**Project 1**

Title

**Calculation Game**

Course

**CIS-17A**

Section

**44051**

Due Date

**February 17, 2017**

Author

**Nhan Thanh Mac**

**Table of Contents**

[**1. Introduction 3**](#_Toc480139597)

[**2. Development summary 3**](#_Toc480139598)

[**3. Description 3**](#_Toc480139599)

[**a) Input and Output 3**](#_Toc480139600)

[**b) Flowcharts 5**](#_Toc480139601)

[**c) Variables 5**](#_Toc480139602)

[**d) Concepts 8**](#_Toc480139603)

[**4. References 8**](#_Toc480139604)

[**5. Program 8**](#_Toc480139605)

# Introduction

Calculation game is a game that measure how long player calculate a math. The game will automatic create mathematical operation such as addition, subtraction, multiplication, division. After that, the player will enter the result of operation. If the player enter correct result, he/she will receive a point based on the time they enter the result. The less time they have, the more points they get. Otherwise, the longer they have, the less points they get. In case, they player enter incorrect result, the game will be over and print the score they have. Then, the program will ask the player enter their name and store the scores to file. However, the game just stores 5 player who had highest score.

# Development summary

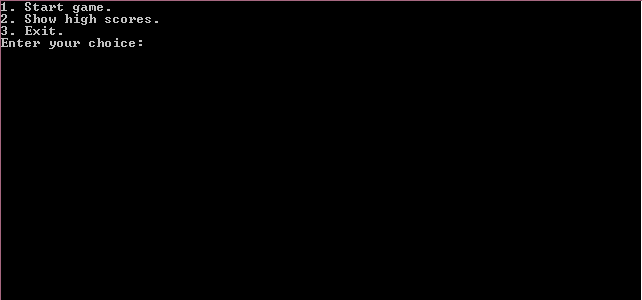
|  |  |
| --- | --- |
| 1. Project size | 428 lines (including comment line) |
| Global variable | 7 |
| The number of method | 12 |
| The number of structure | 1 |

The project included many concepts from the chapter 9, 10, 11, and 12 in textbook. When the program was coding, it just one thing I had to refer from the internet that is how to measure the time of a process. I had spent a day to have an idea about a game without graphic. After having idea, I spent 2 days to finish the program. I am not satisfied with this program, but it was a good experienced and revised what I learned from the chapters 9, 10, 11, 12 in textbook.

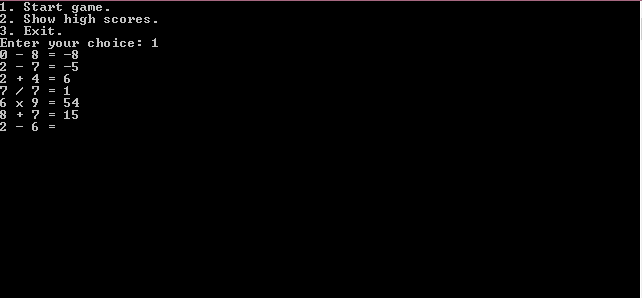
# Description

## Input and Output

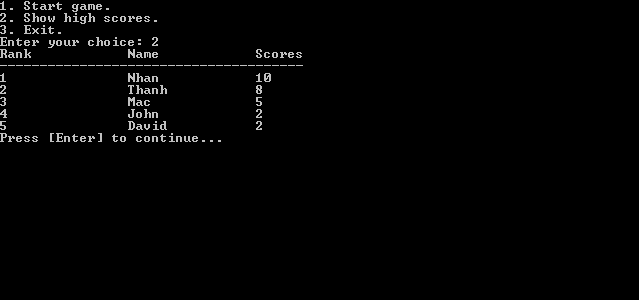
This is a game without graphic, so it will be displayed in command prompt. Player will play the game with only keyboard, no mouse. From the beginning, the game will ask player that they want to play game, show the high scores, or exit the game.



Player will enter the number that they want the game will execute. If player enter 1 and then press [Enter], the game will be started. After that, the game will show the mathematical operation, and the player enter the result for this. The program will be continue until the player win or lose.



In the second choice, if the player enter 2 from the menu. The game will show 5 player who had the highest score.



In the last choice, if the player enter 3, the game will be closed.

