School of Computing and Information Systems

comp20005 Intro. to Numeric Computation in C
Semester 1, 2023
Assignment 2

Learning Outcomes

In this project you will demonstrate your understanding of structures and arrays of structures, and will develop a computational solution for a non-trivial problem. You are expected to make extensive use of functions; and to demonstrate that you have adopted a clear and elegant programming style. You will find it difficult to complete unless you plan your program carefully, and develop it incrementally.

Noise!

Suppose that you are an employee of NoiseBeGone, a company that specializes in providing sound analysis for large industrial sites. An important design criteria in any new site is to understand the noise levels that will be generated by a given set of equipment, so that the impact on employees and nearby residents can be estimated, and, if necessary, noise barriers installed. Your task in this project is to design and implement a program that models the sound levels caused by a set of noise sources, assuming that there are no walls or other abatement devices currently present. The rest of this section gives some information about how sound intensities can be calculated, but is *not* part of the examinable material for this subject. That is, you may regard these computations as being "formula" that you are asked to incorporate into a program, but do not need to understand their derivation, or memorize them.

Noise is measured via *sound pressure*, the variation in air pressure above and below its equilibrium value. The measure most commonly used is the root-mean-square (rms) sound pressure p_{rms}. Because the range of sound pressure amplitude variations that the human ear can detect extends over several factors of ten, a compressed scale based on the logarithms is used, using *decibels*, abbreviated dB. The corresponding noise descriptor is called the *sound pressure level*, SPL, and is defined as

$$\text{SPL} = 10 \log_{10} \left(\frac{p_{\text{rms}}}{p_0} \right)^2 \text{dB} \,,$$

where p_0 is a reference pressure, standardized at 20 micropascals. This very small reference pressure corresponds to 0 dB and represents the weakest sound that can be heard by an average young alert person with an undamaged hearing mechanism.[1]

Since decibels are logarithmic measures, sound pressure levels cannot be added by ordinary arithmetic (that is, 30 dB + 25 dB $\neq$ 55 dB). The sound pressure level $\text{SPL}_{\text{total}}$ corresponding to the combination of n sound pressure levels SPL_i is calculated as

$$\text{SPL}_{\text{total}} = 10 \log_{10} \left(\sum_{i=1}^{n} 10^{\text{SPL}_i/10} \right) \text{dB} \,.$$

According to this formula, 30 dB + 25 dB = 31.2 dB.

The value of the sound pressure level depends on many factors such as the "strength" of the source, on the nature of the surroundings, on the distance from the source to the measurement point, and in some cases, on the direction. To measure the strength of a sound source W_i, the *sound power level* POW_i is defined as

$$\text{POW}_i = 10 \log_{10} \left(\frac{W_i}{W_0} \right) \text{dB}$$

[1] Note that the BlueTooth ear-bud had not been invented when these definitions were first conceived, and there are probably no more "young people with undamaged hearing mechanisms".

where W_i is the energy involved in generating the sound, measured in Watts, and W_0 is a reference power standardized at 10^{-12} watts. Note that these Watts do not (directly) correspond to loudspeaker ratings on TVs and other domestic appliances. For example, a chainsaw might have a sound power of around 0.1 Watts, and a typical conversation would generate 10^{-5} Watts.

For the i th source in a set of sources, the sound pressure level SPL_i and the sound power level POW_i are then related by

$$\text{SPL}_i = \text{POW}_i + 10 \log_{10} \left(\frac{Q}{4\pi r_i^2} + \frac{4}{R} \right) \text{dB}$$

where

SPL_i	=	sound pressure level due to i th source (dB relative to 20 micropascals)
POW_i	=	sound power level due to i'th source (dB relative to 10^{-12} watts)
Q	=	directivity factor (dimensionless)
R	=	surface absorption factor (m^2)
r_i	=	straight line distance to measurement point from i th source (m)

For a source located in an open grass field, which is the situation being considered in this project, $Q = 2$ and $R = (2 + \alpha)\pi r_i^2$, with the absorption coefficient $\alpha = 0.5$.

