nov. 19, 12 9:32 **espaces.h** Page 1/1

```
#ifndef ESPACES_H
#define ESPACES_H

int compteEspace(char chaine[]);
void supprimeEspacePtr(char *chaine);

#endif
```

nov. 20, 12 10:51 **espaces.c** Page 1/1

```
#include "espaces.h"

int
compteEspace(char chaine[]){
  int cptr = 0;
  for(int i = 0; chaine[i] != '\0'; ++i)
    if (chaine[i] == ' ' || chaine[i] == '\t')
      ++cptr;
  return cptr;
}

void
supprimeEspacePtr(char *chaine){
  char *nouvelleChaine = chaine;
  while (*chaine != '\0'){
    if (*chaine != ' ' && *chaine != '\t')
      *nouvelleChaine++ = *chaine;
    ++chaine;
  }
  *nouvelleChaine = '\0';
}

/* Les versions qui suivent utilisent une variable  qui compte le nombre
   d'elements a supprimer

void
supprimeEspacePtr(char *chaine){
  int aSupprimer = 0;
  while (*chaine != '\0'){
    if (*chaine == ' ' || *chaine == '\t')
      ++aSupprimer;
    else
      *(chaine-aSupprimer) = *chaine;
    ++chaine;
  }
  *(chaine-aSupprimer) = *chaine;
}

void
supprimeEspacePtr(char *chaine){
  int aSupprimer = 0;
  while (*(chaine + aSupprimer) != '\0')
    if (*(chaine+aSupprimer) == ' ' || *(chaine+aSupprimer) == '\t')
      ++aSupprimer;
    else{
      *chaine = *(chaine + aSupprimer);
      ++chaine;
    }
  *chaine = '\0';
}

*/
```

nov. 20, 12 9:51 **test_espaces.c** Page 1/1

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <assert.h>

#include "espaces.h"

static void
usage(char *commande){
  fprintf(stderr, "usage : %s \"chaine\"\n",commande);
  exit(EXIT_FAILURE);
}

int
main(int argc,char *argv[]){
  if (argc != 2)
    usage(argv[0]);
  char *chaine= malloc((strlen(argv[1])+1)*sizeof(char));
  assert (chaine != NULL);
  strcpy(chaine,argv[1]);

  printf("La chaine \"%s\" comporte %d espaces ou tabulations\n",chaine,compteEspace(chaine));
  supprimeEspacePtr(chaine);
  printf("Privee de ses espaces, la chaine \"%s\" devient \"%s\"\n)",argv[1],chaine);

  free(chaine);
  return EXIT_SUCCESS;
}
```

nov. 19, 12 10:03 **Makefile** Page 1/1

```
CC = gcc
CFLAGS = -g -std=c99 -Wall -Werror

OUTFILE = test_espaces

OBJS =  test_espaces.o espaces.o

all: $(OUTFILE)

$(OUTFILE): $(OBJS)


test_epaces.o : test_espaces.c espaces.h
espaces.o : espaces.c espaces.h
```

nov. 10, 12 10:13 **carre_magique.h** Page 1/1

```
#ifndef _CARRE_MAGIQUE_H_
#define _CARRE_MAGIQUE_H_

typedef struct carreMagique *carreMagique;

carreMagique creerCarre(int n);
// cree un carre magique de dimension n

void libererCarre(carreMagique c);
// libere la memoire utilisee par le carre magique c

void afficherCarre(carreMagique c);

#endif
```

nov. 20, 12 17:57 **carre_magique.c** Page 1/2

```
#include <stdlib.h>
#include <stdio.h>
#include <assert.h>
#include "carre_magique.h"

struct carreMagique {
  int taille;
  int** carre;
};

static void remplirCarre(carreMagique c);

carreMagique
creerCarre(int n) {
  carreMagique c = malloc(sizeof(*c));
  assert (c != NULL);
  c->taille = n;
  c->carre = (int**) malloc(n*sizeof(int*));
  assert(c->carre != NULL);
  for (int i=0; i<n; i++){
    c->carre[i] = (int*)malloc(n*sizeof(int));
    assert(c->carre[i] != NULL);
    for (int j=0; j<n; j++) // on peut eviter cette initialisation
      c->carre[i][j] = 0;   // en remplacant malloc par calloc
  }
  remplirCarre(c);
  return c;
}

void
libererCarre(carreMagique c) {
  for (int i=0; i < c->taille; i++)
    free(c->carre[i]);
  free(c->carre);
  free(c);
}

static void
remplirCarre(carreMagique c) {
  int n = c->taille;
  c->carre[0][n/2] = 1; //placer le 1
  int lig = 0;
  int col = n/2;

  for (int nb = 2; nb <= n*n;nb++){
    lig = (lig -1 + n) % n;
    col = (col + 1) % n;
    if (c->carre[lig][col] !=  0){
      lig = (lig+ 2) % n;
      col = (col - 1 + n) % n;
    }
    c->carre[lig][col] = nb;
  }
}

void
afficherCarre(carreMagique c) {
  int n = c->taille;
  for (int i=0; i<n; i++){
    for (int j=0; j<n; j++)
      printf("%d \t", c->carre[i][j]);
    printf("\n");
```

```
  }
  printf("\n");
}
```



```
#include <stdlib.h>
#include <stdio.h>
#include "carre_magique.h"

static void
usage(char *command){
  fprintf(stderr,"Usage : %s <nombre impair> \n",command);
  exit(EXIT_FAILURE);
}

int
main(int argc, char* argv[]){
  if (argc != 2)
    usage(argv[0]);
  int taille = atoi(argv[1]);
  if (taille % 2 != 1)
    usage(argv[0]);
  carreMagique c = creerCarre(taille);
  afficherCarre(c);
  libererCarre(c);
  return EXIT_SUCCESS;
}
```

```
CC = gcc
CFLAGS = -g -std=c99 -Wall -Werror

OUTFILE = testCarre

OBJS =  carre_magique.o testCarre.o

all: $(OUTFILE)

$(OUTFILE): $(OBJS)


testCarre.o : testCarre.c carre_magique.h
carre_magique.o : carre_magique.c carre_magique.h
```