

Завдання:

- 1. Переробити програми, що були розроблені під час виконання лабораторних робіт з тем "Масиви" та "Цикли" таким чином, щоб використовувалися функції для обчислення результату.
- 2. Функції повинні задовольняти основну їх причетність - уникати дублювання коду.
Тому, для демонстрації роботи, ваша програма (функція `main()`) повинна мати можливість викликати розроблену функцію з різними вхідними даними.
- 3. Слід звернути увагу: параметри одного з викликів функції повинні бути згенеровані за допомогою генератора псевдовипадкових чисел `random()`.
- 4. Слід звернути увагу (#2): продемонструвати встановлення вхідних даних через аргументи додатка (параметри командної строки).
Обробити випадок, коли дані не передались - у цьому випадку вони матимуть значення за умовчуванням, обраними розробником.

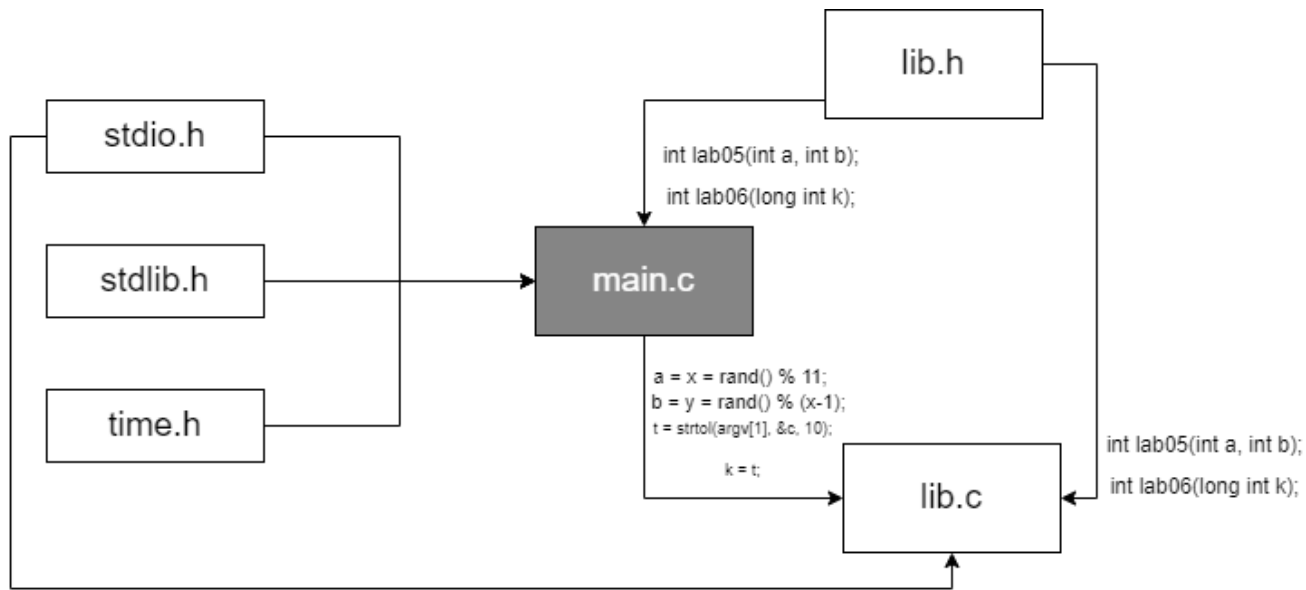
Опис програми

Функціональне призначення

Програма виконує дві операції.

- * Розраховує кіл-ть щасливих квитків.
 - При запуску програми вводиться номер квитка, від якого почнеться розрахунок.
 - Якщо не буде введено номер квитка, то розрахунок буде здійснено з білета під номером 1000 (можна вводити лише 4-цифрові числа. Якщо ввести більше, або менше цифр, то програма не працюватиме).
- * Знаходить найбільше просте число в діапазоні випадкових чисел.
 - Програма генерує випадкові числа та знаходить найбільше просте число.

Графічна структура програми:



Вміст файлу "main.c"

Головний файл

Це файл, який містить точку входу функцією `main`, виклики функцій `lab05`, `lab06` та значення для аргументів цих функцій.

```
main(int argc, char *argv[])
```

Головна функція.

