GraphSC_main.c

```
int main(int argc, char *argv[]) {
  VARTEX *varray ; // 頂点構造体配列
  int i ,
    vid , // 頂点番号(識別子)
    startv , // 探索開始頂点
    *vertices , // 隣接頂点配列
    *visitedv , // 訪問頂点配列
    nv , // 頂点総数
    degree ; // 次数
    char *stations[] = {"A", "B", "C", "D", "E", "F", "G", "H", "I", "J"};

  int graph_array[11][11] = {
    {10, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
    {1, 2, 2, 7, 0, 0, 0, 0, 0, 0, 0},//A
    {2, 4, 1, 3, 7, 10, 0, 0, 0, 0, 0},//B
    {3, 6, 2, 4, 5, 6, 7, 9, 0, 0, 0},//C
    {4, 2, 3, 5, 0, 0, 0, 0, 0, 0, 0},//D
    {5, 3, 3, 4, 9, 0, 0, 0, 0, 0, 0},//E
    {6, 3, 3, 7, 8, 0, 0, 0, 0, 0, 0},//F
    {7, 4, 1, 2, 3, 6, 0, 0, 0, 0, 0},//G
    {8, 3, 6, 9, 10, 0, 0, 0, 0, 0, 0},//H
    {9, 3, 3, 5, 8, 0, 0, 0, 0, 0, 0},//I
    {10, 2, 2, 8, 0, 0, 0, 0, 0, 0, 0},//J
  } ;

  startv = 探索開始頂点の設定;
  nv = sizeof(stations)/sizeof(*stations);// 頂点総数

  vertices = allocate_arrayI(nv) ; // 隣接頂点配列
  visitedv = allocate_arrayI(nv + 1) ; // 訪問頂点配列
  printf("nv = %d¥n", nv) ;
  varray = allocate_arrayV(nv + 1) ; // 頂点構造体配列の割り当て
  init_arrayV(varray, nv) ; // 隣接頂点配列の初期化


  for(i = 1; i <= nv; i++){
    vid = 頂点;
    degree = 次数;
    create_vlist(vid, degree, varray, graph_array[i][2]を先頭にした配列(隣接
頂点リスト)) ; // 頂点リスト生成
  }
  trav_v_depth(nv, visitedv, varray, startv) ; // 縦型探索

  printf("縦型探索¥n") ;
  for(i = 1; i <= nv; ++i)
    printf("訪問頂点: %d:%s¥n", visitedv[i], stations[visitedv[i]-1]) ; // 訪
```

```
問順に頂点番号を出力

  return ;
}
```