Самостоятельная работа.

Тема 11. Процедуры.

1. Напишите программу, в которой реализуется замкнутая цепочка из экземпляров структуры: каждый экземпляр ссылается на следующий, а последний экземпляр ссылается на первый.

Имя	Значение	Тип
value	Поле структуры хранимое значение	int
next	Указатель на следующий экземпляр структуры	List
i	Счетчик цикла	List
size	Размер списка	int
begin	Указатель на начало списка	list
ptrOnStruct	Указатель на текущий экземпляр структуры	list
ptrOnStructNext	Указатель на следующий экземпляр структуры	list
fitst	Ссылка на первый элемент списка	list
start	Ссылка на первый элемент списка	list

Код программы:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct List {
	int value;
	struct List *next;
}list;
list* create_list(int size) {
	list* begin = malloc(sizeof(list));
	begin->value = 0;
	list *ptrOnStruct, *ptrOnStructNext;
	ptrOnStruct = begin;
	for (int i = 1; i < size; i++) {
		ptrOnStructNext = malloc(sizeof(list));
		ptrOnStructNext->value = ptrOnStruct->value + 1;
		ptrOnStruct->next = ptrOnStructNext;
		ptrOnStruct = ptrOnStructNext;
	}
	ptrOnStruct->next = begin;
	return begin;
}

void show_list(list* begin) {
	list* ptrOnStruct = begin;
	do {
		printf("%d ", ptrOnStruct->value);
		ptrOnStruct = ptrOnStruct->next;
	}while (ptrOnStruct != begin);
}
void delete_list( list *begin) {
```

```
        list *ptrOnStructNext = begin->next;
        list *first = begin;
        while (ptrOnStructNext != first) {
                free(begin);
                ptrOnStructNext = ptrOnStructNext->next;
                begin = ptrOnStructNext;
        }
}
int main(){
        list* start = create_list(10);
        show_list(start);
        delete_list(start);

    return 0;
}
```

Результат выполненной работы:

```
C:\Users\grish\OneDrive\Рабочий стол\лабы по си\лаба 11>tasksam1
0 1 2 3 4 5 6 7 8 9
```

3. Напишите программу, в которой с помощью структур реализуется бинарное дерево: каждый экземпляр структуры элемента дерева (за исключением экземпляров последнего уровня - листьев дерева) содержит ссылку на два других экземпляра (поддерево или лист).

Список идентификаторов:

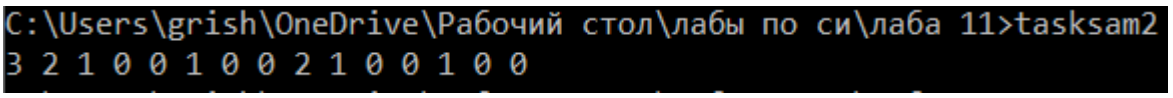
Имя	Значение	Тип
value	Поле структуры, хранимое значение	int
left	Поле структуры, указатель на левый узел или лист	Tree
right	Поле структуры, указатель на правый узел или лист	Tree
ptrOnTreeLeft	Указатель на правый отросток дерева	tree
ptrOnTreeRight	Указатель на левый отросток дерева	tree
root	Узел дерева	tree
level	Уровень узла в дереве	int

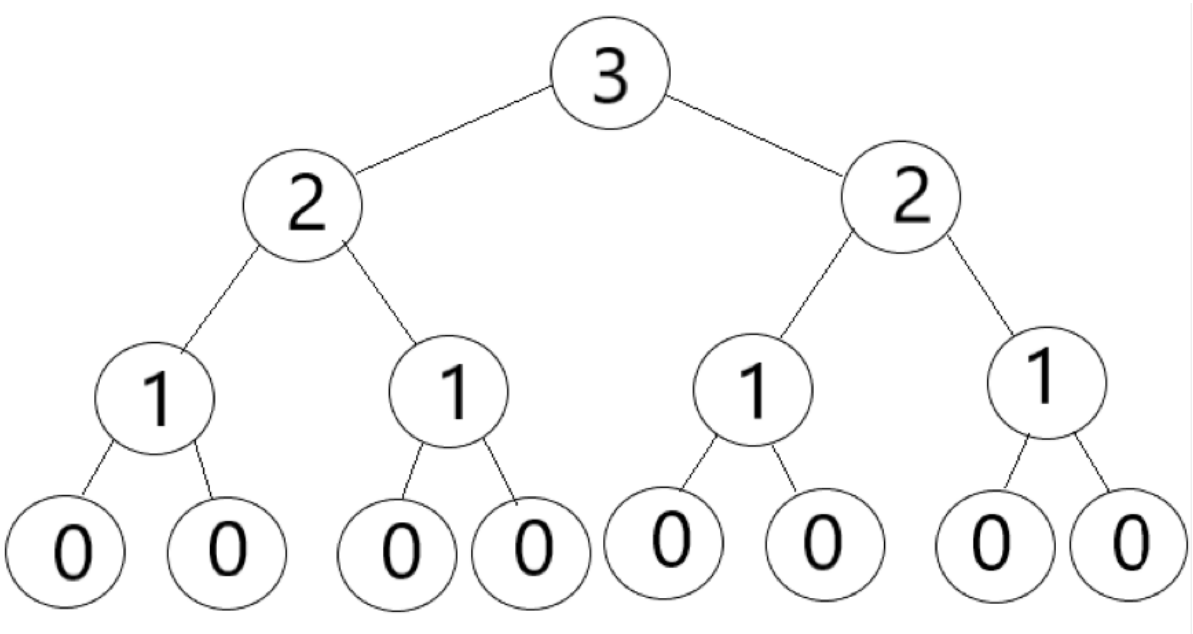
Код программы:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct Tree {
        int value;
        struct Tree* left;
        struct Tree* right;
}tree;
void create_tree(tree *root, int level) { // создание узла на n-ом уровне
        tree *ptrOnTreeLeft, *ptrOnTreeRight;
        root->value = level;
        if(level) {
```

```
                ptrOnTreeLeft = malloc(sizeof(tree));
                ptrOnTreeRight = malloc(sizeof(tree));
                root->left = ptrOnTreeLeft;
                root->right = ptrOnTreeRight;
                level--;
                create_tree(ptrOnTreeLeft, level);
                create_tree(ptrOnTreeRight, level);
        }
        else {
                root->left = NULL;
                root->right = NULL;
        }
}
void print_tree(tree* root) {
        printf("%d ", root->value);
        if (root->left != NULL)
                print_tree(root->left);
        if (root->right != NULL)
                print_tree(root->right);
}
void delete_tree(tree* root) {
        if (root->left != NULL)
                delete_tree(root->left);
        if (root->right != NULL)
                delete_tree(root->right);
        free(root);
}
int main() {
        tree* root = malloc(sizeof(tree));
        create_tree(root, 3);
        print_tree(root);
        delete_tree(root);
        return 0;
}
```

