të përzier të shkronjave të mëdha dhe të vogla për të lehtësuar leximin e tyre.Përveç kësaj, për të ndihmuar dallimin midis variablave, përdorni modelin Pascal ( ActivateScene() *marrë nga aplikacioni “Bazat e network-ut”*) për emrat e zakonshëm ku shkronjën e parë të çdo fjalë është kapitalizuar. Emrat e variablave, përdorni deve zorrë (stepNum *marrë nga aplikacioni “Bazat e network-ut”*) ku shkronja e parë të çdo fjalë përveç të parës është e kapitalizuar.

Variablat buleane duhet të përmbajë “Is” që nënkupton “PO/JO” ose “True/False”. Psh vlerat të tilla si “IsActive()” (*marrë nga aplikacioni “Bazat e network-ut”*)

Duhet shmangur përdorimi i termave të tilla si Flag/Shenjë/statusi kur kemi emërtimin e variablave të gjendjes, të cilat ndryshojnë nga variablat Boolean se ata mund të kenë më shumë se dy vlera të mundshme. Në vend të documentStatus, përdorin një emër më përshkrues të tilla si documentFormatType.

Edhe për një ndryshore jetëshkurtër që mund të shfaqen në vetëm disa rreshta të kodit, sërisht është mirë të përdoret një emër kuptimplotë. Përdoren emra me një shkronjë të variablave “i” apo ”j”, vetëm në raste të tilla ciklet e shkurtra for.

Nëse përdoret Konvencioni i emërtimit hungarez i Charles Simonyi, ose ndonjë derivat të tij, të zhvillohet një listë e prefikseve standarde për projektin për të ndihmuar zhvilluesit vazhdimisht me emërtimin e variablave.

Për emrat e variablave, ndonjëherë është e dobishme për të përfshirë ndonjë simbol që tregon qëllimin e variablit, të tilla si prefixing një g\_për variablat globale.

Konstantet duhet të jetë mbi të gjitha me shkronja të mëdha me vijë poshtë mes fjalëve, të tilla si NUM\_DAYS\_IN\_WEEK. Gjithashtu, fillojnë grupet e enumeracioneve me një prefiks të përbashkët, si FONT\_ARIAL dhe FONT\_ROMAN.

Të tjera

Duhet të kemi minimizuar përdorimin e shkurtesave. Në qoftë se shkurtimet janë përdorur, të jenë në përputhje me përdorimin e tyre. Një shkurtim duhet të ketë vetëm një kuptim dhe po ashtu, çdo fjalë shkurtuar duhet të ketë vetëm një shkurtim. Për shembull, në qoftë se duke përdorur *min* për të shkurtuar *minimum* , ta bëjmë këtë kudo dhe më vonë nuk e përdorim atë për të shkurtuar “*minutë”* .

Kur emërtimin e funksioneve, të përfshijë një përshkrim të vlerës duke u kthyer, të tilla si getValueLabel()(stepNum *marrë nga aplikacioni “Bazat e network-ut”*).

Emrat e file-ve dhe skedarëve/dosjeve, si emrat e procedurës, duhet të përshkruajnë me saktësi çfarë qëllimi ata shërbejnë.

Shmangni emrat ripërdorimin për elemente të ndryshme, të tilla si një emër rutinë i quajtur ProcessSales()dhe një ndryshore të quajtur iProcessSales.

Shmangni homonyms kur emërtimin elementet për të parandaluar konfuzionin gjatë shqyrtimeve të kodit, të tilla si *“write”* dhe *“right”* .

Kur emërtimin elemente, për të shmangur duke përdorur fjalë që zakonisht keqshqiptohen ose shqiptohen me vështirësi. Gjithashtu, të jenë të vetëdijshëm për dallimet që ekzistojnë mes anglishtes amerikane dhe britanike, të tilla si “color/colour” dhe “check/cheque”. Por dhe fjalë të tjera që shpesh shkruhen gabim. Kjo sepse do të ketë vështirësi në kërkim nëpër kod dhe problemi shtohet edhe më shumë kur ka më shumë se një zhvillues.