## Flowcharts

[**Flowcharts.pdf**](Flowcharts.pdf)

## Variables

* **Constant variables**

|  |  |
| --- | --- |
| const char OPERATORS[4] = {'+', '-', '\*', '/'} | Contain 4 operators: add,sub,div,multi |
| const int NUMBER\_OF\_LEVEL = 3 | The number of level |
| const int ROUND\_EACH\_LEVEL = 10 | Round for each level |
| const int SIZE\_SCORE\_TO\_FILE = 5 | Maximum recored scores to write down file |
| const string FILE\_NAME = "Scores.dat" | The name of file will save the high score |
| const int SIZE\_TIME\_SCORES = 5 | How many level of points |
| const int SCORE\_TIME[SIZE\_TIME\_SCORES][2] = {  {0, 10},  {1, 8},  {2, 6},  {3, 4},  {4, 2}  } | Array stored the points for each timing - Description: [time][points] |

* **startGame() method variables**

|  |  |
| --- | --- |
| int level | The level of game |
| int totalScores | The points that player receives if they enter correct answer |
| bool isGameOver | Flag notices that game is over |
| int time | The time in second from begin input to end input |
| Scores score | Structure stores information of player |
| string name | Name of player |

* **calcTime() method variables**

|  |  |
| --- | --- |
| time\_t start | Representing time begin of process |
| time\_t end | Representing time end of process |
| int dif | The difference time between start of process and end of process |
| int &result | The value number receive from input |

* **getRandomNumber()method variables**

|  |  |
| --- | --- |
| int min | The minimum of range number |
| int max | The maximum of range number |
| int number | Random number |

* **checkPrimeNumber()method variables**

|  |  |
| --- | --- |
| int number | The number will be checked prime |
| bool isPrime | Flag notice that the number is prime or not |

* **generationOperation()method variables**

|  |  |
| --- | --- |
| int level | The level of game |
| int number1 | The first random number |
| int number2 | The second random number |
| char oper | Character contains operator |
| int result | The result mathematical operation of first number and second number |
| int time | The time in second |

* **callMainMenu()method variables**

|  |  |
| --- | --- |
| int level | The level of game |
| int choice | The choice uses to select menu |

* **save()method variables**

|  |  |
| --- | --- |
| string fileName | The name of file |
| Scores score | Structure stores information of player |
| const int size = 5 | Maximum record |
| Scores \*sFile | The size of record will be stored in file |
| fstream in | Object performs input operation file |
| int i | Counting the record will be store to file |
| bool isChange | Flag notice that have new high score |
| Scores temp | Structure store a record from file |
| fstream out | Object performs output operation file |

* **printScoreFromFile()method variables**

|  |  |
| --- | --- |
| Scores \*sFile | Array stores records high score from file |
| string fileName | The name of file |
| const int size = 5 | Maximum record |
| Scores score | Structure stores information of player |
| fstream file | Object performs input operation file |

* **cinValidNumber()method variables**

|  |  |
| --- | --- |
| float temp | Storing the value from player’s input |
| bool isCorrect | Flag notice that the input is correct. |

## Concepts

* Chapter 9:
  + The relationship between arrays and pointers
  + Dynamic memory allocation
* Chapter 10:
  + C-string
  + C++ string class
* Chapter 11:
  + Structure
  + Arrays of structure
  + Structures as function arguments
  + Returning a Structure from a function
* Chapter 12:
  + File operations
  + Reading and Writing files
  + Binary files
  + Creating records with Structures

