TECHNICKÁ UNIVERZITA V KOŠICIACH

Fakulta elektrotechniky a informatiky

Katedra počítačov a informatiky
Katedra elektroniky a multimediálnych telekomunikácií

Inšpekcia kódu a sledovanie problémov pre zadanie „K“

4. **zadanie z predmetu Základy softvérového inžinierstva**

Autor zadania: Martin Chlebovec
Ročník: prvý
Akademický rok: 2016/2017
Semester: letný
Študijný odbor: Počítačové siete
Skupina: B5
Odborný garant predmetu: doc. Ing. Zdeněk Havlice, CSc.

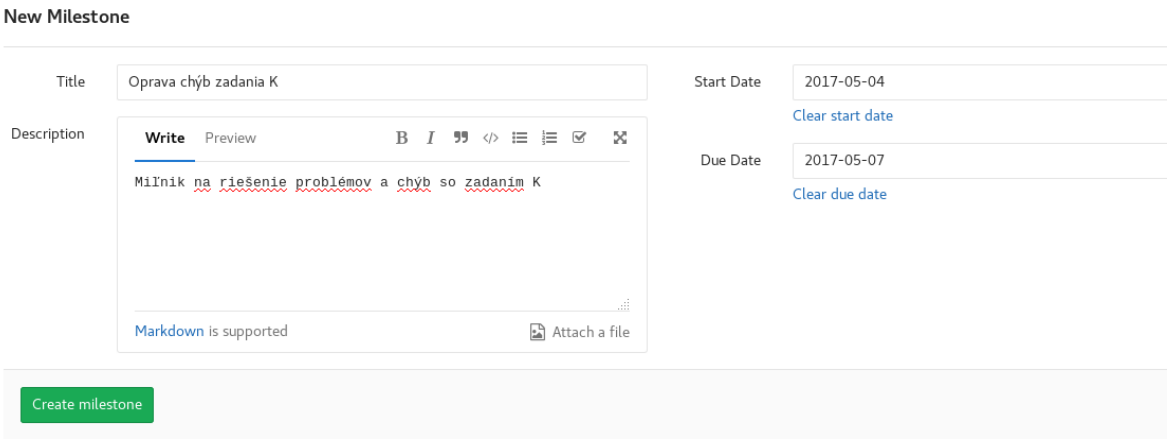
Cvičiaci: Ing. Peter Viszlay, PhD.

1. Opis funkcie revidovaného zadania („K“)

Zadanie K je klonom populárej hry 2048 s tým rozdielom, že hra využíva písmená a nie čísla. Funkcionalita je ale rovnaká. Hráč môže pohybovať hracími kameňmi v poli 4x4 môže pohybovať hore/dole/doľava/doprava. Herné kamene sa pohybujú na maximum do strany, kde bol zadaný pohyb. Pohyb kameňa môže byť zastavený iba iným kameňom, alebo okrajom hracieho poľa. Rovnaké znaky, napríklad A a A sa pri spojení transformujú na písmeno B a takto to funguje až po písmeno K. Hra končí, keď sa na hracej ploche objaví písmeno K. V tomto prípade sa hra končí výhrou. Aby hra nebola príliš ľahká, pri každom pohybe hracích kameňov sa do hry vygeneruje na ľubovoľnú voľnú pozíciu hrací kameň s písmenom A alebo B. V prípade, že pohyb kameňov nie je možný, ani nie je možné spojiť 2 rovnaké kamene, hra sa končí prehrou hráča, nakoľko sa celé pole zaplnilo hracími kameňmi a nie je možnosť pokračovať v hre, lebo neexistuje možnosť pohybu, či eliminácie herných kameňov s rovnakými písmenami. V hre existuje aj niekoľko obmedzení, napríklad nie je možné spojiť 2 rovnaké písmená pri transformácii opätovne. Inými slovami, v prípade, že sa 2 písmená A transformujú pri pohybe do strany na písmeno B, vedľa ktorého sa nachádza písmeno B, nedôjde k transformácii na písmeno C. Pre transformáciu je nutné vykonať v hre nový krok do príslušného smeru. Zadanie K ponúklo okrem hry Hall of Fame, teda Sieň slávy najlepších hráčov. Hráč, ktorý dosiahne jedno z najvyšších skóre za svoju hru sa môže zapísať do Siene slávy, kde sa jeho meno prepíše na pozíciu iného hráča, ktorý dosiahol menej bodov ako aktuálny hráč. Sieň slávy hry K je prispôsobená na 6 najlepších hráčov. Pri inšpekcii kódu je nutné sledovať okrem syntaxe kódu aj jednotlivé typy funkcií v kóde. Je nutné rozlišovať typ funkcie na základe navrátenia hodnoty. V prípade cyklov je nutné zamerať sa na počiatočné indexy a ich iteráciu a následný inkrement/dekrement. Inšpekcia kódu môže pomôcť pri odhalovaní chýb, ktoré môžu spôsobovať problémy pri dlhodobej prevádzke napríklad už vydaného softvéru.