Аргументи

`int argc, argv *c[]` Аргументи які зберігають значення введені через командний рядок

`argc` - зберігає к-ть значень

`argv` - зберігає значення

Послідовність дій

- * Присвоїти значення аргументам `argc` і `argv`.
 - `argc` - `int` аргумент, що необхідний для обчислення всіх щасливих квитків.
 - `argv` - `char` масив, що необхідний для збереження в собі стартового значення для перевірки всіх квитків.
- * Створити змінні, яким буде надано значення для аргументів функцій `link lab05`, `lab06`.
 - `t` - зберігає стартове значення без змін, від якого почнеться перевірка квитків для функції `lab06`. Його програма використовує, якщо користувач не ввів інших даних.

- x - зберігає випадкове значення, яке є мінімальним в діапазоні для функції lab05.

- y - зберігає випадкове значення, яке є максимальним числом в діапазоні для функції lab05.

- c - змінна яка використовується для перевірки аргументів командного рядка.

- * Згенерувати випадкові числа за допомогою генератора rand() та функції srand(), після чого привласнити їх змінним x і y.
- * Викликати функцію lab05 і присвоїти її аргументам значення примінних x та y.

```
srand((unsigned int)time(NULL));  
x=rand()%11;  
y=rand()%(x-1);  
lab05(x, y);
```

- * Перевірити, чи були введені аргументи через командний рядок.
- * Якщо перевірка була пройдена, то переобразувати аргумент командного рядка в значення типу int і присвоїти його змінній t
- * Перевірити, чи не менше t мінімального значення квитка
- * Якщо перевірка була пройдена, то присвоїти t аргументам функції lab06
- * Якщо одна з перевірок не була пройдена, то присвоїти аргументам функції lab06 значення за замовчуванням.

```
if(t>1000 && t<9999)
```

```
{
```

```
lab06(t);
```

```
}else
```

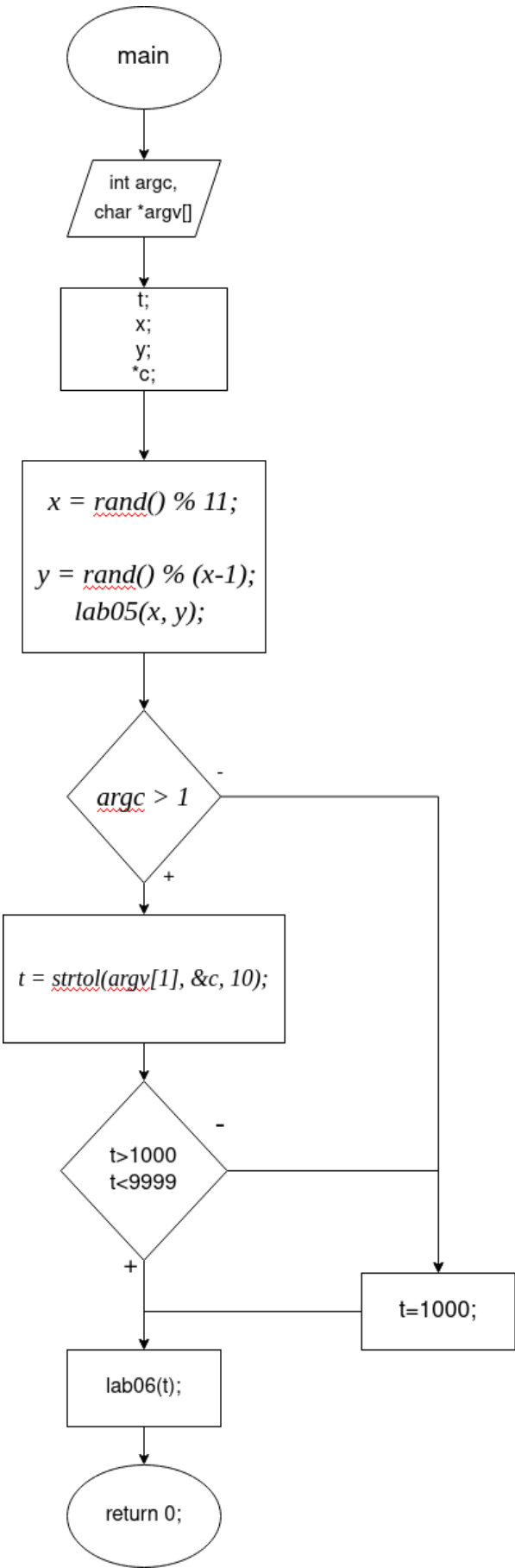
```
{
```

```
t=1000;
```

```
GitHub
```

```
}
```

function main:



Вміст файлу "lib.c"

Бібліотечний файл

Цей файл містить реалізацію функцій lab05, lab06.

```
int lab05(a, b)
```

Ця функція знаходить найбільше просте число у випадковому діапазоні.