1. **References**

* Textbook
* stackoverflow.com

1. **Program**
2. #include <cstdlib>
3. #include <time.h>
4. #include <iostream>
5. #include <string>
6. #include <fstream>
7. **using** **namespace** std;
8. #include "Scores.h"
9. /\*
10. \* A group of function prototype
11. \*/
12. **int** calcTime(**int** &);
13. **void** callMainMenu(**int** level);
14. **bool** cinValidNumber(**int** &);
15. **bool** cinValidNumber(**float** &);
16. **int** getRandomNumber(**int** min, **int** max);
17. **char** getRandomOperations();
18. **int** generationOperation(**int** level);
19. **void** startGame(**int**);
20. **void** save(string, Scores, **const** **int**);
21. **void** printScoreFromFile(string, **const** **int**);
22. **bool** checkPrimeNumber(**int**);
23. **int** countScores(**int**);
24. /\*
25. \*  Constant
26. \*/
27. **const** **char** OPERATORS[4] = {'+', '-', '\*', '/'}; /\*!< Contain 4 operators: add,sub,div,multi \*/
28. **const** **int** NUMBER\_OF\_LEVEL = 3; /\*!< The number of level \*/
29. **const** **int** ROUND\_EACH\_LEVEL = 10; /\*!< Round for each level \*/
30. **const** **int** SIZE\_SCORE\_TO\_FILE = 5; /\*!< Maximum recored scores to write down file \*/
31. **const** string FILE\_NAME = "Scores.dat"; /\*!< The name of file will save the high score \*/
32. **const** **int** SIZE\_TIME\_SCORES = 5; /\*!< How many level of points \*/
33. **const** **int** SCORE\_TIME[SIZE\_TIME\_SCORES][2] = {
34. {0, 10},
35. {1, 8},
36. {2, 6},
37. {3, 4},
38. {4, 2}
39. }; /\*!< Array stored the points for each timing - Description: [time][points] \*/
41. **int** main(**int** argc, **char**\*\* argv) {
42. /\* initialize random seed: \*/
43. srand(time(NULL));
44. **int** level = 1;
45. callMainMenu(level);
46. **return** 0;
47. }
49. /\*!
50. \* \brief startGame function is a function will execute the game
51. \* and print the result of game when user win or lose the game
52. \* \param level current level of game
53. \* \return none
54. \*/
55. **void** startGame(**int** level) {
56. //init total score from beginning
57. **int** totalScores = 0;
58. **bool** isGameOver = **false**;
59. //loop number of level
60. **for** (**int** i = 0; i < NUMBER\_OF\_LEVEL; i++) {
61. //loop round for each level
62. **for** (**int** j = 1; j <= ROUND\_EACH\_LEVEL \* level; j++) {
63. //create operator-->input from player-->count time
64. **int** time = generationOperation(level);
65. //if time  < 0 --> game over
66. **if** (time >= 0) {
67. totalScores += countScores(time);
68. } **else** {
69. //game over
70. isGameOver = **true**;
71. **break**;
72. }
73. }
74. //game over
75. **if** (isGameOver) {
76. **break**;
77. }
78. //passed level
79. cout << "Congratulation! You passed level " << level << "." << endl;
80. level++;
81. cin.ignore();
82. cout << "Press [Enter] to continue next level ...";
83. cin.get();
84. }
85. Scores score;
86. string name;
87. score.scores = totalScores;
88. //game over
89. **if** (isGameOver) {
90. //print info
91. cout << "GAME OVER!" << endl;
92. cout << "Your scores: " << totalScores << endl;
93. cout << "Enter your name: ";
94. cin >> name;
95. score.name = name;
96. //save score to file
97. save(FILE\_NAME, score, SIZE\_SCORE\_TO\_FILE);
98. cout << "Press [Enter] to play again.";
99. cin.ignore();
100. cin.get();
101. level = 1;
102. callMainMenu(level);
103. } **else** {
104. //complete game
105. cout << "Congratulation! You won!" << endl;
106. cout << "Your scores: " << totalScores << endl;
107. cout << "Enter your name: ";
108. cin >> name;
109. score.name = name;
110. save(FILE\_NAME, score, SIZE\_SCORE\_TO\_FILE);
111. cin.ignore();
112. cout << "Press [Enter] to play again.";
113. cin.get();
114. callMainMenu(level);
115. }
116. }
118. /\*!
119. \* \brief return the points for user
120. \* depend on the time user enter their result.
121. \* \param time The time in second from begin input to end input
122. \* \return The score
123. \*/
124. **int** countScores(**int** time) {
125. **for** (**int** i = 0; i < SIZE\_TIME\_SCORES; i++) {
126. //SCORE\_TIME[i][0] description [time][point]
127. **if** (SCORE\_TIME[i][0] == time)
128. **return** SCORE\_TIME[i][1];
129. }
130. **return** 0;
131. }
133. /\*\*
134. \* \brief Get time in second from begin input to end input
135. \* @param result the value from input
136. \* @return the time in second
137. \*/