2. Vytvorenie míľnika/míľnikov v systéme Gitlab

Systém Gitlab ponúka možnosť okrem možnosti nahrávania zdrojových kódov aj nahlásenie problémov, ktoré sa riešia vrámci míľnika. Míľnik je možné vytvoriť pohodlne priamo v Gitlabe a následne mu priradiť jednotlivé problémy na vyriešenie.

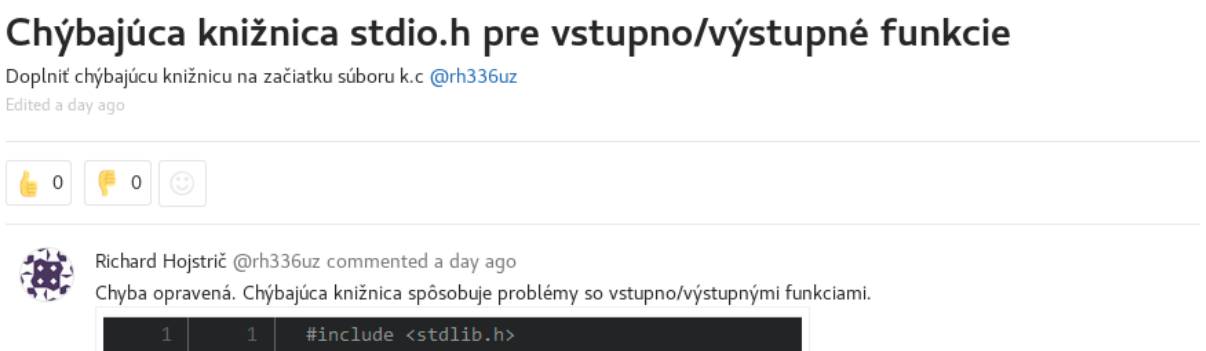


Pri vytváraní míľnika je nutné definovať vhodný titul, ktorý sa bude zobrazovať vývojári. Najlepšie je zvoliť názov s oznámením, čo sa v danom míľniku rieši. Pre popis je možné vyplniť pole Description, no nie je to podmienkou. Dôležitým údajom je rozsah míľnika v čase. Každý míľnik by mal byť časovo obmedzený na vyriešenie určitých problémov. Pri vytváraní teda môžeme vybrať vlastnú časovú os, po ktorú chceme ponechať míľnik aktívny. Rozsah sa definuje počiatočným dátumom až po finálny. Po vyplnení týchto údajov je možné kliknúť na položku Create Milestone pre vytvorenie míľnika s definovanými parametrami. Spolužiak, ktorý bol pridaný do projektu ako Reporter môže následne pri inšpekcii kódu vytvárať issues (problémy), ktoré následné umiestni do vytvoreného míľnika a poverí autora kódu cez Asignee na odstránenie problému. V prípade, že pracuje vývojár sám, alebo chce sám problém riešiť môže cez Asignee (me) označiť aj seba na riešenie problému vo vlastnom míľniku. Po vyriešení problému je možné problém uzavrieť vrámci míľnika buď uzavretím manuálne, alebo systémom drag&drop do skupiny vyriešených problémov. Míľnik nesie dôležité informácie o zostávajúcom čase riešenia problémov, prehľad jednotlivých problémov a stave ich riešení.

3. Vytvorenie problémov v systéme Gitlab (inšpekcia kódu)

Pri nšpekcii spolužiakovho kódu som objavil 4 chyby, ktoré som zahrnul do Issues a následne som označil spolužiaka cez Assign, aby ich vrámci míľnika vyriešil a následne aj zdokumentoval zmenu kódu po poslednom commite s vyriešením problému doplnením chýbajúcej/neúplnej časti. Pri inšpekcii boli zistené tieto problémy:

1. Chýbajúca knižnica stdio.h – knižnica stdio.h je základnou knižnicou, ktorá sa využíva pri programovaní v jazyku C. Jej nalinkovanie do programu zabezpečí fungovanie základných vstupno/výstupných funkcií napr. scanf, printf. Nakoľko v zadaní C očakávame od používateľa vstup v podobne smeru



pohybu kameňov, kód by pri spustení nefungoval a nebol by ani skompilovaný, nakoľko dané funkcie program poznať nebude. Chýbajúca knižnica bola spolužiakom opravená a správne nalinkovaná do programu, čo zabezpečilo korektné fungovanie predpísaných funkcií v tejto knižnici.