Аргументи

a, b - діапазон чисел у якому відбувається пошук найбільшого числа

a - найбільше число діапазону

b - найменше число діапазону

Послідовність дій

- * Створення змінних i, max.
 - i - використовується для перевірки просте число чи ні.
 - max - зберігає найбільше просте число.
- * Запуск циклу для перевірки чисел у встановленому діапазоні.

```
while(a>b)
```

- * Запуск циклу для того, щоб дізнатися число просте чи ні.

```
while(b%i != 0){
```

```
    i++;
```

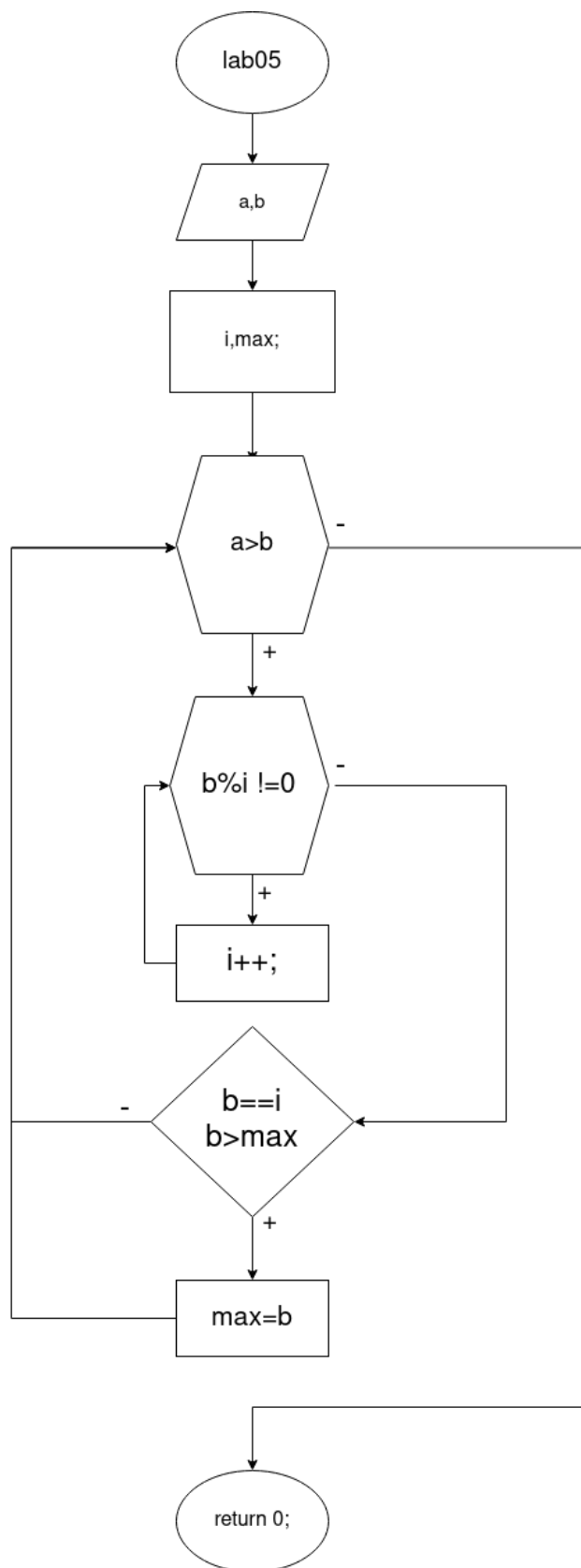
```
}
```

- * Перевірити, щоб дізнатися число виявилось простим чи ні.
Якщо воно просте, то перевірити чи більше воно числа яке зараз зберігається в змінній max, якщо перевірка була пройдена то присвоїти змінної max це число.

```
if(b==i && b>max){
```

```
    max=b;
```

```
}
```



`int lab06(k)`

Ця функція знаходить усі щасливі квитки в діапазоні від k до 9999.