Stage 1 – Control of Reading and Printing (marks up to 8/20)

Input consists of lines each containing three numbers: x and y coordinates, measured in meters east and north from an assumed origin at $(0.0, 0.0)$; and a corresponding W value, which is the wattage associated with a sound emitter at the location specified by (x, y). At most 100 sound sources will be specified, and all (x, y) locations will be in the range $[0..74) \times [0..100)$ meters[2]. You may assume that the input is always free from errors, and just use `scanf("%lf%lf%lf",...)` to obtain the data triples. There are no header lines in the input files.

Write a program (including functions as appropriate) that reads the set of supplied triples x, y, W from standard input into an array of structure elements. Then write further functions, so that for the five sound sources in the test file `noisy1.txt`, the output exactly matches:

```
S1, number of sound sources: 5
S1, 30.0m E, 70.0m N, power 0.00050W, contributes 47.6 dB at origin
S1, 35.0m E, 63.0m N, power 0.00060W, contributes 48.9 dB at origin
S1, 36.5m E, 27.0m N, power 0.00080W, contributes 54.1 dB at origin
S1, 70.0m E, 25.0m N, power 0.00200W, contributes 53.8 dB at origin
S1, 20.0m E, 50.0m N, power 0.00250W, contributes 57.6 dB at origin
```

The "dB" number that you are to compute and print as part of this output is the sound pressure level caused by that sound source at the south-west corner of the land rectangle, the origin point $(0.0, 0.0)$.

Your program take its input from `stdin` (exactly the same requirement as Assignment 1). Do not print any prompts, and do not use any of the file commands described in Chapter 11 of the textbook.

Stage 2 – Sound Safety (marks up to 16/20)

Suppose we want to identify any "danger" spots, in which the SPL is greater than or equal to 80 dB (the value above which long-term hearing damage is likely to result; think again about how close your BlueTooth ear-plugs are to your ears). One approach is to logically overlay a regular grid, and evaluate the SPL level at each point in the grid.

Since the plant in question is in a field of size 74 meters "wide" (in the east-west direction) and 100 meters "tall" (in the north-south direction), with edges perfectly aligned east-west and north-south, you decide to use a *grid sampling* approach, using a grid size of 1.0 m.

[2] A land area of 0.74 hectares, or 1.8 acres, around 10–15 suburban Melbourne house lots.

Add to your program so that it evaluates the sound intensity at every whole-number-of-meters grid point strictly within the field and on its boundaries (that is, at the 7,575 points $x \in [0, 1 \ldots 74]$ combined with $y \in [0, 1 \ldots 100]$, and computes the percentage of those locations at which the sound intensity is ≥ 80 dB. Then, because you know your boss will then ask how confident you are, you also use a grid size of 0.5 m and 0.25 m. For `noisy1.txt` the output from this stage must be:

```
S2, grid = 1.00m, danger points =    140 /   7575 = 1.85%
S2, grid = 0.50m, danger points =    570 /  29949 = 1.90%
S2, grid = 0.25m, danger points =   2230 / 119097 = 1.87%
```

Note that you need to be a bit careful, because if you happen to test the exact (or a very close) location of one of the sound sources, you will get a zero radius and will then divide by zero. Such locations should simply be counted as "above 80" rather than allowed to result in program failure. You also need to be careful with the calculations that yield each grid position, and make sure that you visit all required points.

Stage 3 – Character Plots (marks up to 20/20)

Having finished your program for Stage 2, you proudly show it to your boss, only to have them say "that's good to know, but can you show me where the dangerous locations are?"

So you decide that the best way to show what is going on is to generate crude character map. Each character displayed represents a cell one meter wide (in the east-west direction) and two meters tall (in the north-south direction), with the 1:2 ratio (roughly) corresponding to the relative width and height of characters in a terminal font, and represents the sound pressure level at the center-point of that grid cell. For the field in question, your plot is thus 74 characters wide and 50 characters high, and the top-left character represents the point location $(0.5, 99.0)$ in meters.

