Les Génies du Ponant

Machine à courber les bananes

1.13.43

14/05/2024

Sommaire

src1.c

src2.c

src3.c

src1.c

Programme de jeu de sudoku

FAUCHET Malo
0.2
2023-11-20

Ce programme permet de jouer au sudoku en selectionnant une grille au lancement du programme. Le programme s arrete lorsque la grille est pleine.

DEFINES

#define n 5
Taille d un bloc de la grille

#define TAILLE n*n
*Taille de la grille (n*n)*

STRUCTURES

aleatoire
Structure contenant deux entiers

```
typedef struct {
    int lorem;      commentaire lorem
    int ipsum;      commentaire ipsum
} aleatoire;
```

aleatoire2
Structure contenant deux entiers 2

```
typedef struct {
    int lorem2;     commentaire lorem 2
    int ipsum2;     commentaire ipsum 2
} aleatoire2;
```

GLOBALES

CELLULE_VIDE
Caractere representant une cellule vide

```
const char CELLULE_VIDE = '.' ;
```

VENIAM
Entier representant le nombre de grilles disponibles

```
int VENIAM = 2;
```

FONCTIONS

int main()

Description : Programme principal

Retourne : Code de sortie du programme (0: sortie normale).

Détail : Le programme principal permet de jouer une partie de sudoku en selectionnant
une grille au lancement du programme. Le programme s arrete lorsque
la grille est pleine.

int chargerGrille(tGrille grille)

Description : Charge une grille de jeu a partir d un fichier

Paramètre : grille Grille de jeu a initialiser

Retourne : faux si tout s est bien passe, sinon vrai

Détail : La fonction charge une grille de jeu a partir d un fichier
dont le nom est saisi au clavier.

void genererNbAleatoire(int *num, int maximum)

Description : Genere un nombre aleatoire entre 1 et maximum (inclus)

Paramètre : num la variable dans laquelle stocker le nombre aleatoire

Paramètre : maximum borne maximale pour la generation du nombre aleatoire

void afficherGrille(tGrille grille)

Description : Affiche la grille de jeu de maniere lisible

Paramètre : grille Grille de jeu a afficher

src2.c

Implémentation de hashmap générique.

DEFINES

#define INITIAL_SIZE (256)
Taille initiale de la table de hachage.

#define MAX_CHAIN_LENGTH (8)
Longueur maximale d'une chaîne avant de commencer à lier.

STRUCTURES

hashmap_element
hashmap_element

```
typedef struct {
        key;        clé
        int in_use;      est-ce que cet emplacement est utilisé ?
        any_t data;     data used
} hashmap_element;
```

hashmap_map
hashmap_map

```
typedef struct {
        int table_size;        taille de la table
        int size;   nombre d'éléments dans la table

} hashmap_map;
```

GLOBALES

lorem_ipsum_length
Longueur du texte lorem ipsum utilisé pour générer des clés.

```
const int lorem_ipsum_length = 1435;
```

load_factor
Facteur de charge de la hashmap.

```
const float load_factor = 0.75;
```

FONCTIONS

map_t hashmap_new()

Description : une hashmap vide, ou NULL en cas d'erreur.

unsigned long crc32(const unsigned char *s, unsigned int len)

Description : Retourne un CRC de 32-bit du contenu des buffer.

unsigned int hashmap_hash_int(hashmap_map *m, char *keystring)

Description : Fonction d'hashage pour une chaîne de caractères.

int hashmap_hash(map_t in, char *key)

Description : Retourne l'entier de l'emplacement dans les données pour stocker le point sur l'élément, ou MAP_FULL.

int hashmap_rehash(map_t in)

Description : Double la taille de la hashmap, et rehashes tous les éléments

int hashmap_put(map_t in, char *key, any_t value)

Description : Ajoute un pointeur à la hashmap avec une clé

src3.c

Allocateur de mémoire basé sur malloc.

Source : https://github.com/5cover/MyMalloc

DEFINES

#define DUMP_BMP_HEIGHT 8
Entier : hauteur en pixels des images crées par les fonctions heapDumpChunksBitmap et heapDumpDataBitmap

#define HEAP_SIZE 256
Entier : la taille du tas

STRUCTURES

Chunk
Bloc de mémoire allouée

```
typedef struct {
    intptr_t start; Adresse de début du bloc de mémoire allouée
    size_t size;    Taille du bloc de mémoire allouée
} Chunk;
```

mlqkdsjgfmoqisj
Bloc de mémoire allouée

```
typedef struct {
    int amljdsgf;  Adresse de début du bloc de mémoire allouée

} mlqkdsjgfmoqisj;
```

GLOBALES

gs_chunkCount
Le nombre de blocs actuellement alloués

```
const size_t gs_chunkCount = 0;
```

FONCTIONS

void myAlloc(size_t size)

Description : Alloue de la mémoire.

Paramètre : size_t size Nombre d'octets contigus à allouer

Retourne : void* Pointeur vers le bloc de mémoire alloué, ou NULL si l'allocation a échoué.

Détail : Le comportement ce cette fonction est identique à la fonction malloc de la bibliothèque standard du langage C.

void myFree(void const *ptr)

Description : Libère de la mémoire précédemment allouée via myAlloc.

Paramètre : void ptr Pointeur vers le bloc de mémoire à libérer

Détail :Le comportement ce cette fonction est identique à la fonction free de la bibliothèque standard du langage C.

bool isAreaAllocated(intptr_t start, size_t size)

Description : Détermine si un zone de la mémoire est au moins en partie allouée.

Paramètre : intptr_t start Adresse de début de la zone mémoire

Paramètre : size_t size Taille (longueur) de la zone mémoire

Retourne : bool vrai si au moins un octent de la zone de mémoire spécifiée est actuellement alloué, faux sinon.

void removeChunkAt(size_t index)

Description : supprime le bloc de mémoire à l'adresse spécifiée

Paramètre : index Index du bloc de mémoire à supprimer dans gs_chunks.

bool rangesOverlap(intptr_t x1, intptr_t x2, intptr_t y1, intptr_t y2)

Description : Détermine si deux intervalles se chevauchent.

Paramètre : intptr_t x1 début de l'intervalle 1

Paramètre : intptr_t x2 fin de l'intervalle 1

Paramètre : intptr_t y1 début de l'intervalle 2

Paramètre : intptr_t y2 fin de l'intervalle

Retourne : bool vrai si les intervalles tel que [x1 ; x2] et [y1 ; y2] se chevauchent, faux sinon.

void heapDumpChunksConsole(void)

Description : Affiche dans la console l'état de la mémoire, quels blocs sont allouées, etc...

void heapDumpChunksBitmap(char const *filename)

Description : Crée un fichier bitmap schématisant les blocs actuellement alloués dans la mémoire.

Paramètre : char* filename Le nom du fichier à créer

void heapDumpDataBitmap(char const *filename)

Description : Crée un fichier bitmap représentant en nuances de gris les

données actuellement contenues dans la mémoire.

Paramètre : char* filename Le nom du fichier à créer