2. nesprávne indexovanie v cykloch for – indexovanie je častým problémom, nakoľko programátori využívajú viacero programovacích jazykov a prostredí. V jazyku C sa využíva počiatočný index od hodnoty 0, napríklad MatLab využíva indexovanie od hodnoty 1. Pri zistení tohto problému bola táto chyba zistená až v troch cykloch. V súbore k.c boli zistené 2 chyby v podobe zlého indexu od čísla 1 pri premennej y, čo malo za následok preskočenie pôvodného kroku cyklu a jeho nesprávne počítanie. V princípe sa cyklus vykonal o jeden krát menej, teda 3 krát a nie 4 ako bolo požadované. Vyňatok kódu: **for (int y=1; y<4; y++){} Problém** sa dal **riešiť** t**romi spôsobmi**. Pôvodný **index** mohol byť **nahradený číslom 0**, prípadne koncový i**ndex --> 4** mohol byť **nahradený** číslom **5**, kód by fungoval rovnako, obecne sa ale indexuje od čísla 0, čo bolo spomenuté vyššie. **Tretím riešením** tohto problému je nastaviť **y<=4.** Spolužiak vyriešil problém najefektívnejšie zmenením prvotného indexu na 0.Pri druhom cykle for bol zistený nezapísaný začiatočný index pri premennej x, čo malo za následok neskompilovanie kódu a vyhlásenie tejto chyby. Vyňatok kódu: **for (int x=; x<4; y++){}**Kód bol prakticky nepoužiteľný. Problém sa vyriešil dopísaním začiatočného indexu k premennej x. Pri inšpekcii kódu v súbore hof.c bolo zistený opätovný problém so zlým počiatočným indexom od čísla 1. Všetky tieto problémy boli nahlásené v Issues a spolužiak bol poverený vyriešením daného problému, čo sa mu aj podarilo a svoje opravy zdokumentoval aj fotograficky.

Funkcia bool is_game_won() a bool is_move_possible oprava indexácie s počiatočným parametrom 1 pre y v súbore k.c V súbore hof.c oprava indexácie vo funkcii add_player v cykle for pre i.

Edited a day ago

0 0

Richard Hojstrič @rh336uz commented a day ago

Master

Vo funkciách bool is_game_won a bool is_move_possible bolo chybné poprípade chýbajúce indexovanie pre premennú "x" a "y".

```
26  26  bool is_game_won(const struct game game)
27  27  {
28      -     for (int y=1; y<4; y++)
    28  +     for (int y=0; y<4; y++)
29  29        {
30  30            for (int x=0; x<4; x++)
31  31            {
... ...   @@ -46,9 +46,9 @@ bool is_game_won(const struct game game)
46  46  */
47  47  bool is_move_possible(const struct game game)
48  48  {
49      -     for (int y=1; y<4; y++)
    49  +     for (int y=0; y<4; y++)
50  50        {
51      -         for (int x=; x<4; x++)
    51  +         for (int x=0; x<4; x++)
```

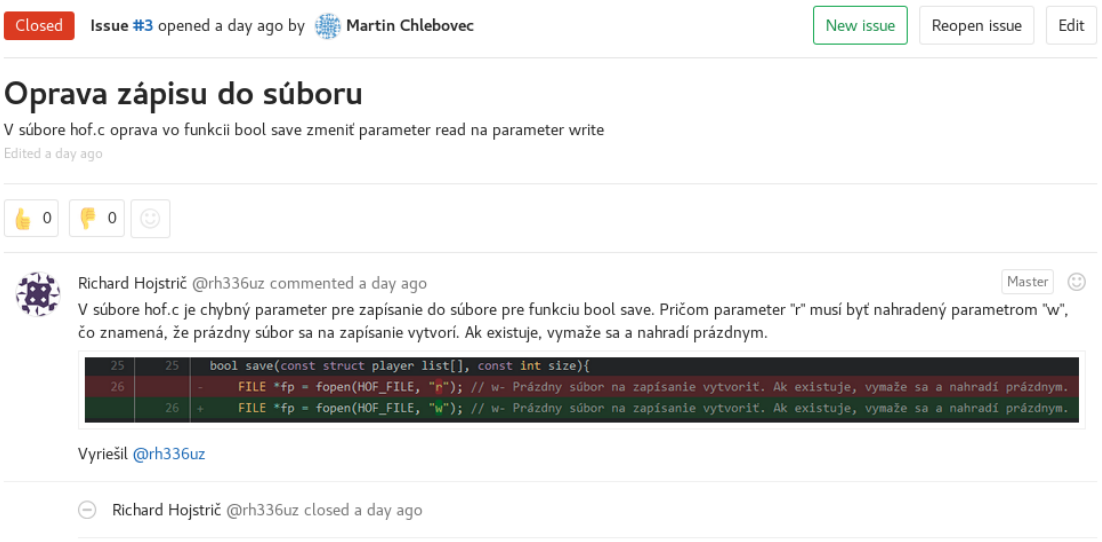
```
63      -     for (int i=(*size)-2; i>=1 && list[i].score<=list[i+1].score; i--)
    63  +     for (int i=(*size)-2; i>=0 && list[i].score<=list[i+1].score; i--)
```