Результат выполненной работы:





5. Реализовать структуру (struct) с вложенным поименованным объединением (union) и дополнительным целочисленным полем внутри данной структуры, в котором по условному номеру хранится информация о том, какие именно данные записаны в основное поле типа union. Внутри данного вложенного объединения определить два поля типа char и int. Создать и заполнить динамический массив таких структур целыми и символьными данными, заполняя вспомогательное целое поле, для сохранения информации о хранимом в каждом объединении типе данных (char/int). Реализовать распечатку данных массива таких структур в консоль.

Список идентификаторов:

Имя	Значение	Тип
type	Номер типа данных	int
ch	Символ в объединении	char
i	Целое число в объединении	int
k	Счетчик цикла	int
unstrct	Структура с объединением	unionStruct
first	Указатель на первый элемент массива	unionStruct
amount	Количесво структур в массиве	int

Код программы:

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct myStruct {
```

```
	int type; //0 - char, 1 - int
	union data{
		char ch;
		int i;
	}twin;
}unionStruct;
unionStruct* set_struct(unionStruct * unstrct, int amount) {
	unionStruct* first = malloc(amount * sizeof(unionStruct));
	for (int k = 0; k < amount; k++) {
		printf("\ntype  = ");
		scanf("%d", &first[k].type);
		if (0 == first[k].type){
			printf("enter char: ");
			scanf("%s", &first[k].twin.ch);
		}
		else if (1 == first[k].type) {
			printf("enter num: ");
			scanf("%d", &first[k].twin.i);
		}
		else printf("error type");

	}
	return first;
}
void show_struct(unionStruct *unstrct, int amount) {
	printf("\n");
	for (int k = 0; k < amount; k++) {
		printf("\n%d = ", k + 1);
		if (0 == unstrct[k].type) {
			printf("%c", unstrct[k].twin.ch);
		}
		else if (1 == unstrct[k].type) {
			printf("%d", unstrct[k].twin.i);
		}
	}
}
void delete_struct(unionStruct* unstrct) {
	free(unstrct);
}
int main(){
	int amount;
	unionStruct* unstrct;
	scanf("%d", &amount);
	unstrct = set_struct(unstrct, amount);
	show_struct(unstrct, amount);
	delete_struct(unstrct);
    return 0;
}
```

Результат выполненной работы:

```
C:\Users\grish\OneDrive\Рабочий стол\лабы по си\лаба 11>tasksam5
3

type  = 0
enter char: g

type  = 0
enter char: o

type  = 1
enter num: 25


1 = g
2 = o
3 = 25
```

+1. Изобразить график функции y = x*x;

Математическая модель: y = x*x.

Список идентификаторов:

Имя	Значение	Тип
width	Количество пикселей монитора по горизонтали	int
high	Количество пикселей монитора по вертикали	int
x0	Текущее значение x	int
y0	Текущее значение y	int
x1	Следующее значение x	int
y1	Следующее значение y	int
i	Счетчик цикла	int

Код программы:

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>

int main(void)
{
int GrDr,GrMod,rez ;
GrDr=DETECT ;
initgraph(&GrDr,&GrMod, 0) ;
rez=graphresult();
if(rez != grOk)
{
printf("\n Ошибка инициализации графики") ; return(0) ;
}
int width = 640, high = 480;
line(width / 2, 0, width / 2, high);
line(0, high / 2, width, high / 2);
```

```
int x0 = -15, y0 = x0*x0;
int x1 = -14, y1 = x1 * x1;
for (int i = 0; i < 20; i++) {
	line(width/2 + x0*32, high / 2 - y0*32, width / 2 + x1*32, high / 2 -y1*32);
	x0 = x1;
	y0 = y1;
	x1 = x0 + 1;
	y1 = x1 * x1;
}
getch();
closegraph();
return(1);
}
```

Результат выполненной работы: