

## Repetitionsövningar 2022-06-02

1. Du har en lysdiod ansluten till PIN 17 samt en tryckknapp ansluten till PIN 27, som skall implementeras i ett C-program med det virtuella filsystemet `sysfs`. För att kontrollera att kopplingarna är korrekta så skall du testa att tända lysdioden samt läsa av tryckknappens insignal, både vid nedtryckning samt i normalläge. Ange Linuxkommandon för att genomföra följande:
  - a) Reservera PIN 17 samt 27 för aktuell process. Sätt PIN 17 till utport samt PIN 27 till inport.
  - b) Tänd sedan lysdioden.
  - c) Läs därefter av tryckknappens insignal. Vad bör du kunna avläsa vid nedtryckt respektive icke nedtryckt tryckknapp.
  - d) Släck lysdioden.
  - e) Ta bort reservation av PIN 17 samt 27.
2. I följande program skall en dynamisk vektor fyllas med 15 osignerade tal 100 – 114. Dessa tal skall sorteras i fallande ordning, alltså från 114 ned till 100, och sedan skrivs till en fil döpt `vector.txt`. Innehållet från denna fil skall sedan läsas in rad för rad och skrivs ut i konsolen. Programmet är dock inte slutfört.
  - a) Slutför samtliga vektorfunktioner i filen `vector.c` samt funktionen `file_read` i filen `main.c`.  
Skriv kompletterande programkod direkt i funktionerna på efterföljande sidor!
  - b) Efter att ha slutfört programmet skall detta testas i Ubuntu. Ange Linuxkommandon för att genomföra följande:
    - Skapa en ny katalog döpt `vector_example` och dirigera till denna.
    - Öppna filerna `header.h`, `main.c` samt `vector.c` i katalogen `vector_example` med radnummer i nano-editorn.
    - Kompilera dessa filer med GCC-kompilatorn och skapa en körbar fil döpt `main`.
    - Köra programmet från terminalen.

### Filen `header.h`:

```
#ifndef HEADER_H_
#define HEADER_H_

/* Inkluderingsdirektiv: */
#include <stdio.h>
#include <stdlib.h>

/* Enumerationer: */
enum sort_direction { SORT_DIRECTION_ASCENDING, SORT_DIRECTION_DESCENDING };

/* *****
 * vector: Dynamisk vektor för lagring av osignerade heltal.
 * ***** */
struct vector
{
    size_t* data; /* Pekare till dynamiskt fält innehållande osignerade heltal. */
    size_t size; /* Antalet element i fältet. */
};

/* Funktionsdeklarationer: */
void vector_new(struct vector* self, const size_t size);
void vector_assign(struct vector* self, const size_t start_val);
void vector_sort(struct vector* self, const enum sort_direction sort_direction);
void vector_print(const struct vector* self, FILE* stream);

#endif /* HEADER_H_ */
```

### Filen *main.c*:

```

/*****
 * Funktionen file_read är icke fullständig! Fyll i resten av funktionen!
 *****/

/* Inkluderingsdirektiv: */
#include "header.h"

/*****
 * file_read: Läser varje rad från en fil en efter en och skriver ut i konsolen.
 *****/
static void file_read(const char* filepath)
{
    FILE* fstream = fopen(filepath, "r");

    if (!fstream)
    {
        fprintf(stderr, "Could not open file at path %s!\n\n", filepath);
    }
    else
    {
        char s[100];

    }

    return;
}

/*****
 * main: Fyller en dynamisk vektor / array med 15 osignerade heltal 100 - 114.
 *       Innehållet sorteras i fallande ordning och skrivs till filen vector.txt.
 *       Innehållet från filen läses sedan in rad för rad och skrivs ut i konsolen.
 *****/
int main(void)
{
    struct vector v;
    vector_new(&v, 15);
    vector_assign(&v, 100);
    vector_sort(&v, SORT_DIRECTION_DESCENDING);

    FILE* fstream = fopen("vector.txt", "w");
    vector_print(&v, fstream);
    fclose(fstream);

    file_read("vector.txt");
    return 0;
}
```

### Filen *vector.c*:

```
/* *****
 * Slutför samtliga vektorfunktioner!
 * ***** */

/* Inkluderingsdirektiv: */
#include "header.h"

/* *****
 * vector_new: Initieringsrutin för objekt av strukten vector.
 * ***** */
void vector_new(struct vector* self, const size_t size)
{
    self->data = (size_t*)malloc(sizeof(size_t) * size);

    return;
}

/* *****
 * vector_assign: Fyller dynamisk vektor med osignerade heltal i stigande ordning.
 * ***** */
void vector_assign(struct vector* self, const size_t start_val)
{
    size_t num = start_val;

    return;
}

/* *****
 * vector_sort: Sorterar innehåll lagrat i dynamisk vektor i valbar ordning.
 * ***** */
void vector_sort(struct vector* self, const enum sort_direction sort_direction)
{
    for (register double* i = self->data; i < self->data + self->size - 1; ++i)
    {

    }
    return;
}

/* *****
 * vector_print: Skriver ut innehåll lagrat i en vektor till en valbar utenhet.
 * ***** */
void vector_print(const struct vector* self, FILE* stream)
{
    if (!stream) stream = stdout;
    fprintf(stream, "-----\n");

    fprintf(stream, "-----\n\n");
    return;
}
```