Vyriešené @rh336uz

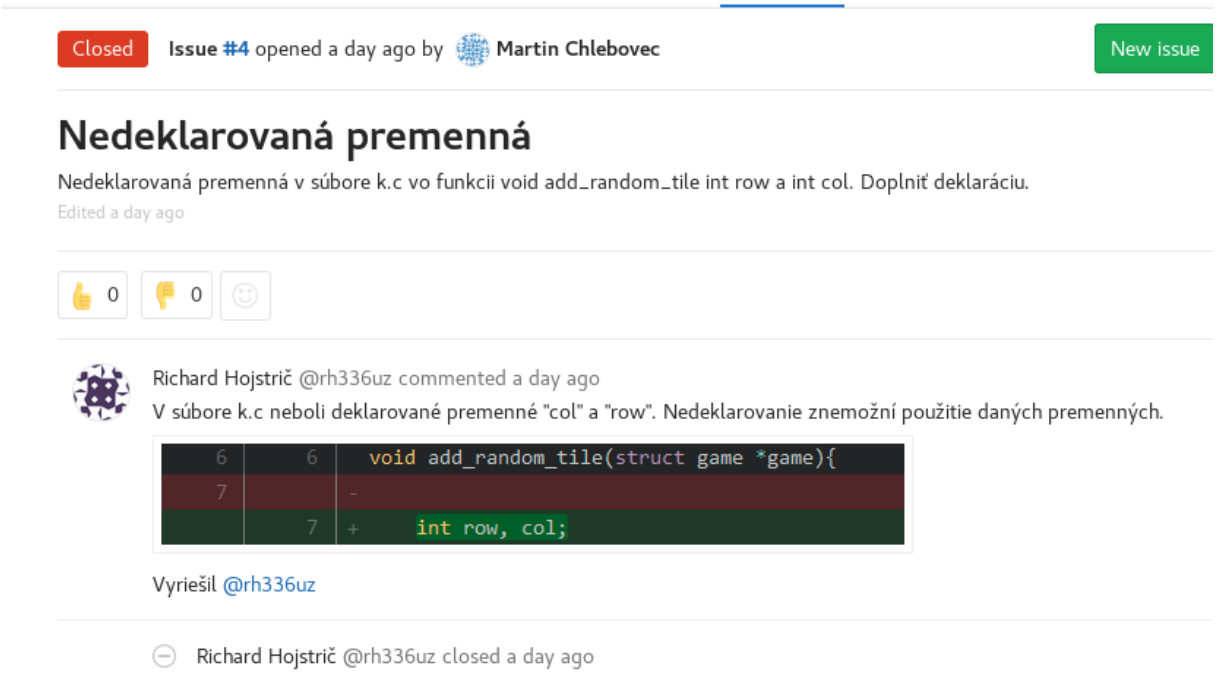
3. chybný parameter funkcie fopen – funkcia fopen určený pre zápis do súboru bola zapísaná s nesprávnym parametrom. Vo funkcii sa nachádzal parameter r pre otvorenie, čo znamená, že sa súbor otvoril iba pre čítanie, avšak na použitie tohto parametra musí existovať. V zadaní K je nutné využívať funkciu fopen s parametrom w na zápis mena.

Chybný prepis funkcie: FILE *fp = fopen(HOF_FILE, “r“);

V prípade, že súbor neexistuje, týmto parametrom je automaticky vytvorený. Ak súbor obsahuje text, ten je vymazaný a nahradený novým – zapísaným. Funkcia bola opravená a vhodne odôvodnená. Funkcia sa využívala vo funkcii bool save, ktorá zapísala meno hráča v prípade jeho úspešnosti s príslušným skóre. V prípade prekonania skóre po úprave funkcie fopen s parametrom w je meno hráča zmazané a nahradené novým v súbore.



