Южно-Уральский государственный колледж  
  
  
  
  
  
  
  
  
**Лабораторная работа №14 по теме  
Разработка и использование рекурсивных процедур и функций   
Вариант 11**

Выполнил студент группы ИСп320Д:  
Бураков Х.Ш.  
Проверил преподаватель:  
Замятин А.П.

Челябинск, 2023

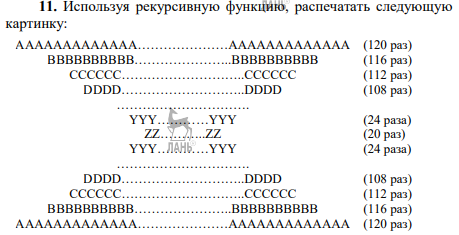
**Цель работы:**

1) изучить особенности работы с рекурсивными функциями в языке Си;

2) написать программу обработки данных с использованием рекурсивных данных в соответствии с заданным вариантом (тестовые параметры задаются самостоятельно);

3) сделать вывод о полученных результатах работы программы.

**Задание:**



**Результат:**

#include <stdio.h>

#include <string.h>

struct PaintFig // для печати узора символов

{

// Конструктор для структуры

PaintFig(char from, char to, int min\_n, int tab)

: from(from), to(to - 1), min\_n(min\_n), tab(tab), down(1)

{ // Инициализация массива tab\_str пробелами и добавление нужного символа

memset(this->tab\_str, ' ', sizeof(this->tab\_str));

this->tab\_str[tab] = '\0';

}

// Перегрузка оператора вызова функции

void operator()()

{

printR(from);

down = 0;

printR(from - 1);

}

private:

// Рекурсивная функция для печати заданного узора

void printR(char n) {

if (n == to) {

return;

}

// Печать узора для каждой строки

if (down)

{

printR(n - 1);

}

// Печать узора после каждой строки

for (int i = 0; i < n - to - 1; ++i)

{

printf("%s", tab\_str);

}

// Печать символа новой строки после каждой строки

for (int i = 0, end = 16 + (from - n) \* min\_n; i < end; ++i)

{

printf("%c", n);

}

printf("\n");

if (!down)

{

printR(n - 1);

}

}

char from; // начальный узор символа

char to; // конечный узор символа

int min\_n;

int tab; // кол-во табуляций перед каждой строкой

char tab\_str[100]; // строковое представление табуляций

int down; // для определения направления рекурсии

};

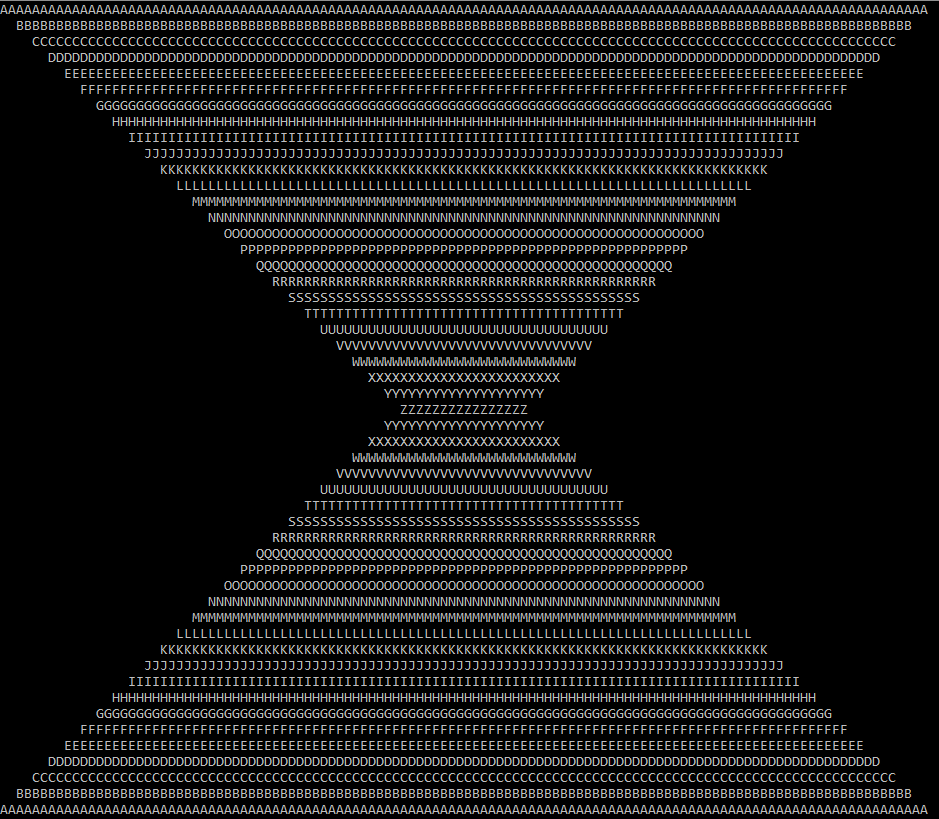
int main()

{

PaintFig('Z', 'A', 4, 2)(); // вызов функции

return 0;

}



**Контрольные вопросы**

1. **Понятие рекурсивных функций.**

Рекурсивная функция — это функция, которая вызывает сама себя.

1. **Особенности использования рекурсивных функций.**

Базовый случай: Это условие, при котором функция прекращает вызывать саму себя и возвращает результат.

Рекурсивный случай: Рекурсивная функция должна также иметь рекурсивный случай, в котором она вызывает саму себя с другими аргументами. Это обеспечивает продолжение выполнения функции до достижения базового случая.

Стек вызовов: При каждом вызове рекурсивной функции создается новый фрейм стека вызовов, который содержит локальные переменные и возвращаемый результат. Каждый фрейм стека вызовов сохраняется в памяти до достижения базового случая, а затем извлекается в обратном порядке для получения результата.

Ограничение глубины рекурсии: Рекурсивные функции могут вызываться множество раз, что может привести к переполнению стека вызовов и вызвать ошибку "Stack Overflow". Поэтому важно обосновывать, что рекурсия будет завершена в конечном итоге или использовать итеративные алгоритмы вместо рекурсивных.

Эффективность: В некоторых случаях рекурсивные функции могут быть менее эффективными, чем итеративные алгоритмы. Это связано с дополнительными расходами на вызов функции и управление стеком вызовов.

**Таким образом**, я изучил особенности работы с рекурсивными функциями в языке Си и написал программу обработки данных с использованием рекурсивных данных в соответствии с заданным вариантом.