Lab Exercises 15/04/2021: Packbits

Esercizio 1

Write a command line program in C++ with this syntax:

```
packbits [c|d] <input file> <output file>
```

When the "c" option is specified, the program opens the specified file (the file must be treated as a binary file, that is, it can contain any value from 0 to 255 in each byte), compresses it with the Packbits algorithm and saves it in the output file (since no header is defined, use ".packbits" extension). When the "d" option is specified, the program tries to decompress the file content.

If the input were:

```
aaaaabbbbbcdefghaaaaaaaaaaa
```

the output file would be (every box is a byte):

```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|252| a |252| b | 5 | c | d | e | f | g | h |246| A |128|
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

that, in a hexadecimal editor, would be:

```
FC 61 FC 62 05 63 64 65 66 67 68 F6 61 80
```

Implementation Notes
Initially treat each repetition as a run. Remember that nor runs, nor copies can be longer than 128 bytes.

General Important Notes
In general, solutions should avoid loading the entire file into memory or keep the entire sequence compressed or decompressed in memory. Everything should be done by reducing as much as possible the amount of data kept in RAM.

For the first solution it's acceptable to leave everything in memory, but try later to find a solution that does not require it. Try your algorithms with BIG files.

Esercizio 2

As a further extension of the previous exercise, create another version of the program that, if there is a copy of bytes followed by a 2-byte run, does not generate the run, but incorporates it into the copy, allowing for further copying. For example, consider the sequence:

```
xyaabcd
```

Compressed with the basic algorithm, it would become:

```
+---+---+---+---+---+---+---+---+---+---+
| 1 | x | Y |255| a | 2 | b | c | d |128|
+---+---+---+---+---+---+---+---+---+---+
```

With the considered variant, instead:

```
+---+---+---+---+---+---+---+---+---+
| 6 | x | y | a | a | b | c | d |128|
+---+---+---+---+---+---+---+---+---+
```

Leading to a 1 byte saving. Obviously, if the 2-byte run had been followed by another run, there would have been no savings.

General Important Notes
In general, solutions should avoid loading the entire file into memory or keep the entire sequence compressed or decompressed in memory. Everything should be done by reducing as much as possible the amount of data kept in RAM.

For the first solution it's acceptable to leave everything in memory, but try later to find a solution that does not require it. Try your algorithms with BIG files.