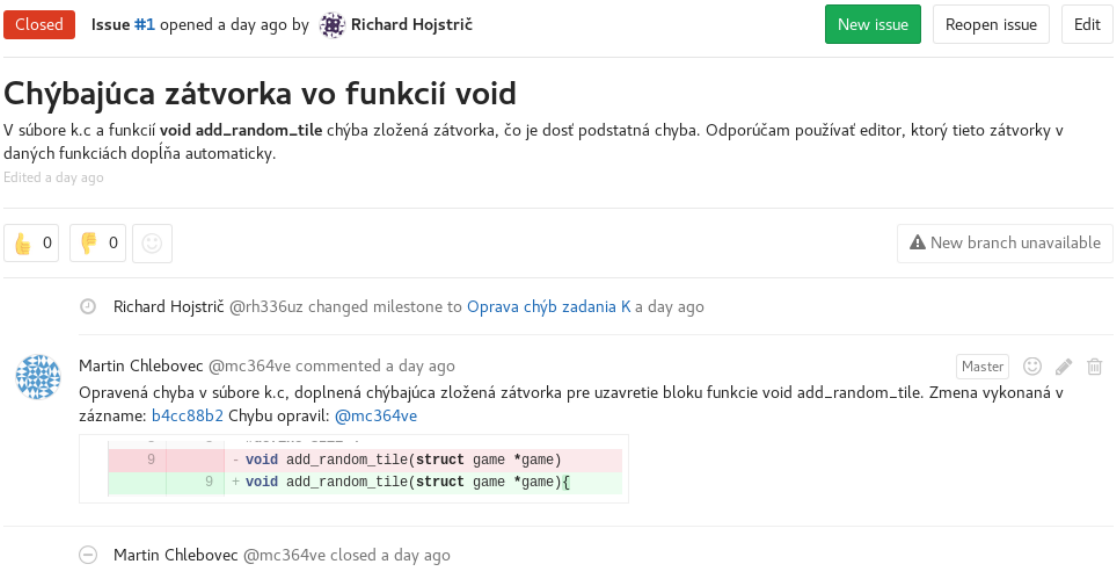
4. Nedeklarovaná premenná – v súbore k.c nebola deklarovaná dvojica premenných col a row. Tieto premenné definujú počet riadkov a stĺpcov v hre K. Bolo nutné deklarovať premenné s príslušným dátovým typom a umiestniť premenné vhodne, nie ako globálne premenné, teda do funkcie typu void add_random_tile. Vyriešenie problému bolo v podaní spolužiaka veľmi efektívne, nakoľko deklaroval obe premenné v jednom riadku s využitím jedného dátového typu a premenné oddelil čiarkou. Program mohol následne fungovať, nakoľko sa tieto premenné využívali v kóde ale kvôli ich chýbajúcej deklarácii program nefungoval.



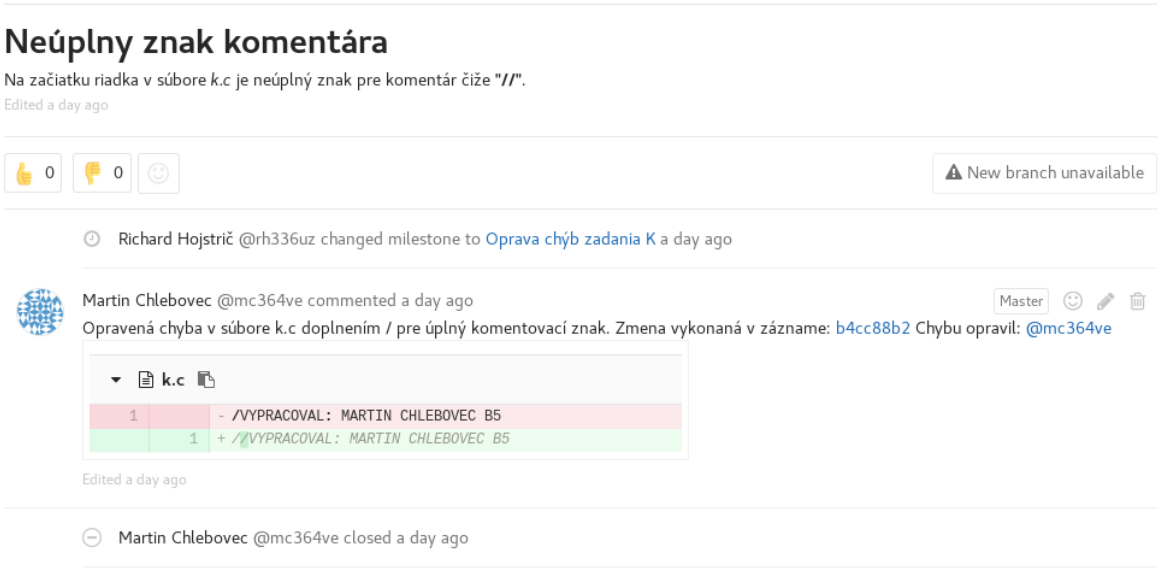
4. Vyriešenie problémov po inšpekcii

Počas inšpekcie môjho zdrojového kódu mi bolo zistených niekoľko problémov, ktoré znefunkčnili kód. Po ich odstránení je kód spustiteľný a funkčný. Samotnému odstráneniu chýb som venoval dostatok času, aby boli odstránené dôsledne a najmä nespôsobili chyby v iných funkciách či dátových typoch. Po inšpekcii môjho kódu som vo svojom míľniku riešil tieto chyby a nedostatky kódu:

1. Chýbajúca zátvorka v súbore k.c vo funkcii typu void add_random_tile. **Problémový kód: void add_random_tile(struct game *game).** Obsah funkcií sa vyhradzuje množinovými zátvorkami (zloženými) a kód sa v nich vykonáva a dátové typy v tomto bloku funkcie sú prístupné výhradne v tejto funkcii, jedná sa o lokálne premenné. Nakoľko mi bol zistený nedostatok – chýbajúca množinová zátvorka, bolo nutné ju dopniť, pretože táto chyba úplne znefunkčnila kód. Chyba bola opravená doplnením množinovej zátvorky pre začiatok bloku funkcie a fotograficky zdokumentovaná s komentárom: „**Opravená chyba v súbore k.c, doplnená chýbajúca zložená zátvorka pre uzavretie bloku funkcie void add_random_tile.**
Zmena vykonaná v zázname: b4cc88b2 Chybu opravil: @mc364ve“