Then, to show the sound level contours, you use the following relationship to convert the total decibels at the *center point* of that grid cell into a character to be displayed:

< 20	< 25	< 30	< 35	< 40	< 45	< 50	< 55	< 60	< 65	< 70	< 75	< 80	< 85	< 90	≥ 90
' '	'2'	' '	'3'	' '	'4'	' '	'5'	' '	'6'	' '	'7'	' '	'8'	' '	'9'

Figure 1 shows the expected output for `noisy1.txt`. For example, the first value to be output (in the top left corner) should correspond to the sound intensity at the point $(x, y) = (0.5, 99.0)$, since that point is at the center of the grid cell that runs spans $[0.0 \ldots 1.0]$ in the horizontal direction, and spans $[98.0 \ldots 100.0]$ in the vertical direction. Similarly, the bottom right cell represents the location $(73.5, 1.0)$, as the center of another 1×2 meter grid cell. The last line of output is also required.

You will want to develop your own test files too, rather than just test on my ones. If you develop an "entertaining" (that is, artistic) input file, send it to me, and I'll run it and post the output to the Discussion. Fame and eternal glory to the person who generates the most creative plot.

Modifications to the Specification

There are bound to be areas where this specification needs clarification or correction, and you should refer to the "Assignment 2" LMS Discussion regularly and check for possible updates to these instructions.

The Boring Stuff...

This project is worth **20% of your final mark**, and is due at **6:00pm on Friday 19 May**.

Submissions that are made after the deadline will incur penalty marks at the rate of two marks per day or part day late. Students seeking extensions for medical or other "outside my control" reasons should email `ammoffat@unimelb.edu.au` as soon as possible after those circumstances arise. If you attend a GP or other health care service as a result of illness, be sure to obtain a letter from them that describes your illness and their recommendation for treatment. Suitable documentation should be attached to **all** extension requests.

Submission: Your `.c` file must be uploaded to GradeScope via the LMS "Assignment" page. *Don't forget to include, sign, and date the Authorship Declaration that is required at the top of your program.*

Multiple submissions may be made; only the last submission that you make before the deadline will be marked. If you make any late submission at all, your on-time submissions will be ignored, and if you have not been granted an extension, the late penalty will be applied.

Marking Rubric: A rubric explaining the marking expectations is linked from the assignment's LMS page, and you should study that rubric very closely. Feedback, marks, and a sample solution will be made available approximately two weeks after submissions close.

Academic Honesty: You may discuss your work during your workshop, and with others in the class, but what gets typed into your program must be individual work, not copied from anyone else, and not developed jointly with anyone else. So, do **not** give hard copy or soft copy of your work to anyone else; do **not** "lend" your "Uni backup" memory stick to others for any reason at all; and do **not** ask others to give you their programs "just so that I can take a look and get some ideas, I won't copy, honest". The best way to help your friends in this regard is to say a very firm "**no**" if they ask to see your program, pointing out that your "**no**", and their acceptance of that decision, are the only way to preserve your friendship. See `https://academicintegrity.unimelb.edu.au` for more information. Note also that solicitation of solutions via posts to "tutoring" sites or online forums, whether or not there is payment involved, and whether or not you actually employ any solutions that may result, is also serious misconduct. In the past students have had their enrolment terminated for such behavior.

The LMS page links to a program skeleton that includes an Authorship Declaration that you must "sign" and date and include at the top of your submitted program. Marks will be deducted (see the rubric linked from the LMS page) if you do not include the declaration, or do not sign it, or do not comply with its expectations. A sophisticated program that undertakes deep structural analysis of C code identifying regions of similarity will be run over all submissions. Students whose programs are identified as containing significant overlaps will have substantial mark deductions applied for failure to comply with instructions, or risk being referred to the Student Center for possible disciplinary action, without further warning.

Nor should you post your code to any public location (`github`, `codeshare.io`, etc) while the assignment is active or prior to the release of the assignment marks.

