|  |
| --- |
| Министерство образования Российской Федерации  Пензенский государственный университет  Кафедра «Вычислительная техника» |
| Отчет  по лабораторной работе №4  по курсу «Логика и основы алгоритмизации в инженерных задачах» на тему «Обход графа в глубину» |
|  |
|  |
| Выполнила студентка группы 19ВВ3:  Ланцов А.С.  Принял:  Митрохин М. А. |
| Пенза  2020 |

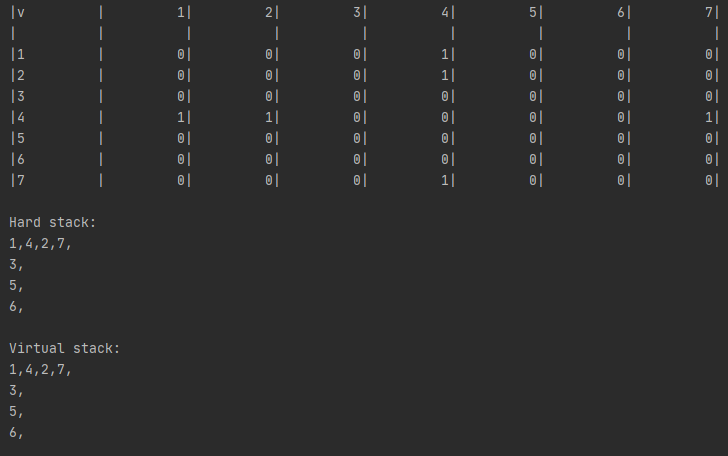
### **Задание 1**

Листинг:

void DFS\_Matrix(mtrx\_grph\_t\* graph, void DFS(mtrx\_grph\_t\*,bool\*,int)){  
 bool\* visited=calloc(graph->size,1);  
  
 int next\_index=0;  
 while (next\_index!=-1){  
 DFS(graph,visited,next\_index);  
 next\_index=fst\_false(visited,graph->size);  
 printf("\n");  
 }  
 free(visited);  
  
}

#define \_PTR\_VAL(value\_type,pointer) \*((value\_type\*)pointer)

void virtualStack(mtrx\_grph\_t\* graph, bool\* visited, int v\_index){  
 list\_t\* vrtx\_stack = list\_create();  
 list\_t\* cycle\_states\_stack = list\_create();  
  
 int i=0;  
 int current\_v\_index=v\_index;  
  
 visited[v\_index]=true;  
 printf("%s,",graph->vertexes[v\_index]);  
  
 for (i; i<graph->size; i++){  
 if (graph->matrix[v\_index][i] && !visited[i]){  
 printf("%s,",graph->vertexes[i]);  
 visited[i]=true;  
  
 append(vrtx\_stack,malloc(sizeof(int)));  
 append(cycle\_states\_stack,malloc(sizeof(int)));  
  
 \_PTR\_VAL(int,cycle\_states\_stack->tail->value)=i;  
 \_PTR\_VAL(int,vrtx\_stack->tail->value)=v\_index;  
  
 current\_v\_index=i;  
 i=-1;  
 }  
  
 v\_index = current\_v\_index;  
  
 if ((i+1)==graph->size){  
 if (cycle\_states\_stack->tail){  
 i = \_PTR\_VAL(int,list\_pop(cycle\_states\_stack))-1;  
 current\_v\_index= \_PTR\_VAL(int,list\_pop(vrtx\_stack));  
  
 v\_index = current\_v\_index;  
 }  
 }  
 }  
}  
  
void hardStack\_lists(list\_graph\_t\* graph, bool\* visited, int v\_index){  
 node\_t\* next = graph->lists[v\_index]->head;  
 int next\_index;  
  
 visited[v\_index]=true;  
 printf("%s,",graph->vertexes[v\_index]);  
 for (int i=0; i<graph->lists[v\_index]->depth; i++){  
 next\_index=getindex((char \*\*) graph->vertexes, graph->size, (char \*) next->value);  
 if (!visited[next\_index]) {  
 visited[v\_index]=true;  
 hardStack\_lists(graph, visited, next\_index);  
 }  
 next = next->next;  
 }  
}

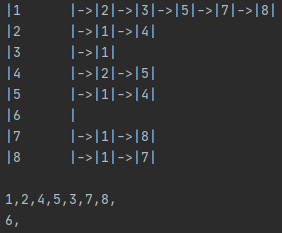
Результат:  


Задание 2.

Листинг:

void hardStack\_lists(list\_graph\_t\* graph, bool\* visited, int v\_index){  
 node\_t\* next = graph->lists[v\_index]->head;  
 int next\_index;  
  
 visited[v\_index]=true;  
 printf("%s,",graph->vertexes[v\_index]);  
 for (int i=0; i<graph->lists[v\_index]->depth; i++){  
 next\_index=getindex((char \*\*) graph->vertexes, graph->size, (char \*) next->value);  
 if (!visited[next\_index]) {  
 visited[v\_index]=true;  
 hardStack\_lists(graph, visited, next\_index);  
 }  
 next = next->next;  
 }  
}

void DFS\_Lists(list\_graph\_t\* graph){  
 bool\* visited=calloc(graph->size,1);  
  
 int next\_index=0;  
 while (next\_index!=-1){  
 hardStack\_lists(graph,visited,next\_index);  
 next\_index=fst\_false(visited,graph->size);  
 printf("\n");  
 }  
  
 free(visited);  
}

Результат:  


Структуры:  
typedef struct node{  
 uint8\_t\* value;  
  
 struct node\* last;  
 struct node\* next;  
}node\_t;  
  
typedef struct list{  
 node\_t\* head;  
 node\_t\* tail;  
  
 int depth;  
  
}list\_t;

typedef struct matrix\_graph{  
 int\*\* src;  
  
 int\*\* matrix;  
 char\*\* vertexes;  
 int size;  
  
}mtrx\_grph\_t;

typedef struct list\_graph{  
 list\_t\*\* lists;  
 uint8\_t\*\* vertexes;  
  
 int size;  
}list\_graph\_t;