CROSSWORD PUZZLE

CO222-PROJECT REPORT

GROUP 35

DISSANAYAKE D.M.I.G. (E/19/090)

WICKRAMASINGHA G.T.G. (E/19/436)

Project Overview

- The task of the project was to create a puzzle solver algorithm when the words and the empty/half-filled puzzle is given using C programming language.

- The objectives of the project were to identify and discuss the differences between static and dynamic memory allocation.

- A brief description on how the program works is given below.
  First , the empty grid is taken into 1D array, and the words are stored in a 2D array. Then each word is checked to see if they are fitted into grid boundaries.
  Next, the word is checked with each row and column to find a suitable position and if there is any , the word is fitted in there.
  If a word is fitted into wrong place, a backup of the puzzle before filling each word is kept to use in backtracking algorithm
  Backtracking algorithm is used to go back to previous states until the incorrectly filled word is found and another word is assigned in there and keep filling the puzzle again.

- This project was done in two phases. In Phase 1 project was completed using static memory allocation. Main issue of this method was it takes more space to statically declare arrays using c. To avoid this issue ,phase 2 was designed using dynamic memory allocation.

Introduction

- In this report the execution time and memory space usage differences between two programs written in C programming language is compared. Phase 1 of the project uses static memory allocation, while phase 2 uses dynamic memory allocation.

1)Comparison of the execution times

Method:

- We have utilized the C language's clock() function in <time.h> header file, which returns the total number of clock ticks since the program was started, to calculate the execution time of two programs.

```c
// main function
int main()
{
    clock_t time;
    long double cpu_time;
    time = clock();
```

```c
time = clock() - time;
cpu_time = (((long double)time) / CLOCKS_PER_SEC);
printf("\n%Lf ms\n", cpu_time);
```

Results:

- Test case #1

```
kalindu@DESKTOP-R08BL8B:~/Testingforthereport$ ./ineshast <puzzel2
****
FIRE
****
*CAT

62000.000000 ns
kalindu@DESKTOP-R08BL8B:~/Testingforthereport$ ./ineshady <puzzel2
****
FIRE
****
*CAT

73000.000000 ns
```

- Test case #2

```
kalindu@DESKTOP-R08BL8B:~/Testingforthereport$ ./ineshady <puzzel3
*F**
GLUE
*Y**
****

97000.000000 ns
kalindu@DESKTOP-R08BL8B:~/Testingforthereport$ ./ineshast <puzzel3
*F**
GLUE
*Y**
****

85000.000000 ns
kalindu@DESKTOP-R08BL8B:~/Testingforthereport$
```

- Test case #3

```
kalindu@DESKTOP-R08BL8B:~/Testingforthereport$ ./ineshady <puzzel4
*F*SON
GLUE**
*Y*ANT
****O*
PYTHON
******

112000.000000 ns
kalindu@DESKTOP-R08BL8B:~/Testingforthereport$ ./ineshast <puzzel4
*F*SON
GLUE**
*Y*ANT
****O*
PYTHON
******

91000.000000 ns
kalindu@DESKTOP-R08BL8B:~/Testingforthereport$
```

- Test case #4

```
kalindu@DESKTOP-R08BL8B:~/Testingforthereport$ ./ineshast <puzzel8
******I***
**MEXICO**
******E***
******L***
***PANAMA*
******N*L*
******D*M*
********A*
********T*
********Y*

99000.000000 ns
kalindu@DESKTOP-R08BL8B:~/Testingforthereport$ ./ineshady <puzzel8
******I***
**MEXICO**
******E***
******L***
***PANAMA*
******N*L*
******D*M*
********A*
********T*
********Y*

161000.000000 ns
```

Conclusion on execution time:

- It has been discovered that the program utilizing static memory allocation ran a little bit more quickly than the version using dynamic memory allocation. This is because the dynamic memory allocation occurs at runtime, whereas the static memory allocation occurs at compile time. Also in dynamic memory allocation, an additional time is spent to free memory blocks inside the program. However, in most instances, the change in execution time was determined to be insignificant.

2) Comparison of the memory space usage

- Static memory is defined as memory allocated during the compilation process. The compiler in this allocates RAM for the variables. Array declaration is an example of static memory allocation. The size of the array must be known in advance. Arrays cannot be resized after memory has been allocated. This was a major drawback in using static allocation.

- Dynamic memory allocation refers to memory allocation that takes place during execution or runtime. Since the actual size of the data required is known at run time, the program is allocated the correct amount of memory, reducing memory waste.

Method:

- "Valgrind" was used to analyze the memory usage of both the programs. This can report two types of issues: memory errors and memory leaks.

Results:

- For static program

```
kalindu@DESKTOP-R08BL8B:~/Testingforthereport$ valgrind --leak-check=full ./ineshast <puzzel2
==132== Memcheck, a memory error detector
==132== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==132== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==132== Command: ./ineshast
==132==
****
FIRE
****
*CAT

10900000.000000 ns
==132==
==132== HEAP SUMMARY:
==132==     in use at exit: 0 bytes in 0 blocks
==132==   total heap usage: 2 allocs, 2 frees, 5,120 bytes allocated
==132==
==132== All heap blocks were freed -- no leaks are possible
==132==
==132== For lists of detected and suppressed errors, rerun with: -s
==132== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
kalindu@DESKTOP-R08BL8B:~/Testingforthereport$
```

- For dynamic program

```
kalindu@DESKTOP-R08BL8B:~/Testingforthereport$ valgrind --leak-check=full ./ineshady <puzzel2
==133== Memcheck, a memory error detector
==133== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==133== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==133== Command: ./ineshady
==133==
```

```
13615000.000000 ns
==133==
==133== HEAP SUMMARY:
==133==     in use at exit: 0 bytes in 0 blocks
==133==   total heap usage: 12 allocs, 12 frees, 5,204 bytes allocated
==133==
==133== All heap blocks were freed -- no leaks are possible
==133==
==133== For lists of detected and suppressed errors, rerun with: -s
==133== ERROR SUMMARY: 46 errors from 15 contexts (suppressed: 0 from 0)
kalindu@DESKTOP-R08BL8B:~/Testingforthereport$
```

Conclusion on memory usage:

- Referring to above data on memory usage, it can be concluded that the program employing dynamic memory allocation (Phase 2) was more effective in terms of memory space utilization. This is so that only the necessary amount of memory can be allocated, as opposed to static memory allocation, which allocates memory for the largest size feasible regardless of actual consumption.

Final Conclusion:

- Overall, there are advantages and disadvantages to both static and dynamic memory allocation. Dynamic memory allocation is more effective in terms of memory space use while static memory allocation gives faster execution time. It is crucial to take into account that a particular program needs and select the proper memory allocation strategy in accordance.

References

- https://web.stanford.edu/class/archive/cs/cs107/cs107.1232/resources/valgrind.html
- https://www.geeksforgeeks.org/difference-between-static-and-dynamic-memory-allocation-in-c/
- https://www.geeksforgeeks.org/clock-function-in-c-c/