Аргументи

k - число від якого починається пошук квитків

Послідовність дій

- * Створення змінних A[9999],sum1,sum2,n1,n2,n3,n4,n,i.
 - A - масив в який записуватимемо щасливі квитки.
 - Задати змінні sum1 і sum2 в яких буде зберігатися суми першої і другої пари чисел.
 - Змінні n1, n2, n3, n4 зберігатимуть по одному числу з номера квитка.
 - Змінна n – це кількість щасливих квитків.

- * Створення циклу який перевіряє всі квитки від k до 9999.

```
while( k<=9999)
```

- * Розбити номер квитка на 4 числа.

```
n1=k/1000;
```

```
n2=(k/100)%10;
```

```
n3=(k/10)%10;
```

```
n4=k%10;
```

- * Дізнатись суму першої і другої пари чисел .

```
sum1=n1+n2;
```

```
sum2=n3+n4;
```

- * Якщо сума першої пари дорівнює сумі другої, то записати цей квиток у масив і додати 1 до n (до кількості щасливих квитків).

```
if(sum1==sum2){
```

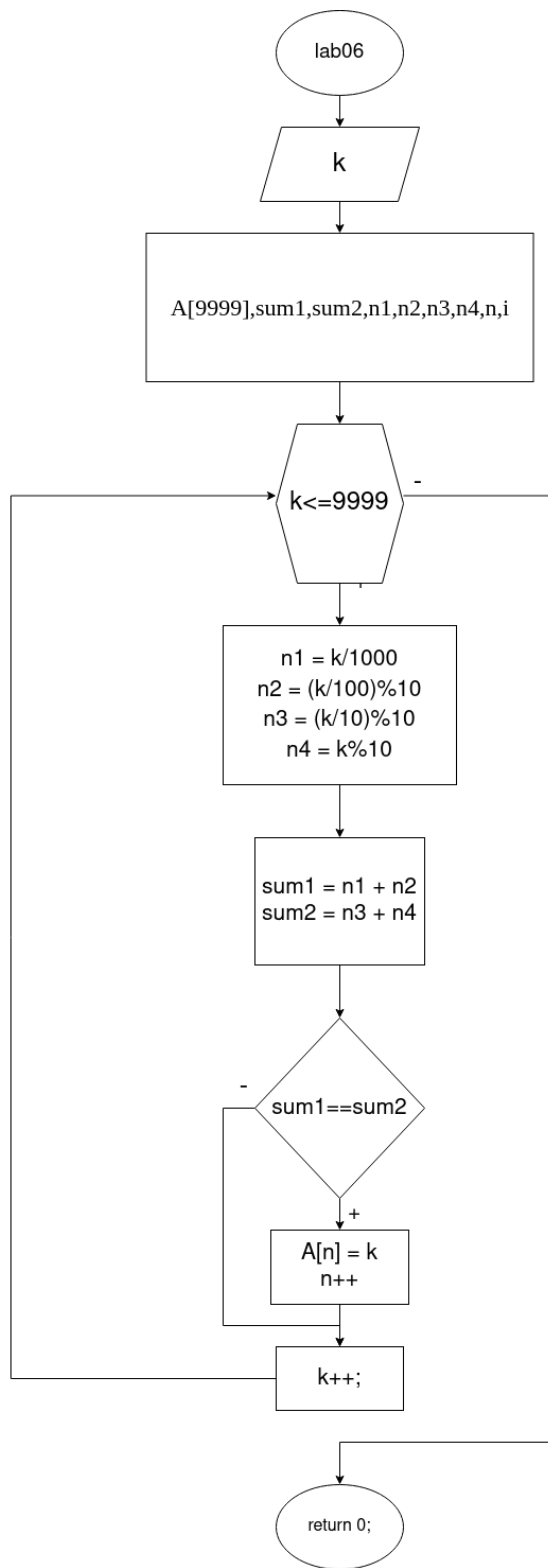
```
A[n]=k;
```

```
n++;
```

```
}
```

- * І збільшити номер квитка на +1.

```
k++;
```



Вміст файлу "lib.c"

Бібліотечний файл

Цей файл містить декларацію функцій lab05, lab06.

```
int lab05(int a, int b);
```