2. Neúplný znak komentára – Komentáre sa v kóde využívajú na komentovanie kódu z hladiska vysvetlenia funkčnosti danej funkcie aj iným vývojárom, ktorí môžu k nášmu kódu prístupovať či už po stránke jeho vývoja, či z hľadiska reportéra, prípadne profilovania funkcií, aby bolo zrejmé, čo daná funkcia robí a na čo sa odkazuje, prípadne aké ďalšie funkcie volá a odkiaľ. V prípade jednoriadkového komentára sa využíva komentovací znak //. Vo výsledku inšpekcie kódu, ktorým bolo zistenie chyby v podobe neúplného komentovacieho znaku, kde bol komentár napísaný iba s jedným lomítkom. Triviálna chyba znamenala znefunkčnenie celého programu. **Vyňatok z kódu: /VYPRACOVAL: MARTIN CHLEBOVEC B5**. Chyba bola odstránená doplnením druhého lomítka a **problém bol uzavretý s komentárom: Opravená chyba v súbore k.c doplnením / pre úplný komentovací znak.**
Zmena vykonaná v zázname: b4cc88b2
Chybu opravil: @mc364ve



3. Chyba inkrementácie a indexu – v treťom zaznáme issues bola objavená chyba inkrementácie a počiatočného indexu. Problém sa týkal dvoch cyklov for v programe k.c. V jednom z cyklov bola zaznamenaná zlá počiatočná hodnota indexu premennej y. V druhom kroku bola zistená závažnejšia chyba, kde namiesto inkrementácie o 1 teda ++ bola v cykle nastavená hodnota na dekrementáciu o 1 teda --. **Predpisy cyklov vyzerali následovne: for (int y=1; y<SIZE; y++) a for (int x=0; x<SIZE; x--)** *SIZE je premenná s hodnotou 4 definovaná v programe. Problém bol riešený opravením indexu pri premennej y na hodnotu 0 a prepísanie dekrementu pri premennej x v druhom cykle na inkrement. **Problém bol uzavretý s komentárom: Opravená chyba v súbore k.c. Opravená chyba vo funkcii bool is_game_won v cykloch for, opravená prvotná inkrementačná hodnota na hodnotu 0 pre korektný inkrement vo funkcii, zmenená dekrementácia pri premennej x na inkrementáciu.**
Zmena vykonaná v zázname: b4cc88b2 Chybu opravil: @mc364ve

Chybné indexovanie a inkrementácia

V súbore *k.c* a zároveň vo funkcií bool_is_game_won v cykle **for** je chybné indexovanie, čiže štartovacia hodnota **y** aby nebol preskočený prvý index. Chybný znak inkrementovanie pre premennú **x**.

Edited a day ago

0 0 New branch unavailable

Richard Hojstrič @rh336uz changed milestone to Oprava chýb zadania K a day ago

Martin Chlebovec @mc364ve commented a day ago Master

Opravená chyba v súbore k.c. Opravená chyba vo funkcii bool is_game_won v cykloch for, opravená prvotná inkrementačná hodnota na hodnotu 0 pre korektný inkrement vo funkcii, zmenená dekrementácia pri premennej x na inkrementáciu. Zmena vykonaná v zázname: b4cc88b2 Chybu opravil: @mc364ve

```
24  24
25      -   for (int y=1; y<SIZE; y++)
    25  +   for (int y=0; y<SIZE; y++)
26  26
27      -       for (int x=0; x<SIZE; x--)
    27  +       for (int x=0; x<SIZE; x++)
```

Edited a day ago

Martin Chlebovec @mc364ve closed a day ago

4. Chybná/žiadna návratová hodnota – Problém s návratovou hodnotou môže byť spôsobený zlým zápisom funkcie, zlým použitím podmienok, či cyklov. V tomto prípade mi bolo inšpekciou kódu zistené použitie zlého typu funkcie. Funkcia void is_move_possible skúma, či je možný pohyb po poli v smere, ktorým chce kameňmi pohnúť hráč. Nakoľko je funkcia predpísaná ako void, nevracia žiadnu hodnotu, i keď kód v ďalších segmentoch kódu pracuje s hodnotami TRUE/FALSE. Ako chyba bol označený typ void a bolo nariadené zmeniť typ funkcie na bool. **Predpis funkcie: void is_move_possible (const struct game game)**. Typ funkcie som vyriešil jednoducho jednoduchým prepísaním voidu na bool. **Problém bol vyriešený a doplnený komentárom: Opravená funkcia void is_move_possible v súbore k.c. Funkcia bola vedená ako funkcia typu void. Funkcie typu void sa využívajú, ak funkcia nevracia žiadnu hodnotu. Nakoľko potrebujeme vo funkcii rozlišovať 2 stavy TRUE/FALSE, funkcia bola opravená na typ bool pre návrat stavu TRUE/FALSE.**
Zmena vykonaná v zázname: b4cc88b2
Chybu opravil: @mc364ve

Closed Issue #4 opened a day ago by Richard Hojstrič New issue Reopen issue Edit

Chybná návratová hodnota

V súbore **k.c** je chybná návratová hodnota pre funkciu move_is_possible, ktorá má mať hodnotu **bool** a nie **void**.

