Звіт з лабораторної роботи
за дисципліною "алгоритми і структури даних"
студента групи ПА-17-1
Панасенка Єгора Сергійовича
Кафедра комп’ютерних технологій, фпм, дну
2017-2018 навч.р.

1. Постановка задачі:

   Розробити програму, в якій

   1. реалізувати абстрактний тип даних, необхідний для виконання індивідуального завдання згідно з варіантом (стек або чергу);
   2. реалізувати основні операції для розробленої структури;
   3. реалізувати індивідуальне завдання.

   Обов’язкові вимоги:

   1. після завантаження програма пропонує користувачу меню для вибору дії з такими пунктами:
      1. Додати елемент до структури даних.
      2. Видалити елемент із структури даних.
      3. Вивести елементи структури на екран.
      4. Очистити структуру.
      5. Вивести кількість елементів структури, індекс першого та останнього елементу.
      6. Пункти щодо індивідуальних завдань згідно з варіантом ;
      7. Завершення роботи.
   2. Організувати роботу програми таким чином, щоб перелічені вище дії можна було б виконувати неодноразово за один сеанс роботи програми.
   3. При видаленні елементів передбачити обробку помилкових ситуацій, коли будуть виконуватися спроби видалення елемента із порожньої структури.
   4. Структуру даних реалізувати на базі масиву із n елементів (розмір масиву n вводити з клавіатури) або на базі однозв’язного списку – на вибір студента, якщо спосіб реалізації структури не зазначений в індивідуальному завданні.

5. Операції над структурою даних реалізувати у вигляді окремих функцій.
6. Індивідуальні завдання за варіантом – окремі функції.
7. Для виконання операцій над елементами структури використовувати розроблені операції-функції, які відповідають порядку роботи даної структури (наприклад, неможна видаляти елемент з середини стеку або черги).

Варіант 7

Розробити програму для конвертації цілих десяткових чисел у двійкову систему числення. Для розв’язання задачі використати стек на базі статичного масиву.

2. Опис структури программи та реалізованих функцій:
   a) Програма задає тип stack з struct stack, а також задає struct stack з такими полями:
      1. a – де зберігається адреса масиву з даними
      2. size – де зберігається кількість елементів
      3. chunks – де зберігається фізичний розмір масиву у чанках (блоках, за замовчуванням по 128 байт)
   b) Программа має такі функції:
      1. void ign_other(FILE * input)
         1. Ігнорує непотрібні данні до кінця рядка. Наприклад якщо ми запросили одне число, а ввели число і якийсь текст то программа забере тільки число.
         2. Аргументи:
            1. input - вхідний поток.
         3. Функція нічого не виводить.
      2. int get_number (FILE * input, FILE * output, char message[], int min, int max)
         1. Циклічно забирає число у проміжку з min до max з вхідного потоку доки не отримає потрібне число
         2. Аргументи
            1. input – вхідний потік даних
            2. output – вихідний потік даних
            3. message – повідомлення у якому повинен буди запит на число
            4. min – мінімальне доступне значення

        5. max – максимально доступне значення
    3. Функція виводить число від min до max
3. stack * stack_init ()
    1. Створює стек
    2. Аргументів немає
    3. Функція виводить адресу нового стеку
4. void stack_push (stack * st, int x)
    1. Додає один елемент у стек
    2. Аргументи
        1. st – адреса стеку
        2. x – значення яке буде у нового елемента
    3. Функція нічого не виводить
5. char stack_empty (stack * st)
    1. Перевіряє чи є стек пустим
    2. Аргументи
        1. st – адреса стеку
    3. Функція 1 якщо стек пустий та 0 якщо не пустий
6. int stack_size (stack * st)
    1. Виводить розмір стеку
    2. Аргументи
        1. st – адреса стеку
7. int stack_pop (stack * st)
    1. Виводить та видаляє останній елемент зі стеку
    2. Аргументи
        1. st – адреса стеку
    3. Виводить останній елемент зі стеку
8. int stack_top (stack * st)
    1. Виводить останній елемент зі стеку
    2. Аргументи
        1. st – адреса стеку
9. void stack_print (FILE * output, stack * st)
    1. Показує усі елементи списку в output

2. Аргументи
    1. output – вихідний потік даних
    2. st – адреса стеку
3. Функція нічого не виводить