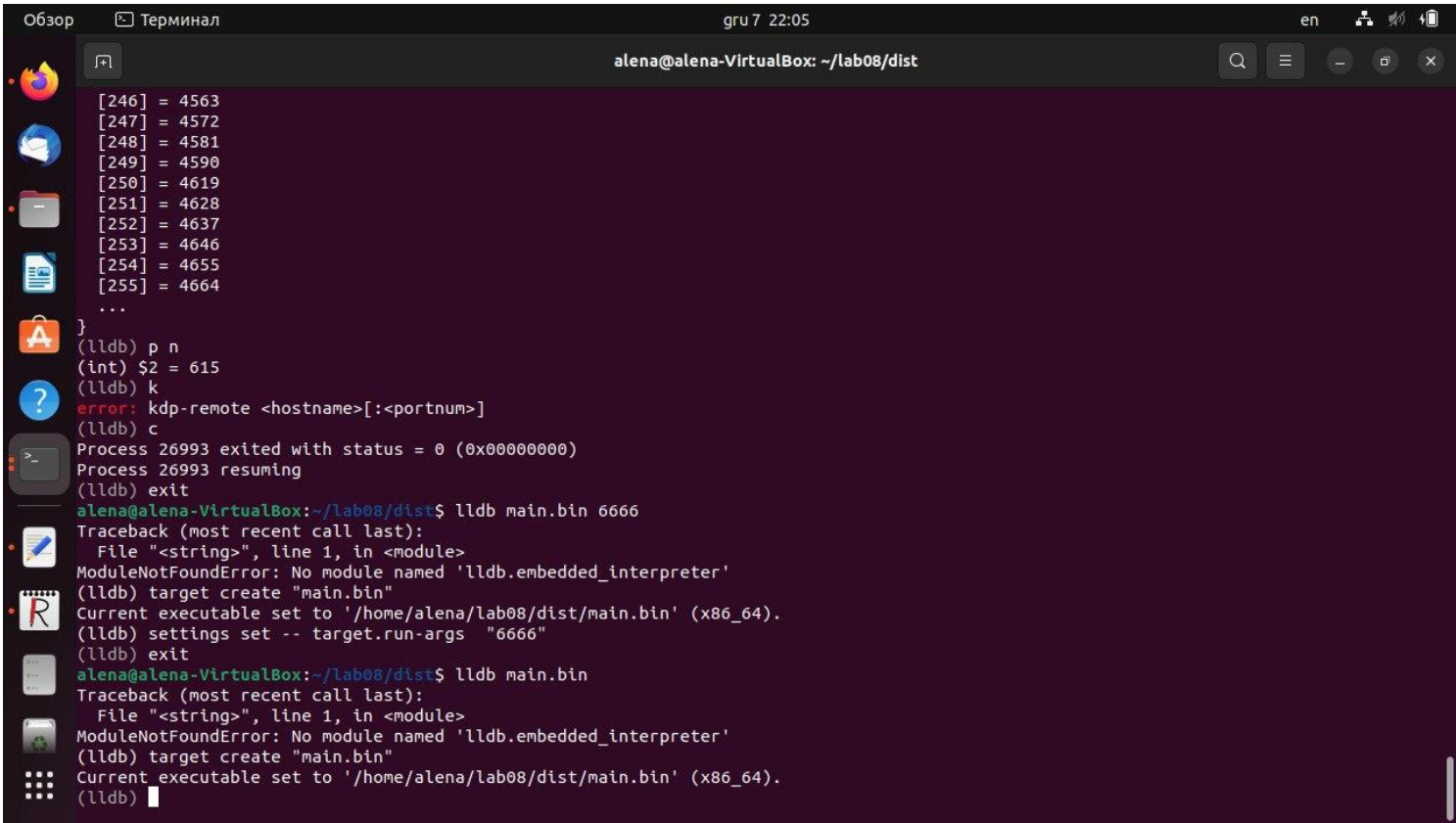
```
int lab06(long int k);<br>
```


Структура проекту лабораторної роботи:

- └─ lab07
- └─ Makefile
- └─ README.md
- └─ src
- └─ lib.c
- └─ lib.h
- └─ main.c

