# MMS Star Camp 17

## КОНТРОЛНО №3

Възползвайте се от флаговете на **gcc**, които разгледахме по време на занятията:

**gcc -Wall -Wextra -Werror -fsanitize=address -static-libasan -g prog.c**

## ЗАДАЧА 1: - 10 т.

Създайте проста реализация на командата **xxd**.

Програмата приема 2 аргумента от командния ред:

* **filename**
* **print\_mode**

Ако **print\_mode** е -**H**, тогава се печата **HEX DUMP** на файла с име **filename.** Това означава, че се печата стойността на всеки байт от файла в шестнадесетична бройна система. Тази функционалност съответства на работата на командата **xxd filename.**

Ако **print\_mode** е -**B**, тогава се печата стойността на всеки байт от файла в двоична бройна система. Тази функционалност съответства на работата на командата **xxd -b filename.**

Компилирайте програмата в **executable файл** с името **my\_xxd**.

**Примерни извиквания**

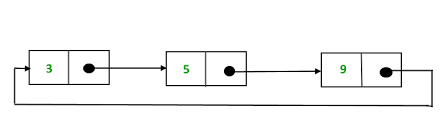
| **Команда** | **Символи, изпечатани на стандартния изход** |
| --- | --- |
| **cat test\_file** | **HELLO WORLD!** |
| **xxd test\_file** | **00000000: 4845 4c4c 4f20 574f 524c 4421 0a HELLO WORLD!.** |
| **./my\_xxd test\_file -H** | **48 45 4c 4c 4f 20 57 4f 52 4c 44 21 0a** |
| **xxd -b test\_file** | **00000000: 01001000 01000101 01001100 01001100 01001111 00100000 HELLO**  **00000006: 01010111 01001111 01010010 01001100 01000100 00100001 WORLD!**  **0000000c: 00001010 .** |
| **./my\_xxd test\_file -B** | **01001000 01000101 01001100 01001100 01001111 00100000 01010111 01001111 01010010 01001100 01000100 00100001 00001010** |

## 

## 

## ЗАДАЧА 2: - 10 т.

Цикличен свързан списък **(cyclic linked list)** е свързан списък, в който последният елемент пази указател (**pointer**) към първия елемент.



Направете структурата **Node**, за която да се съхраняват следните данни::

* **int data;**
* **struct Node \*next;** (Указател към следващия елемент на цикличния свързан списък)

1. **(3т.)** От стандартния вход се въвеждат числото **N>0** и след това **N цели числа, разделени със SPACE**. Създайте цикличен свързан списък в **динамичната памет**, чиито елементи са въведените числа. Погрижете се за освобождаването на динамичната памет.
2. **(3т.)** Напишете функцията **void print(struct Node \*cll)**, която приема указател към елемент на цикличен свързан списък (**cll**) като параметър. Функцията печата всичките елементи на **cll** на стандартния изход, точно по веднъж. Тествайте функцията, като я извикате в **main**.
3. **(4т.)** Напишете функцията:

**struct Node\* insertAfter(struct Node \*cll, int skipCount, int newElem)**

Функцията приема следните параметри:

* указател към първия елемент на цикличен свързан списък **(cll)**
* брой позиции, които ще бъдат пропуснати **(0 <= skipCount <= 1000000)**
* нов елемент, който да бъде добавен **(newElem)**.

Функцията започва от **cll**, пропуска следващите **skipCount** на брой елемента и след това добавя новата стойност (**newElem**). Функцията връща указател към новодобавения елемент.

Тествайте функцията, като я извиквате в **main**, с различни тестови аргументи.

## Задача 3 - 10т:

1. **(3т)** Създайте следната структура:

**struct Product**

**{**

**double old\_price;**

**double new\_price;**

**unsigned int id;**

**char product\_type;**

**char product\_name[??];**

**};**

Изберете подходящ размер за **product\_name**, така че да не се добавят празни байтове при запазването на структурата в паметта.

Създайте масив от 40 продукта със следните случайни стойности:

* **old\_price** и **new\_price** са десетични дроби между **4.99** и **100.45** с точност до 2 цифри след десетичната запетая.
* **product\_type** е символ измежду **‘A’, ‘B’, ‘C’ или ‘D’.**
* **product\_name** се избира от масив с предварително избрани символни низове.
* **id** е уникално положително число за всеки продукт.

Изпечатайте описателна информация за всеки продукт.

1. **(4т)** Сортирайте масива от продукти по следния начин:

* Гледа се колко се е покачила цената на един продукт. По-напред ще бъдат сортирани продуктите, които са имали **по-голямо покачване в цената**. Покачването на цената се смята като **(new\_price - old\_price)**.
* Ако покачването в цената на 2 продукта е еднаква (ако разликата е по-малка от 0.01), то тогава се сравнява **product\_type**. По-напред ще бъдат сортирани продуктите с **лексикографски по-голям product\_type**. Например продуктът с **product\_type=’B’** ще бъде сортиран по-напред от този с **product\_type=’A’**.
* Ако **product\_type** е еднакъв за два продукта, тогава се сравняват имената на продуктите. По-напред ще бъде сортиран продукт с **лексикографски по-малък product\_name**.
* Ако **product\_name** е еднакъв за два продукта - сравняват се техните **id-**та. По-напред ще бъде сортират продукт с **по-малко id**.

Изпечатайте масива от продукти след сортирането.

1. **(3т)** Напишете функция, която записва съдържанието на масива от **Product** в бинарен файл.

Създайте отделна програма, която да прочита съдържанието на такъв бинарен файл и да изпечатва описателен текст за всеки продукт на стандартния изход.