

```

1  /*-----
•  --*/
2  /*» actions.c»» fichier contenant la déclaration des »» */
3  /*» » » différentes fonctions utilisées»» » » */
4  /*-----
•  --*/
5
6
7  #include "actions.h"
8
9
10
11 /*-----
•  --*/
12 /*CreateAction» » Insérer une nouvelle action à la suite de l'agenda» */
13 /*» » » » » » » » » */
14 /*En entrée:» » liste_actions»Structure de type actions_t» */
15 /*» » » chaine_actions» Chaîne de caractères »» » */
16 /*» » » » » » » » » */
17 /*En sortie:» » La fonction retourne une nouvelle liste chaînée contenant
18 /*» » une nouvelle action si la plage horaire n'est pas déjà occupée» */
19 /*-----
•  --*/
20
21 void CreateAction (actions_t * liste_actions, char * chaine_actions)
22 {
23     int » » » i;
24     char » » » jour_heure[3];
25     char » » » nom_action[10];
26     actions_t » » * ptrAction;
27
28     for (i=0; i<13; i++)
29     {
•      » » » /*on divise la case chaine_actions avec le jour et l'heure de
        côté */
30         if (i<3)
31         {
•          » » » /*et le nom de l'action de l'autre*/
32             jour_heure[i] = chaine_actions[i];
33         }
34         else
35         {
36             nom_action[i-3] = chaine_actions[i];
37         }
38     }
39
40     ptrAction = Recherche( liste_actions, jour_heure); /*ptrAction représente
•      la case précédente avant laquelle */
41     » » » » » /*il faut insérer la nouvelle action*/
42     if (* ptrAction != NULL)
43     {
44         if (jour_heure != (* ptrAction)->suiv->jour_heure)
45         {
46             AlloueAction( * ptrAction, jour_heure, nom_action); /*on alloue un
•              nouvel élément*/
47         }
48         else

```



```

49     ....{
50     .....printf("La plage horaire est deja occupee, vous ne pouvez pas ajouter
    •      d'autres elements");.....
51     ....}
52     ..}
53     ..else
54     ..{
55     ....AlloueAction( * ptrAction, jour_heure, nom_action);
56     ..}
57     ..
58     }
59
60
61
62
63
64     /*-----
    •     --*/
65     /*AlloueAction» » Allouer de l'espace mémoire à la suite du pointeur */
66     /*» » » Insérer le nouvel élément dans cet espace mémoire*/
67     /*» » » » » » » */
68     /*En entrée:» » ptrAction»Pointeur sur l'adresse de l'élément précédent»
69     /*» » » jour» » Entier représentant Le jour et l'heure */
70     /*» » » nom» » Chaîne de caractères représentant Le */
71     /*» » » » nom de l'action» » » */
72     /*» » » » » » » */
73     /*En sortie:» » La fonction retourne une nouvelle liste chaînée contenant
74     /*» » un nouvel espace mémoire et une nouvelle action» » */
75     /*-----
    •     --*/
76
77     void AlloueAction( actions_t * ptrAction, char jour [3], char action [10]
78     {
79     ..int » » » i;
80     ..actions_t » *»nouv;
81     ..
82     ..nouv = (actions_t *) malloc(sizeof(actions_t));
83     ..nouv->suiv = ptrAction->suiv;
84     ..ptrAction = nouv;
85     ..
86     ..
87     ..
88     ..for (i=0; i<3; i++)
89     ..{» » » » /*on divise la case chaîne_actions avec le jour et l'heure de
    •      côté */» » » /*et le nom de l'action de l'autre*/
90     ....nouv->jour_heure[i] = jour[i];
91     ..}
92     ..for (i=0; i<10; i++)
93     ..{» » » » /*on divise la case chaîne_actions avec le jour et l'heure de
    •      côté */» » » /*et le nom de l'action de l'autre*/
94     ....nouv->nom_action[i] = action[i];
95     ..}
96     }
97
98

