Seminar 12: Architecture

Today we are going to implement a simple data structure as an abstracted module with an opaque type.

Original code

Here is the code inside one `main` function that implements an expandable array (vector). Study it.

```
#include <inttypes.h>
#include <malloc.h>
#include <stdio.h>

int main() {
  int64_t *array = malloc(sizeof(int64_t) * 5);
  // capacity - how much memory is allocated
  size_t capacity = 5;
  // amount - how much memory is actually used from the allocated one
  size_t count = 0;
  // fill the array with squares of numbers from 0 to 100
  // if there is not enough space, we expand it twice
  for (size_t i = 0; i <= 100; i++) {
    if (count == capacity) {
      array = realloc(array, sizeof(int64_t) * capacity * 2);
      capacity = capacity * 2;
    }
    array[count++] = i * i;
  }

  for (size_t i = 0; i < 100; i++) {
    printf("%" PRId64 " ", array[i]);
  }
  return 0;
}
```

An expandable array, unlike a regular array, has a non-fixed size; elements can be added to the end of such an array. How it works:

- We allocate memory with a margin.
- We store two additional numbers: the number of allocated slots for elements and the number of filled slots in the array.
- If we have enough allocated slots, we add elements to the array, increasing the number of occupied slots.
- If the slots are no longer enough, then we increase the number of slots by 2 times. To do this, we use `realloc` to copy the allocated memory to the extended section.

Question Read the `man malloc` for the `realloc` function.

Task

Your task is to extract from this code at least a module with a vector implementation with a header file.

• The vector must be implemented as an opaque data structure (https://stepik.org/lesson/581687/step/6?unit=576408).

• The only access to its elements must be controlled and done through getter and setter.

• Try to reuse code as much as possible and not duplicate anything.

• Implement vector output as a separate function that accepts a FILE * into which you want to output its contents. This function can also be decomposed into a `foreach` function and a single item printer.

• Strive to make functions as small as possible.

You should decompose given program into several files.

The result program should be a program of several files that does the same as the given program. In `main` only filling the vector with numbers and calling a function that prints it to `stdout` should be implemented.

Read the "Rules for C" documents to make sure your code adheres to it.

Question 1. Send your archived program through the form.