10. void stack_clean (stack * st)
    1. Очищує стек
    2. Аргументи
        1. st – адреса стеку
    3. Функція нічого не виводить
11. void stack_print_info (FILE * output, stack * st)
    1. Показує кількість елементів, перший та останній індекс
    2. Аргументи
        1. output – вихідний потік даних
        2. st – адреса стеку
    3. Функція нічого не виводить
12. char * convert_to_binary (stack * st, int x)
    1. Перетворює число у бінарний вигляд.
    2. Аргументи
        1. st – адреса стеку
        2. x – число яке буде перетворене
    3. Виводить строку бінарного вигляду
13. stack * stack_delete (stack * st)
    1. Видаляє стек
    2. Аргументи
        1. st – адреса стеку
    3. Виводить NULL

c) У головній функції виконуються такі дії:
1. Цикл доки не отримаємо запит на вихід програми
    1. Вихід меню
    2. Запит числа
    3. За пунктом меню виконує потрібну функцію
    4. Якщо не потрібно виходити з програми

1. Показує стек
2. Запитує ENTER для продовження
3. Запитує данні доки не отримаємо ENTER

2. Видаляємо стек
3. Виходимо з програми

3. Код:

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>

#define CHUNK 128

void ign_other(FILE * input) {
	char c = 0;
	while (c!='\n')
		c=fgetc(input);
}

int get_number (FILE * input, FILE * output, char message[], int min, int max) {
	int number = min-1;
	while (number < min || number > max) {
		fprintf(output,"%s:\n",message);
		fscanf(input,"%i",&number);
		ign_other(stdin);
		if (number < min || number > max) {
			system("clear");
			fprintf(output,"Вы ввели неправильне число, дозволено тільки ціле
число від %i до %i\n",min,max);
		}
	}
	return number;
}

typedef struct stack stack;
struct stack
{
	int * a;
	int size;
	int chunks;
};

stack * stack_init () {
	stack * st = (stack*) malloc(sizeof(stack));
	st->chunks = 1;
	st->size = 0;
	st->a = (int *) calloc(st->chunks*CHUNK,sizeof(int));
	return st;
}

void stack_push (stack * st, int x) {
	if (st->chunks*CHUNK == st->size) {
```

```
            st->chunks++;
            st->a = (int *) realloc (st->a, st->chunks * CHUNK * sizeof(int));
      }
      st->a[st->size++]=x;

}

char stack_empty (stack * st) {
      if (st->size == 0) return 1;
      else return 0;
}

int stack_size (stack * st) {
      return st->size;
}

int stack_pop (stack * st) {
      if (stack_empty(st)) return 0;
      int x = st->a[st->size-1];
      st->size--;
      return x;
}

int stack_top (stack * st) {
      if (stack_empty(st)) return 0;
      return st->a[st->size-1];
}

void stack_print (FILE * output, stack * st) {
      if (stack_empty(st)) {
            fprintf(output,"Stack is empty\n");
            return;
      }
      for (int i=0;i<stack_size(st)-1;i++)
            fprintf(output,"%i ",st->a[i]);
      fprintf(output,"%i\n",st->a[stack_size(st)-1]);
}

void stack_clean (stack * st) {
      st->chunks = 1;
      st->size = 0;
      st->a = (int *) realloc (st->a, st->chunks * CHUNK * sizeof(int));
}

void stack_print_info (FILE * output, stack * st) {
      if (stack_empty(st)) fprintf(output,"Stack is empty\n");
      fprintf(output,"Size - %i, first index - %i, last index - %i\n",st-
>size,0,st->size-1);
}

char * convert_to_binary (stack * st, int x) {
      stack_clean(st);
      int tmp = x;
      while (tmp > 1) {
            stack_push(st,tmp%2);
            tmp/=2;
      }
      if (tmp%2)
```

```
	stack_push(st,1);
	char * str = (char*) calloc(stack_size(st)+1,sizeof(char));
	for (int i=0;stack_size(st) > 0;i++) {
		str[i]=stack_pop(st)+'0';
	}
	return str;
}

stack * stack_delete (stack * st) {
	free(st->a);
	free(st);
	return NULL;
}

int main () {
	stack * st = stack_init();
	int answer = 0;
	while (answer != 7) {
		printf("Choose action:\n");
		printf("1. Push to stack\n");
		printf("2. Pop from stack\n");
		printf("3. Print stack\n");
		printf("4. Clean stack\n");
		printf("5. Print info about stack\n");
		printf("6. Convert number to binary\n");
		printf("7. Exit\n");
		scanf("%i",&answer);
		ign_other(stdin);
		system("clear");
		switch (answer) {
			case 1:
				stack_push(st, get_number(stdin,stdout,"Enter
number",INT_MIN+1,INT_MAX-1));
				break;
			case 2:
				printf("Last element: %i\n",stack_pop(st));
				break;
			case 3:
				//~ stack_print(stdout,st);
				break;
			case 4:
				stack_clean(st);
				break;
			case 5:
				stack_print_info(stdout,st);
				break;
			case 6:

	printf("%s\n",convert_to_binary(st,get_number(stdin,stdout,"Enter
number",INT_MIN+1,INT_MAX-1)));
				break;
			case 7:
				break;
			default:
				printf("Wrong input, you need to type number from 1 to
8\n");
		}
		if (answer != 7) {
```

```
                stack_print(stdout,st);
                printf("Click ENTER to exit.\n");
                ign_other(stdin);
                system("clear");
            }
        }
        stack_delete(st);
        return 0;
}
```

4. Результати роботи програми:

Приклад 1:

```
Choose action:
1. Push to stack
2. Pop from stack
3. Print stack
4. Clean stack
5. Print info about stack
6. Convert number to binary
7. Exit
1
Enter number:
10
10
Click ENTER to exit.

Choose action:
1. Push to stack
2. Pop from stack
3. Print stack
4. Clean stack
5. Print info about stack
6. Convert number to binary
7. Exit
1
Enter number:
123
10 123
Click ENTER to exit.

Choose action:
1. Push to stack
2. Pop from stack
3. Print stack
4. Clean stack
5. Print info about stack
6. Convert number to binary
7. Exit
1
Enter number:
12312
10 123 12312
Click ENTER to exit.

Choose action:
1. Push to stack
```

```
2. Pop from stack
3. Print stack
4. Clean stack
5. Print info about stack
6. Convert number to binary
7. Exit
1
Enter number:
1231341
10 123 12312 1231341
Click ENTER to exit.

Choose action:
1. Push to stack
2. Pop from stack
3. Print stack
4. Clean stack
5. Print info about stack
6. Convert number to binary
7. Exit
5
Size - 4, first index - 0, last index - 3
10 123 12312 1231341
Click ENTER to exit.

Choose action:
1. Push to stack
2. Pop from stack
3. Print stack
4. Clean stack
5. Print info about stack
6. Convert number to binary
7. Exit
3
10 123 12312 1231341
Click ENTER to exit.

Choose action:
1. Push to stack
2. Pop from stack
3. Print stack
4. Clean stack
5. Print info about stack
6. Convert number to binary
7. Exit
2
Last element: 1231341
10 123 12312
Click ENTER to exit.

Choose action:
1. Push to stack
2. Pop from stack
3. Print stack
4. Clean stack
5. Print info about stack
6. Convert number to binary
7. Exit
```

```
3
10 123 12312
Click ENTER to exit.

Choose action:
1. Push to stack
2. Pop from stack
3. Print stack
4. Clean stack
5. Print info about stack
6. Convert number to binary
7. Exit
5
Size - 3, first index - 0, last index - 2
10 123 12312
Click ENTER to exit.
6
Choose action:
1. Push to stack
2. Pop from stack
3. Print stack
4. Clean stack
5. Print info about stack
6. Convert number to binary
7. Exit
6
Enter number:
50
110010
Stack is empty
Click ENTER to exit.

Choose action:
1. Push to stack
2. Pop from stack
3. Print stack
4. Clean stack
5. Print info about stack
6. Convert number to binary
7. Exit
1
Enter number:
123
123
Click ENTER to exit.

Choose action:
1. Push to stack
2. Pop from stack
3. Print stack
4. Clean stack
5. Print info about stack
6. Convert number to binary
7. Exit
4
Stack is empty
Click ENTER to exit.
```

```
Choose action:
1. Push to stack
2. Pop from stack
3. Print stack
4. Clean stack
5. Print info about stack
6. Convert number to binary
7. Exit
1
Enter number:
1231
1231
Click ENTER to exit.

Choose action:
1. Push to stack
2. Pop from stack
3. Print stack
4. Clean stack
5. Print info about stack
6. Convert number to binary
7. Exit
1
Enter number:
1234
1231 1234
Click ENTER to exit.

Choose action:
1. Push to stack
2. Pop from stack
3. Print stack
4. Clean stack
5. Print info about stack
6. Convert number to binary
7. Exit
7
```