```
// Implémentation d'une pile d'entiers tous différents, compris entre
// 0 et M. On joue sur la confusion entre indice et valeur empilée.

// Convention(s) :
//  − Sommet : case 0 du tableau
//  − "suivant" de la valeur e, stockée dans la case n° i de tab : tab[e]
//  − "Fond" : élément dont le suivant est 0, ie n° de case contenant 0
//
// Exemple : pile | 8 |                 _____________________
//                | 5 |      tab : |8| | |0| |3| | |5| | |
//                | 3 |            ----------------------
//                -----              0 1 2 3 4 5 6 7 8 9 10


#define M 10
typedef int pile[M] ;

// ---------- Primitives courantes ----------
void init_pile(pile t)
{ t[0] = 0 ; }

void empiler(pile t , int e)
{
  t[e] = t[0] ;
  t[0] = e ;
}

int depiler(pile t)
{
  // Rq: si la pile est vide, on renvoie 0 sans autre pb
  int tmp = t[0] ;
  t[0] = t[t[0]] ;
  return tmp ;
}

int pile_vide(pile t)
{
  return  (t[0] == 0) ;
}

// ----------Fonctions auxiliaires----------
void voir_pile(pile t)
{
  int i ;
  if (pile_vide(t)) {printf("Pile vide\n") ; return ;}
  for (i=t[0] ; i!=0 ; i=t[i])
    printf("%d|", i) ;
  printf("\n") ;
}
```



```
// ----------Programme principal----------
// Concu pour tester la pile

pile tab ;

int main()
{
  init_pile(tab) ;
  voir_pile(tab) ;
  printf("J'empile %d\n", 3) ; empiler(tab, 3) ;
  printf("J'empile %d\n", 5) ; empiler(tab, 5) ;
  printf("J'empile %d\n", 8) ; empiler(tab, 8) ;
  printf("J'empile %d\n", 9) ; empiler(tab, 9) ;
  voir_pile(tab) ;
  printf("Je dépile\n") ; depiler(tab) ;
  printf("Je dépile\n") ; depiler(tab) ;
  voir_pile(tab) ;
  printf("Je dépile 3 fois\n") ;
  depiler(tab) ; depiler(tab) ; depiler(tab) ;
  voir_pile(tab) ;
}
```