Tartalomjegyzék

- 1 Exercises
  - 1.1 Centroid (from the previous practical)
  - 1.2 String slice
  - 1.3 Smaller than average
  - 1.4 Linked list
  - 1.5 Linked list: index
  - 1.6 Linked list: delete
  - 1.7 Linked list: insert
  - 1.8 Linked list: contains
  - 1.9 Two-way linked list.

Exercises

Open a new project for each exercise or a new file if you're in the command line.

From this point onward use .cpp file extensions!

Centroid (from the previous practical)

Write a C++ program that read an arbitrary number of 3 dimensional points (but at most 10). It then calculates and prints the centroid of the points (the coordinates' average). The input stops if it receives the (0, 0, 0) point. (This (0, 0, 0) won't be part of the calculation.)

Hint: no need for any complex data structure. It's enought to store the coordinates in 3 arrays and then calculate the average.

Once done, modify the code so the first number we read is the number of coordinates that will be entered afterward. This way we won't need to treat the (0, 0, 0) coordinate special. Example input:

```
3
3.2 -1.2 2.2
3.4 5.7 1.8
4.2 5.0 0.2
```

Once this is done as well, try to figure out a way to solve this without using any arrays.

String slice

Write a C++ function that takes a C string and two non-negative integers. The function returns the slice of the string between the two numbers (treated as indices). The returned string should be a new dynamically allocated char array (C string style, with the terminal '\0'). As an example the "http://wiki.math.bme.hubatman"http://wiki.math.bme.hu, 1, 4 input would return the "http://wiki.math.bme.huatma"http://wiki.math.bme.hu string.

Don't forget to **delete** this string in the **main** function!

Smaller than average

Write a function that takes a float array (and its size). Calculates the average of its elements and returns a new (dynamically allocated) array that contains only those numbers that are smaller than the average.

Don't we need to return something else too? (Return it through a pointer parameter.)

Linked list

The linked list from the lecture (try to compile it):

```
#include<iostream>

using namespace std;

struct list_e {
  int num;
  struct list_e *next;
};

void append(struct list_e **start, int n) {
  struct list_e *e = new struct list_e;
  e->num = n;
  e->next = NULL;
  if (*start == NULL) {
    *start = e;
  } else {
    struct list_e *p = NULL;
    for(p = *start; p->next != NULL; p = p->next){}
    p->next = e;
  }
}

int main(void) {
  struct list_e *start = NULL;
  append(&start, 1);
  append(&start, 5);
  append(&start, -2);
  append(&start, 15);
  for(struct list_e *e = start; e != NULL; e = e->next) {
    cout << e->num << endl;
  }
  return 0;
}
```

Linked list: index

Write a function that takes the pointer to the first element of a linked list and an **n** non-negative integer. The function should return the **n**th element's stored value (indexed from 0). No need to make sure that the index isn't out of bounds, we can assume that it exists.

Linked list: delete

Write a function that deletes a linked list. Hint: We can safely delete a given element if we already have the next element's pointer saved in a variable.

Linked list: insert

Write a function that can insert a new linked list element to a given index. For example given a list containing 1, 5, 2 and we insert 6 into the 0th index we would get the 6, 1, 5, 2 list. Doing the same but for the 2th index would result in the 1, 5, 6, 2 list.

Linked list: contains

Write a function that can determine whether a given value is contained in the list or not. (Now in C++ we have the **bool** type as well. These can take on the values **true** and **false**. But in reality they still store true as 1 and false as 0, nothing fancier.)

Two-way linked list.

Implement a two-way linked list where every element store the next and the previous element as well. The previous of the first element should be **NULL**.

Implement the **append** function!