Hodnoceno kým:	Bernhauer David
Celkové procentní hodnocení:	86.50 %
Celkem bodů:	**4.33**

[20 %] Členění do funkcí přiměřené délky:
Většínou, některé funkce jsou příliš dlouhé (70 %)
[15 %] Opakující se kód:
Není, vyčleněn do vhodných funkcí (100 %)
[20 %] Globální proměnné:
Vůbec nebo rozumně použité (tabulky s konstantami, ...) (100 %)
[15 %] Použití vhodných syntaktických konstrukcí:
Občas příliš komplikované podmínky nebo příliš zanořené konstrukce (70 %)
[10 %] Identifikátory:
Drobné výhrady (výstižné názvy, konzistentní pojmenování, ale názvy nejsou v AJ) (70 %)
[10 %] Formátování kódu:
Bez výhrad, konzistentní a přehledné (100 %)
[10 %] Komentáře:
Rozumné, relevantní, v AJ (100 %)
Poznámka:

Řádky 10:0-17:0
Vhodnější vytvoření struktury
```
struct interval
{
    int from;
    int includeFrom;
    int to;
    int includeTo;
};
```
Kategorie: Dělení do funkcí
Komentář: Struktura nepotřebuje mít "include" varianty, stačilo by vhodně upravit "from" a "to". Celý problém by se tím zjednodušil.

Řádky 27:0-27:69
Návrh funkce
```
void numOfStudents(struct interval *i1, int points[], int pointsSize)
```
Kategorie: Dělení do funkcí
Komentář: Funkce by měla vždy pokud možno něco vracet, tady je špatně, že vypisuje "sumu". Tu "sumu" by měla jen vracet a vypisovat by se měla jinde. Takto funkce není unovupoužitelná.

Řádky 61:4-61:44
Nevhodný cyklus
```
for (size = 0; size <= capacity; size++)
```
Kategorie: Syntaktické konstrukce
Komentář: Pro tuto úlohu se určitě nehodí "for", ani nedává zde smysl, podmínka "size <= capacity" je splněna vždy. Hlavně nedává smysl, proč je ve "for" také věc, která by mohla být napsané před "for" (to konkrétně načtení otevírací závorky).

Řádky 78:20-78:28
Funkce "exit"
```
exit(1);
```
Kategorie: Jiné
Komentář: Použití exit je principiálně špatně. Program je násilně ukončen, není uvolněna paměť. Vždy musíme z funkce vrátit nějakou hodnotu, která značí chybu, a v "main" ji zpracovat.

Řádky 94:16-94:43
Kombinace více přístupů
```
while ((getchar()) != '\n')
```
Kategorie: Jiné
Komentář: Není dobré kombinovat více přístupů k načítání/vypisování. Tady "scanf" a "getchar".

Řádky 109:0-111:9
"if" -> "else"
```
}
if (scanfResult != 2)
{
```
Kategorie: Syntaktické konstrukce
Komentář:

Řádky 164:0-176:67
Špatný komentář a funkce
```
/**
 * Reads the interval from the standard input and validates it.
 *  Reading is not successful:
 * - if interval not begins and ends with brackets <> or ()
```

```
 * - if numbers are not intigers
 * - if numbers are not divided with semicolon
 *
 * @param[in] points array of numbers
 * @param[in] i1 interval
 * @param[in] pointsSize size of an array
 * @return 1 if interval is valid, otherwise 0
 */
int scanInterval(struct interval *i1, int points[], int pointsSize)
```

Kategorie: Komentáře

Komentář: Tohle není pravda, ta funkce nenačte jeden interval, načítá intervaly ve while cyklu. Ale ano správně by měla být funkce, která načte jen jedno číslo. Navíc se návratová hodnota nikde nekontroluje.

Řádky 185:8-185:40

Není potřeba načítat newline, stačí na začátek formátovacího řetězce dát mezeru.

```
scanResult = scanf("%c%d;%d%c%c", &frontBracket, &i1->from, &i1->to, &endBracket, &newline);
```

Kategorie: Jiné

Komentář:

Řádky 223:0-227:21

Zbytečné nastavování

```
struct interval i1;
i1.from = 0;
i1.to = 0;
i1.includeFrom = 0;
i1.includeTo = 0;
```

Kategorie: Syntaktické konstrukce

Komentář:

Řádky 236:0-236:0

Celkový pohled

Kategorie: Jiné

Komentář: Obecně rozdělení do funkcí hodně pokulhává, jsou použity, ale ne dobře. Chybí návratové hodnoty nebo jejich zpracování, jedna funkce dělá více věcí. Zcela chybí struktura pro pole, která by se tu hodila.