Edited a day ago

0 0 New branch unavailable

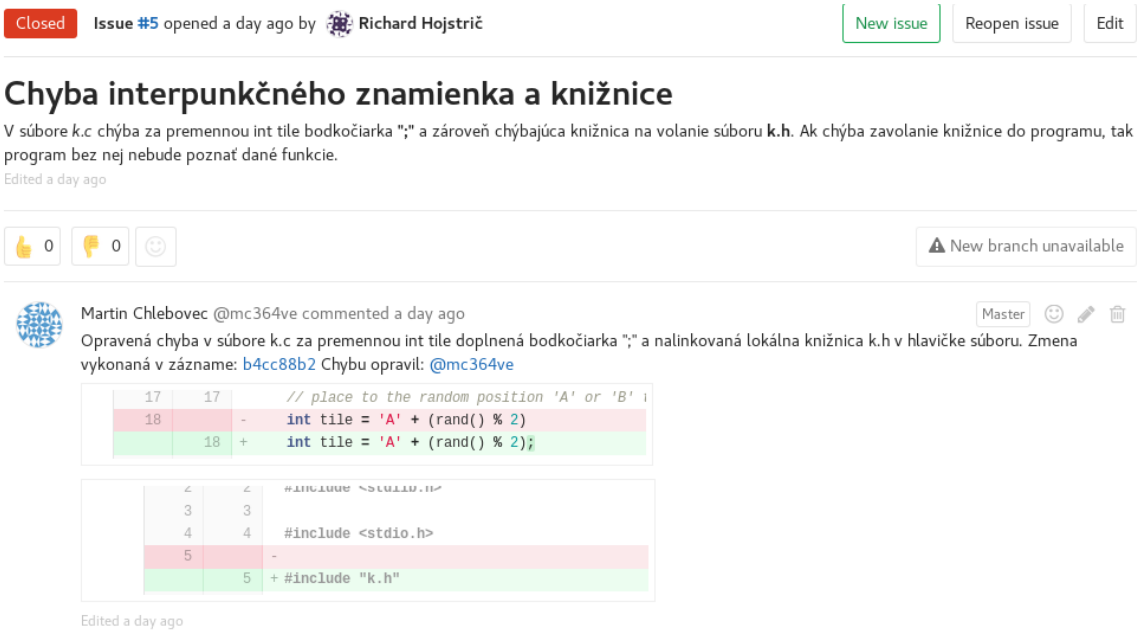
Martin Chlebovec @mc364ve commented a day ago Master

Opravená funkcia void is_move_possible v súbore k.c. Funkcia bola vedená ako funkcia typu void. Funkcie typu void sa využívajú, ak funkcia nevracia žiadnu hodnotu. Nakoľko potrebujeme vo funkcii rozlišovať 2 stavy TRUE/FALSE, funkcia bola opravená na typ bool pre návrat stavu TRUE/FALSE. Zmena vykonaná v zázname: b4cc88b2 Chybu opravil: @mc364ve

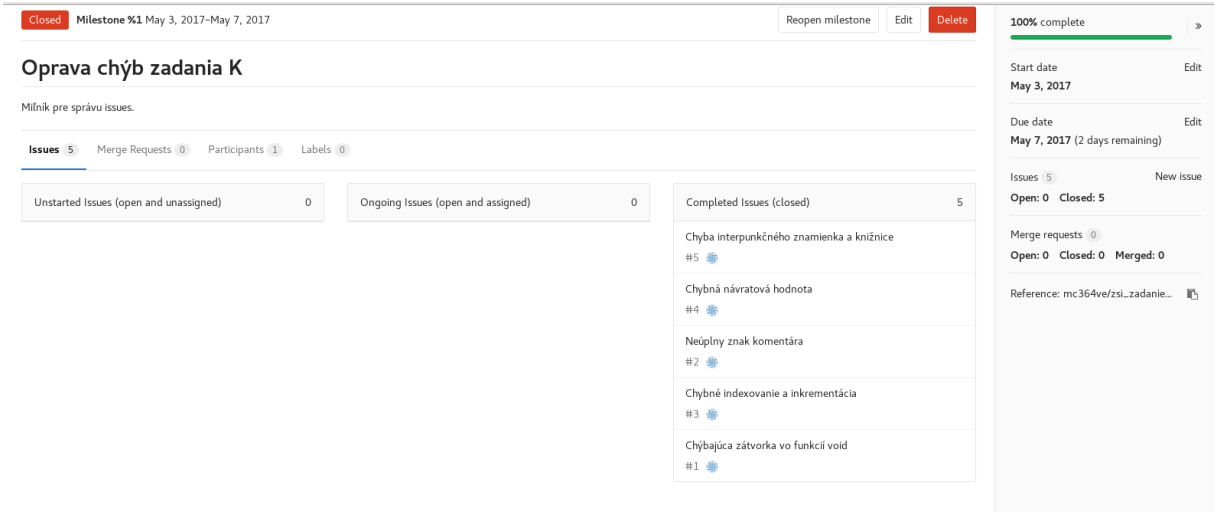
```
37      - void is_move_possible(const struct game game) {
    37  + bool is_move_possible(const struct game game) {
```