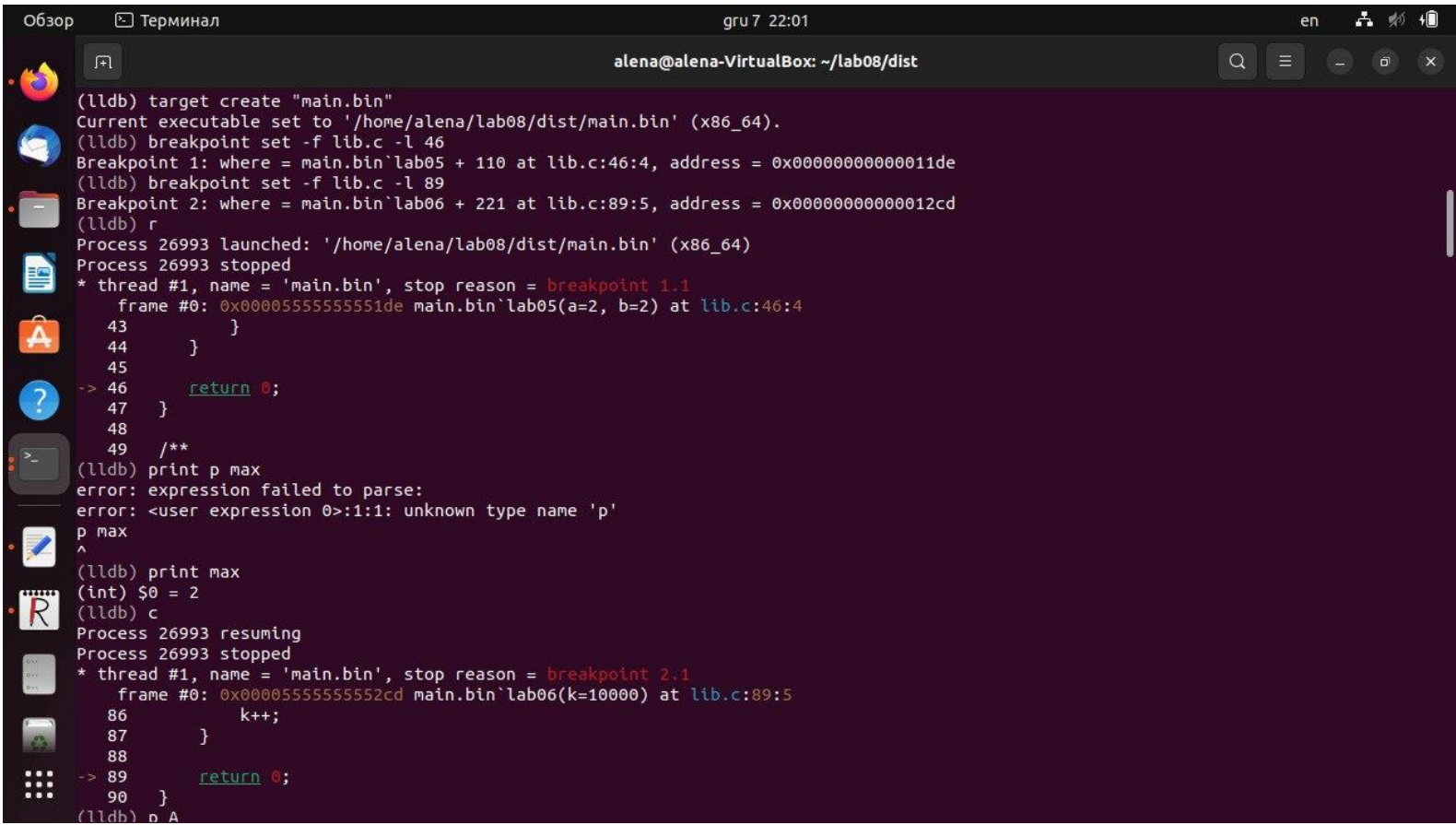
Варіанти використання

- * Ви можете використовувати цю програму двома методами:
Перший спосіб - це при запуску двійкового файлу, це вказати номер квитка, з якого буде починатися розрахунок. Як згадувалося раніше, потрібно ввести лише 4-цифрове число. Якщо введено інше число, то програма почне перевірку білетів з числа 1000. Також слід зауважити що програма може почати пошук щасливих квитків тільки від одного мінімального значення, тому якщо Вам треба дізнатися кількість квитків більше, ніж в одному діапазоні, то необхідно буде запустити програму ще раз.
- * Другий метод використання цієї програми - це при запуску двійкового файлу не вводити значення командного рядка. Тоді програма видасть список щасливих квитків від 1000 до 9999.