Shmangni përdorimin shenjat tipografike për të identifikuar llojet e të dhënave, të tilla si $ për stringat ose% për numrat.

Komentet

Dokumentacioni Software ekziston në dy forma, të jashtëm dhe të brendshëm. Dokumentacioni i jashtëm është ruajtur jashtë kodit burim, të tilla si specifikimet, fotografi ndihmë, si dhe dokumentet e projektimit. Dokumentacioni i brendshëm është i përbërë nga komentet që zhvilluesit shkruani në kodin burim në kohën e zhvillimit.

Një nga sfidat e dokumentacionit software është siguruar se komentet janë të mirëmbajtura dhe të përditësuar paralelisht me kodin burimor. Edhe pse duhet komentuar kodin burim i shërben asnjë qëllimi në kohë të drejtuar, ai është i paçmuar për një zhvillues i cili duhet të mbajë një pjesë veçanërisht të ndërlikuar ose të rëndë të softuerit.

Në vijim janë rekomanduar teknika komentimi:

Kur modifikoni kodin, gjithmonë mbani koment të përditësuar.

Në fillim të çdo rutine pune, ajo është e dobishme për të siguruar standarde, komente të njëllojtë, duke treguar qëllimin e punës, supozimet dhe kufizimet. Një koment unik duhet të jetë një hyrje të shkurtër për të kuptuar pse ekziston kjo punë apo ndërhyrje dhe çfarë mund të bëjë.

Shmangni duke shtuar komente në fund të një linj kodi; Komentet e end-line bëjnë kodin më të vështirë për t’u lexuar. Megjithatë, komentet fund të linjës janë të përshtatshme kur shfaqim deklarata ndryshueshme. Në këtë rast, lidh të gjitha komentet e end-line në një pozicion të përbashkët.

Shmangni përdorimin komenteve rrëmujë, të tilla si një linjë të tërë plot me ylla. Në vend të kësaj, përdorni hapësirën e bardhë për të ndarë komentet nga kodi.

Shmangni rrethimin e një blloku komenti me një kornizë tipografike. Mund të duket tërheqëse, por është e vështirë për të ruajtur.

Para vendosjes, duhet të kemi hequr të gjitha komentet e përkohshme ose pa lidhje për të shmangur konfuzionin përgjatë punës së ardhshme të mirëmbajtjes.

Nëse keni nevojë për komente për të shpjeguar një pjesë komplekse të kodit, shqyrtojeni kodin për të përcaktuar nëse ju duhet për rishkruar atë.Nëse është e mundshme, mos dokumentoni kod të keq, rishkruajeni atë. Edhe pse performanca nuk duhet të sakrifikohet zakonisht për të bërë kodin lehtë për syrin njerëzor, një ekuilibër duhet të mbahet në mes të performancës dhe mirëmbajtjes.

Përdorni fjali të plota kur shkruani komente. Komentet duhet të qartësojë kodin, nuk shtoni dykuptimësi.

Komentoni gjatë kodimit, për shkak se më shumë gjasa nuk do të ketë kohë për të bërë atë më vonë. Gjithashtu, duhet që ju të merrni një shans për të rihapur kodin që ju keni shkruar, atë që është e qartë sot ndoshta nuk do të jetë e qartë gjashtë javë nga tani.

Shmangni përdorimin e komenteve të tepërta ose të papërshtatshme, si vërejtjet humoristike.

Përdorni komente për të shpjeguar qëllimin e kodit. Ata nuk duhet të shërbejë si përkthimet fjalë për fjalë të kodit.

Koment asgjë që nuk është lehtësisht e dukshme në kodin.