Martin Chlebovec @mc364ve closed a day ago

5. chyba interpunkčného znamienka a nenalinkovaná lokálna knižnica – posledným zisteným problémom v kóde súboru k.c bolo objavenie chýb v podobe nezapísaného interpunkčného znamienka – bodkočiarky za premennou typu int, ktorá do hry pridávala pri pohybe hráča na voľnú pozíciu herný kameň s písmenom A alebo B. **Chybný zápis premennej: int tile = 'A' + (rand() % 2).** Problém bolo veľmi jednoduché vyriešiť doplnením bodkočiarky na koniec riadku. Druhým problémom zistením a zaradeným do issue bolo nenalinkovanie príslušnej lokálnej knižnice k.h. Program bez tejto knižnice nebude poznať funkcie definované v súbore k.c tento problém bol zistený aj v súbore main.c, kde samotný program beží a odkazuje sa nielen na knižnicu k.h ale aj na hof.h. **Problém bol vyriešený doplnením knižnice v lokálnom tvare cez dvojicu úvodzoviek. Problém bol uzavretý a doplnený komentárom: Opravená chyba v súbore k.c za premennou int tile doplnená bodkočiarka ";" a nalinkovaná lokálna knižnica k.h v hlavičke súboru.**
Zmena vykonaná v zázname: b4cc88b2
Chybu opravil: @mc364ve



Po opravení všetkých issues a ich uzavretí bolo možné uzavrieť aj míľnik predčasne. Míľnik ponúka prehľad stavu riešenia problémov a taktiež aj prehľadný výpis problémov čakajúcich na vyriešenie, respektíve aj tých ukončených. Aj po uzavretí je ho možné znovu otvoriť, ak to dovoľuje deadline pre prípadné úpravy či znovuotvorenie niektorého z problémov.



Záver a zhrnutie zadania

V zadaní č. 4 sme sa naučili efektívne využívať Gitlab aj na správu problémov v míľniku. Pre vyhotovenie zadania som si vymenil kód so spolužiakom Richardom Hojstričom a vykonali sme si inšpekciu kódu jeden druhému. Zadanie som realizoval s využitím softvérových nástrojov PsPad, NetBeans IDE pre kontrolu kódu. NetBeans IDE mi pomohol identifikovať 2 chyby, ktoré som si pri prebehnutí kódu očami nevšimol. V skutočnosti sa jednalo o chyby, ktoré úplne znefunkčnili kód, bola to bodkočiarka a chýbajúca množinová zátvorka. Najzaujímavejšou vecou pri inšpekcii kódu pre mňa bolo sledovať, ako dokáže maličkosť – bodkočiarka či zátvorka úplne znefunkčniť kód. Jeden chýbajúci symbol dokáže narobiť fatálne problémy, najmä ak sa vo funkcii vykonáva viacero rozhodovaní a zátvorky sa chybou jednej zátvorky posunú a kompilátor hlási chyby tam, kde nie sú. Najnáročnejšie bolo všímať si malé detaily v kóde, napríklad deklarácie jednoduchých premenných. Mnohým chybám, na ktoré som so spolužiakom prišiel sa dá predísť využívaním vhodného IDE nástroja pre vývoj softvéru, ktorý už pri predpise funkcie dokáže automaticky vytvoriť zátvorky, do ktorých sa vpisuje funkčný kód danej funkcie. Na odhalenie problémov pre inšpekciou kódu je možné použiť prepínače kompilátora, napríklad -Werror – Wall ale aj iné, ktoré dokážu odhaliť chyby, duplicitné deklarácie, či úniky v pamäti (valgrind). Inšpekciou kódu som sa naučil lepšie chápať kódu a naučil som sa využívať pokročilejšie komentáre k funkciám, ktoré využíval spolužiak. Nebol pre mňa problém pochopiť čo daná funkcia robí. Komentár bol napísaný prehľadne so všetkými potrebnými informáciami. Gitlab ponúka diff pre zobrazenie zmien oproti poslednému commitu, resp. medzi dvoma verziami súboru. Moje opravy chýb som riešil v jednom commite.