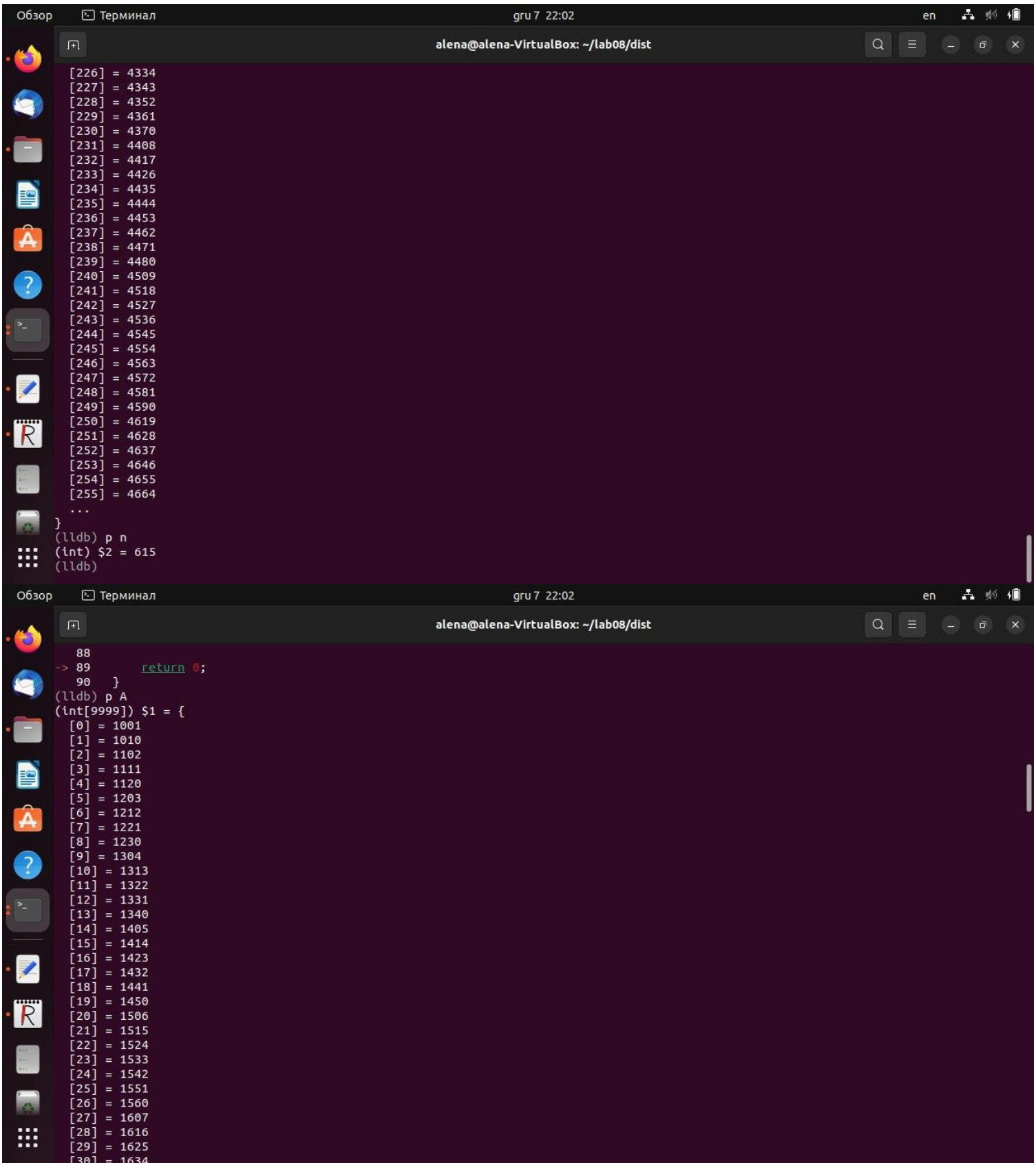
- * Щоб побачити результати роботи програми, вам потрібно завантажити її в LLDB. На початку вкажіть номер квитка. Якщо Вам потрібно дізнатися найбільше просте число, то для цього Вам знадобиться встановити точку зупинки на рядку 21 у файлі lib.c та вивести значення змінної max. Якщо Вам потрібно дізнатися кількість щасливих квитків або переглянути їх, то для цього Вам потрібно

зробити точку зупинки на рядку 48 у файлі lib.c. Щоб дізнатися кількість квитків, Вам потрібно вивести змінну n. А щоб переглянути щасливі квитки, вам потрібно вивести масив A[].