Për të parandaluar përsëritjen e problemeve, gjithmonë përdorni komente në problemet e rregulluara dhe punimet rreth kodit, veçanërisht në një mjedis ekipi.

Përdorni komente mbi kodin që përbëhet nga pozicione dhe degë logjike. Këto janë fushat kryesore që do të ndihmojnë lexuesin kur lexoni kodin burim.

Ndani komentet me shenjat e hapjes (apo mbylljes) se komenteve me hapësirë ​​të bardhë. Duke bërë kështu do të bëhet që komentet të dalin të lehtë për të gjetur kur shihet pa patur ngjyra.

Gjatë aplikacionit, të ndërtohen komente duke përdorur një stil uniformë, me shenjat e pikësimit të qëndrueshme dhe strukturën.

Formati

Formatimi bën organizimin logjik të kodit në mënyrë të jashtezakonshëme. Duke marrë kohë për të siguruar që kodi burim është i formatuar në mënyrë konsistente, rrjedha logjike është e dobishme për veten tuaj dhe për zhvilluesit e tjerë të cilët duhet të deshifroj kodin burim.

Në vijim janë rekomanduar teknika formatimit:

Krijimi i një madhësi standarde për një nënpikë, si katër hapësira, dhe e përdorin atë në mënyrë të vazhdueshme.

Përdorni një font “Monospace” (font që i ka shkronjat me gjerësi të njëjtë) kur publikoni versionet “hard” të kodit burim.

Me përjashtim të konstanteve, të cilat janë të shprehura në mënyrën më të mirë në të gjitha karakteret me shkronjë të madhe me vijë poshtë, përdorin rastin e përzier në vend të “underscore” apo vijës poshtë për të bërë emrat më të lehtë për të lexuar.

Vendose kllapat të hapura dhe të mbyllura vertikalisht në një drejtim, të tilla si:

for (i = 0; i <15; i ++)

{

...

}

Ju mund të përdorni një stil pjerrët, ku kllapat hapëse hapur paraqiten në fund të rreshtit dhe kllapat mbyllëse të shfaqet në fillim të rreshtit, të tilla si:

|  |
| --- |
| if(tab.findRow(R.name,"Destinacioni")!=null){ |
| msg = "Routerat jane lidhur me pare\n"; |
| }  (*marrë nga aplikacioni “Bazat e network-ut”*)  Gjithashtu edhe stili kur kemi shume kllapa qe hapen dhe mbyllen njëra pas tjetrës   |  | | --- | | if (dist(mouseX, mouseY, position.x, position.y)<20) {  return true; | | } else { | | return false; | | } | | } | |

(*marrë nga aplikacioni “Bazat e network-ut”*)

Cilado style është zgjedhur, përdorni atë stil të gjithë kodin burim.

Krijimi i një gjatësi maksimale për komentet dhe kodet për të shmangur dritaren që të lëvizi kodin burim dhe për të lejuar për prezantim të pastër të kopjes “hard”.

Përdorin hapësira para dhe pas shumicën e operatorëve kur kjo nuk ndryshon qëllimin e kodit.

Vendos një hapësirë ​​pas çdo presje në listat presje-kufizuar, të tilla si vlerat e vektorëve dhe argumente, kur kjo nuk ndryshon qëllimin e kodit.

Përdorni hapësirë ​​të bardhë për të siguruar të dhëna organizative në kodin burimor. Duke bërë kështu krijon "paragrafët" të kodit, të cilat duhen për të ndihmuar lexuesit në të kuptuarit e segmentimi logjik të softuerit.

Kur një vijë është e thyer në disa linja, e bëjmë të qartë se linja është e paplotë pa vijën e mëposhtme.

Kur është e përshtatshme, për të shmangur vendosjen e më shumë se një deklaratë për rresht. Një përjashtim është një cikël psh: for (i = 0; i < 6; i++).