138. **int** calcTime(**int** &result) {
139. **time\_t** start, end;
140. //current time start
141. time(&start);
142. cinValidNumber(result);
143. //current time end
144. time(&end);
145. //different time
146. **int** dif = difftime(end, start);
147. **return** dif;
148. }
150. /\*!
151. \* \brief Create a random number in range
152. \* \param min min value in range
153. \* \param max max value in range
154. \* \return a random number
155. \*/
156. **int** getRandomNumber(**int** min, **int** max) {
157. **int** number;
158. //random number in range [min-max]
159. number = (rand() % (max - min + 1)) + min; //
160. **return** number;
161. }
163. /\*!
164. \* \brief Get a random operator from array char OPERATORS[4] = {'+', '-', '\*', '/'}
165. \* \return a character '+' or '-' or '\*' or '/'
166. \*/
167. **char** getRandomOperations() {
168. **return** OPERATORS[getRandomNumber(0, 3)];
169. }
171. /\*!
172. \* \brief Checking a number is a prime number or not
173. \* \param number number will be checked
174. \* \return True: is a prime number | False: is not a prime number
175. \*/
176. **bool** checkPrimeNumber(**int** number) {
177. **bool** isPrime = **true**;
178. //loop to number/2
179. **for** (**int** i = 2; i < number / 2; i++) {
180. **if** (number % i == 0) {
181. isPrime = **false**;
182. **break**;
183. }
184. }
185. **return** isPrime;
186. }
188. /\*!
189. \* \brief Create a Mathematical operation, receive the input from user, calculate the time,
190. \* and check the result from input of user
191. \* \param level The level of game
192. \* \return The time of process
193. \* \note if the result of user is incorrect, the function will return the time = -1
194. \*/
195. **int** generationOperation(**int** level) {
196. **int** number1;
197. **int** number2;
198. **char** oper = getRandomOperations();
199. //generation number depend on level
200. **if** (level == 3) {
201. number1 = getRandomNumber(10, 99);
202. number2 = getRandomNumber(0, 99);
203. } **else** **if** (level == 2) {
204. number1 = getRandomNumber(10, 99);
205. number2 = getRandomNumber(0, 9);
206. } **else** {
207. number1 = getRandomNumber(0, 9);
208. number2 = getRandomNumber(0, 9);
209. }
211. //generation operations and count the time when begin input to end input
212. **int** result;
213. **int** time;
214. **switch** (oper) {
215. //if operator is +
216. **case** '+':
217. //print result;
218. cout << number1 << " + " << number2 << " = ";
219. //count time
220. time = calcTime(result);
221. //check result
222. **if** (result == (number1 + number2))
223. **return** time;
224. **else**
225. **return** -1;
226. **break**;
227. //if operator is -
228. **case** '-':
229. //print result;
230. cout << number1 << " - " << number2 << " = ";
231. //count time
232. time = calcTime(result);
233. //check result
234. **if** (result == (number1 - number2))
235. **return** time;
236. **else**
237. **return** -1;
238. **break**;
239. //if operator is \*
240. **case** '\*':
241. // print result;
242. cout << number1 << " x " << number2 << " = ";
243. //count time
244. time = calcTime(result);
245. //check result
246. **if** (result == (number1 \* number2))
247. **return** time;
248. **else**
249. **return** -1;
250. **break**;
251. //if operator is /
252. **case** '/':
253. //find number2 while it not a zero based-on level
254. **if** (number2 == 0) {
255. **if** (level == 3) {
256. **do** {
257. number2 = getRandomNumber(0, 99);
258. } **while** (number2 == 0);
259. } **else** {
260. **do** {
261. number2 = getRandomNumber(0, 9);
262. } **while** (number2 == 0);
263. }
264. }
265. **if** (number1 % number2 != 0) {
266. **if** (level == 3 || level == 2) {
267. //create number 1 is not a prime number
268. **do** {
269. number1 = getRandomNumber(0, 99);
270. } **while** (!checkPrimeNumber(number1) || number1 % number2 != 0);
271. } **else** {
272. //create number 1 is not a prime number
273. **do** {
274. number1 = getRandomNumber(0, 9);
275. } **while** (!checkPrimeNumber(number1) || number1 % number2 != 0);
276. }
277. }
278. cout << number1 << " / " << number2 << " = ";
279. time = calcTime(result);
280. **if** (result == (number1 / number2))
281. **return** time;
282. **else**
283. **return** -1;
284. **break**;
285. }
286. }
288. /\*!
289. \* \brief Create a menu for choosing
290. \* 1. Start game
291. \* 2. Show high score