The screenshot shows a terminal window titled "alena@alena-VirtualBox: ~/lab08/dist". The terminal displays the following LLDB commands and output:

```
(lldb) target create "main.bin"
Current executable set to '/home/alena/lab08/dist/main.bin' (x86_64).
(lldb) breakpoint set -f lib.c -l 46
Breakpoint 1: where = main.bin`lab05 + 110 at lib.c:46:4, address = 0x00000000000011de
(lldb) breakpoint set -f lib.c -l 89
Breakpoint 2: where = main.bin`lab06 + 221 at lib.c:89:5, address = 0x00000000000012cd
(lldb) r
Process 26993 launched: '/home/alena/lab08/dist/main.bin' (x86_64)
Process 26993 stopped
* thread #1, name = 'main.bin', stop reason = breakpoint 1.1
  frame #0: 0x0000555555551de main.bin`lab05(a=2, b=2) at lib.c:46:4
    43         }
    44     }
    45
-> 46     return 0;
    47 }
    48
    49 /**
(lldb) print p max
error: expression failed to parse:
error: <user expression 0>:1:1: unknown type name 'p'
p max
^
(lldb) print max
(int) $0 = 2
(lldb) c
Process 26993 resuming
Process 26993 stopped
* thread #1, name = 'main.bin', stop reason = breakpoint 2.1
  frame #0: 0x0000555555552cd main.bin`lab06(k=10000) at lib.c:89:5
    86         k++;
    87     }
    88
-> 89     return 0;
    90 }
(lldb) p A
```



The image shows a terminal window within a virtual machine. The window title is "alena@alena-VirtualBox: ~/lab08/dist". The terminal displays the output of a program, which consists of a list of values in square brackets, each followed by an equals sign and a number. The values range from 4334 to 4664. Below this list, there is a closing brace "}" and a line "(lldb) p n". Then, the command "(int) \$2 = 615" is entered, followed by "(lldb)".

```
[226] = 4334
[227] = 4343
[228] = 4352
[229] = 4361
[230] = 4370
[231] = 4408
[232] = 4417
[233] = 4426
[234] = 4435
[235] = 4444
[236] = 4453
[237] = 4462
[238] = 4471
[239] = 4480
[240] = 4509
[241] = 4518
[242] = 4527
[243] = 4536
[244] = 4545
[245] = 4554
[246] = 4563
[247] = 4572
[248] = 4581
[249] = 4590
[250] = 4619
[251] = 4628
[252] = 4637
[253] = 4646
[254] = 4655
[255] = 4664
...
}
(lldb) p n
(int) $2 = 615
(lldb)
```

Below the first terminal window, there is another terminal window with the same title. It shows the execution of a C program. The code includes a loop that prints values from 1001 to 1634. The output of the program is displayed in the terminal, showing a list of values in square brackets, each followed by an equals sign and a number. The values range from 1001 to 1634.

```
88
-> 89     return 0;
90 }
(lldb) p A
(int[9999]) $1 = {
[0] = 1001
[1] = 1010
[2] = 1102
[3] = 1111
[4] = 1120
[5] = 1203
[6] = 1212
[7] = 1221
[8] = 1230
[9] = 1304
[10] = 1313
[11] = 1322
[12] = 1331
[13] = 1340
[14] = 1405
[15] = 1414
[16] = 1423
[17] = 1432
[18] = 1441
[19] = 1450
[20] = 1506
[21] = 1515
[22] = 1524
[23] = 1533
[24] = 1542
[25] = 1551
[26] = 1560
[27] = 1607
[28] = 1616
[29] = 1625
[30] = 1634
```

Висновок:

Навчилася користуватися функціями та правильно документувати код. Під час тестування програми були отримані результати функції lab05 - це отримання найбільшого числа в діапазоні, і робота функції labb06 - це отримання кількості щасливих квитків та їх перегляду, після введення аргументу командного рядка.