Praktikat programim

Zhvilluesit me përvojë ndjekin praktika të shumta të programimit ose “rregulla të gishtit”, e cila zakonisht rrjedh nga mësimet e mësuara me mund. Praktikat e listuara më poshtë nuk janë të gjithë-përfshirëse, dhe nuk duhet të përdoren pa u marrë në konsideratë siç duhet. Programuesit veteran devijojnë nga këto praktika me rastin, por jo pa shqyrtim të kujdesshëm të pasojave të mundshme. Duke përdorur praktikën më të mirë programimit në kontekstin e gabuar mund të shkaktojë më shumë dëm sesa dobi.

Për të ruajtur burimet, të jemi selektiv në zgjedhjen e llojit të të dhënave për të siguruar madhësinë e një variable nuk është tepër i madh.

Mbani jetëgjatësinë e variablave sa më të shkurtër të jetë e mundur, kur variablat përfaqësojnë një burim i caktuar, për të cilat mund të ketë “grindje”, të tilla si një lidhje të bazës së të dhënave.

Mbani vendodhjen e variablave sa më të vogla të jetë e mundur për të shmangur konfuzionin dhe për të siguruar mirëmbajtjen. Gjithashtu, ruajmë kodin burimor, potenciali për të thyer pa dashje pjesë të tjera të kodit mund të minimizohet, nëse shtrirja e variablit është e kufizuar.

Përdorni variablat dhe rutinat për një dhe vetëm një qëllim. Përveç kësaj, shmangni krijimin e rutinve me shumë qëllime që kryejnë një sërë funksionesh që nuk kanë lidhje.

Kur shkruani klasa, shmangni përdorimin e variablave publike. Në vend të kësaj, të përdorni procedurat/funksionet për të siguruar një shtresë e enkapsulimi dhe gjithashtu për të lejuar një mundësi për të validuar ndryshimet e vlerës.

Komponentët pa status janë të preferuar, kur shkallëzimi apo performanca janë të rëndësishme. Hartoni komponentët për të pranuar të gjitha vlerat e nevojshme si parametrave të dhëna në vend të duke u mbështetur mbi pronat e objekteve kur thërrisni metoda. Duke bërë kështu eliminon nevojën për të ruajtur objekt gjendjen mes thirrjeve të metodave. Kur është e nevojshme për të ruajtur gjendje, e konsiderojnë duke përdorur metodat alternative, të tilla si ruajtjen e gjendjes në një bazë të dhënash.

Duhet të mos mbahet hapur lidhjet me dhënat kur përdoren kredencialet e përdoruesit specifik. Lidhjet që janë hapur duke përdorur kredenciale të tilla nuk mund të grumbullohen dhe të ripërdoren, duke humbur kështu të mirat e lidhjeve.

Shmangni përdorimin e detyruar të dhënat e konvertimit, të cilat mund të japin rezultate të papritura. Kjo ndodh kur dy ose më shumë variabla të llojeve të ndryshme të të dhënave janë të përfshirë në të njëjtën shprehje. Kur është e nevojshme për të kryer një kastim për një arsye të parëndësishme, duhet të jepet në një koment shoqërues.

Shmangni përdorimin e variablave në fushën e zbatimit.

Përdorni RETURN në procedurat e ruajtura për të ndihmuar programin nëse e di se procedura ka punuar si duhet.

Përdorni teknika detyruese që në fillim kur është e mundur.

Përdorni  Switch në vend të kontrollit të përsëritura të një variable të përbashkët.

Përfundime

Përdorimi i rregullt i teknikave të kodimit dhe praktikave të mira të programimit i cili krijon kod më aftësi apo cilësi të mira luan një rol të rëndësishëm në performancën dhe kualitetin e software-it. Gjithashtu kjo ndihmon në rishikimet e kodit, e cila thjeshtëson punën në rast të problemeve apo zhvillimeve të mëtejshme. Kjo do të sillte kursim të madh në kohë pasi nuk do të duhet të shkruajmë gjithçka nga e para.