And remember, programming is fun!

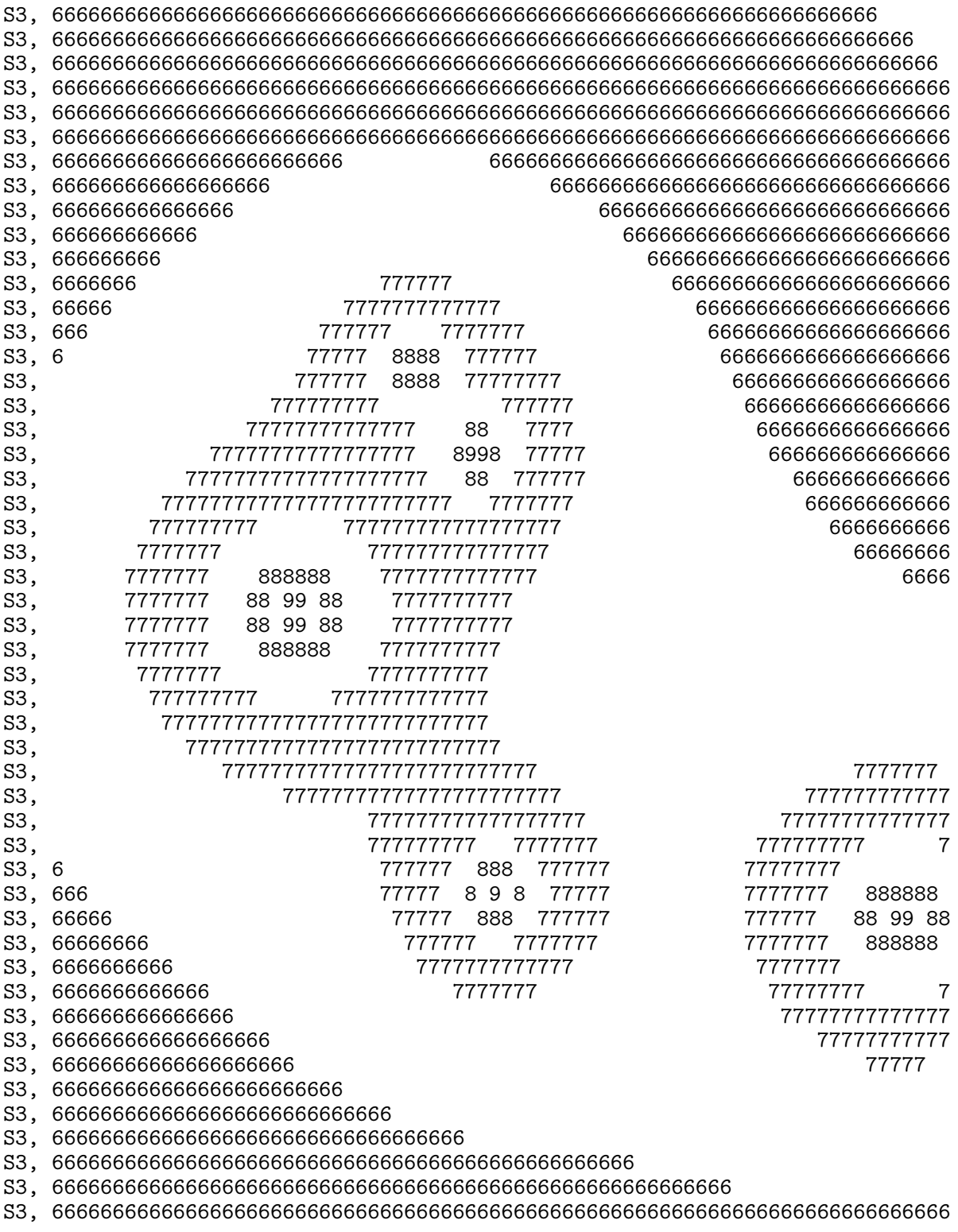

Figure 1: Stage 3 output for `noisy1.txt`.