292. \* 3. Exit
293. \* \param level the level of game
294. \* \return none
295. \*/
296. **void** callMainMenu(**int** level) {
297. **int** choice = -1;
298. //loop while choice is correct
299. **do** {
300. //print list of choice
301. **do** {
302. cout << "1. Start game." << endl;
303. cout << "2. Show high scores." << endl;
304. cout << "3. Exit." << endl;
305. cout << "Enter your choice: ";
306. } **while** (!cinValidNumber(choice));
307. } **while** (choice < 1 || choice > 3);
308. **switch** (choice) {
309. //if choice == 1 --> start game
310. **case** 1:
311. startGame(level);
312. **break**;
313. **case** 2:
314. //choice == 2 --> print the high score
315. printScoreFromFile(FILE\_NAME, SIZE\_SCORE\_TO\_FILE);
316. cin.ignore();
317. cout << "Press [Enter] to continue...";
318. cin.get();
319. callMainMenu(level);
320. **break**;
321. **case** 3:
322. exit(0);
323. **break**;
324. **default**:
325. cout << "Press [Enter] to continue...";
326. cin.ignore();
327. cin.get();
328. callMainMenu(level);
329. }
330. }
332. /\*!
333. \* \brief Save the high score to file. The function will get the scores
334. \* that are stored in file and check the current score. If the current score is
335. \* greater than the other scores in file. The function will insert the current score
336. \* and store those to file.
337. \* \param fileName the name of file stores the high score
338. \* \param score the current score
339. \* \param size the size of record will be stored in file
340. \* \return none
341. \*/
342. **void** save(string fileName, Scores score, **const** **int** size = 5) {
343. //create a array to store high score
344. Scores \*sFile = **new** Scores[size];
345. fstream in;
346. //open document
347. in.open("Scores.dat", ios::in | ios::binary);
348. //read the first element
349. in.read(**reinterpret\_cast**<**char** \*> (&sFile[0]), **sizeof** (sFile[0]));
350. **int** i = 1;
351. //loop to end file && < size
352. **while** (!in.eof() && i < size) {
353. //read next element
354. in.read(**reinterpret\_cast**<**char** \*> (&sFile[i]), **sizeof** (sFile[i]));
355. i++;
356. }
357. **bool** isChange = **false**;
358. Scores temp;
359. //loop to each element high score
360. **for** (**int** j = 0; j < size; j++) {
361. //check if have new high score
362. **if** (!isChange) {
363. //current score < new score
364. **if** (sFile[j].scores < score.scores) {
365. score.rank = sFile[j].rank;
366. temp = sFile[j];
367. sFile[j] = score;
368. isChange = **true**;
369. }
370. } **else** {
371. Scores temp1 = sFile[j];
372. sFile[j] = temp;
373. temp = temp1;
374. }
375. }
376. in.close();
377. //write to file
378. fstream out;
379. out.open("Scores.dat", ios::out | ios::binary);
380. out.write(**reinterpret\_cast**<**char** \*> (&sFile[0]), **sizeof** (sFile[0]));
381. i = 1;
382. **while** (!out.eof() && i < size) {
383. out.write(**reinterpret\_cast**<**char** \*> (&sFile[i]), **sizeof** (sFile[i]));
384. i++;
385. }
386. out.close();
387. **delete** []sFile;
388. //printScoreFromFile(fileName, SIZE\_SCORE\_TO\_FILE);
389. }
391. /\*!
392. \* \brief Printing the high score that stored in file.
393. \* \param fileName the name of file stores the high score
394. \* \param size the size of record will be stored in file
395. \* \return none
396. \*/
397. **void** printScoreFromFile(string fileName, **const** **int** size = 5) {
398. Scores score;
399. fstream file;
400. file.open("Scores.dat", ios::in | ios::binary);
401. file.read(**reinterpret\_cast**<**char** \*> (&score), **sizeof** (score));
402. cout << "Rank\t\tName\t\tScores" << endl;
403. cout << "--------------------------------------" << endl;
404. **while** (!file.eof()) {
405. cout << score.rank << "\t\t" << score.name << "\t\t" << score.scores << endl;
406. file.read(**reinterpret\_cast**<**char** \*> (&score), **sizeof** (score));
407. }
408. file.close();
409. }
411. /\*!
412. \* \brief Checking the input from user. Only check integer data type.
413. \* \param number store the input from user
414. \* \return none
415. \*/
416. **bool** cinValidNumber(**int** &number) {
417. **float** temp;
418. cin >> temp;
419. number = **static\_cast**<**int**> (temp);
420. **bool** isCorrect = **true**;
421. **if** (cin.fail()) {
423. cin.clear();
424. cin.ignore(80, '\n');
425. isCorrect = **false**;
426. }
427. **return** isCorrect;
428. }