```

```

99      ↵
100     ↵
101     ↵
102     ↵
103     /*-----
    •   --*/↵
104     /*Recherche» Rechercher une valeur dans une liste chaînée» */↵
105     /*» » » » » » » » */↵
106     /*En entrée:» » tete_liste» Structure de type actions_t» */↵
107     /*» » » val» Entier représentant le jour et l'heure */↵
108     /*» » » » » » » » */↵
109     /*En sortie:» » La fonction retourne l'adresse de l'élément précédent »
110     /*» » » où il faut insérer la nouvelle valeur ou supprimer» */↵
111     /*» » » une valeur existante» » » » */↵
112     /*-----
    •   --*/↵
113     ↵
114     actions_t ** Recherche ( actions_t * tete_liste, char val [3])↵
115     {↵
116         ..int » » i;↵
117         ..actions_t » ** cour = &tete_liste;↵
118         ..↵
119         ..for (i=0; i<3; i++)↵
120         ..{↵
121             ...while ( ( (*cour) != NULL ) && ( (*cour)->suiv->jour_heure[i] != val
    •         && ( (*cour)->suiv->jour_heure[i] < val[i]) )↵
122             ....{↵
123                 .....*cour = (*cour) -> suiv;↵
124             ....}↵
125         ..}↵
126         ..↵
127         ..return cour;↵
128     }↵
129     ↵
130     ↵
131     ↵
132     ↵
133     ↵
134     /*-----
    •   --*/↵
135     /*Sauvegarde» » fonction qui sauvegarde la SDD sous la forme d'un fichier
136     /*» » » » » » » » */↵
137     /*En entrée:» » fichier_sauvegarde »Fichier créer dans semaine.c» */↵
138     /*» » » liste_actions» Structure de type actions_t*/↵
139     /*» » » semaine» » Chaîne de caractères » */↵
140     /*» » » » » » » » */↵
141     /*En sortie:» » Renvoie un fichier contenant une sauvegarde de la SDD*/↵
142     /*-----
    •   --*/↵
143     ↵
144     void Sauvegarde ( FILE * fichier_sauvegarde, actions_t * liste_actions,
    •     * semaine)↵
145     {↵
146         ..int» » i;↵
147         ..actions_t » * cour = liste_actions;↵

```

```

148     ..char» »   action[19];↵
149     ..↵
150     ..while ( cour != NULL)↵
151     ..{↵
152     ....for (i=0;i<19;i++)↵
153     .....{↵
154     .....if (i<6)↵
155     .....{↵
156     » action[i]= semaine[i];↵
157     .....}↵
158     .....else if (i<9)↵
159     .....{↵
160     » ..action[i]=cour->jour_heure[i-6];↵
161     .....}↵
162     .....else↵
163     .....{↵
164     » action[i]=cour->nom_action[i-9];↵
165     .....}↵
166     ....}↵
167     ....↵
168     ....fputs (action, fichier_sauvegarde);↵
169     ....cour = cour->suiv;↵
170     ..}↵
171     ..↵
172 }↵
173 ↵
174 ↵
175 ↵
176 ↵
177 ↵
178 /*-----
  • --*/↵
179 /*CreateListeFromActions» Fonction qui cherche si une action est présente
180 /*» » » dans l'agenda» » » » */↵
181 /*» » » » » » » */↵
182 /*En entrée:» » liste_actions» Structure de type actions_t*/↵
183 /*» » » rechAction» Chaîne de caractères» » */↵
184 /*» » » » » » » */↵
185 /*En sortie:» » Renvoie, si l'action est présente, Les jours qui »*/↵
186 /*» » contiennent une ou plusieurs fois cette action sous forme de »*/↵
187 /*» » tableau de caractères» » » » */↵
188 /*-----
  • --*/↵
189 ↵
190 char * CreateListeFromActions ( actions_t * liste_actions, char * rechAc
191 {↵
192     ..char » » * liste_jour;↵
193     ..int » »   i = 0;↵
194     ..actions_t » » cour = liste_actions;↵
195     ..↵
196     ..liste_jour = "";↵
197     ..↵
198     ..while ( cour != NULL)↵
199     ..{↵
200     ....if (cour->nom_action == rechAction)↵

```

```

201     ....{
202     .....if ( i == 0 )
203     .....{
204     » liste_jour[i] = cour->jour_heure[0];
205     » i = i+1;
206     .....}
207     .....else
208     .....{
209     » if ( liste_jour[i-1] != cour->jour_heure[0])
210     » ..{
211     » ...liste_jour[i] = cour->jour_heure[0];
212     » ...i = i+1;
213     » ..}
214     .....}
215     }
216     ....}
217     ....cour = cour->suiv;
218     ..}
219     ..return liste_jour;
220 }
221
222
223
224
225
226 /*-----
  • --*/
227 /*SupprimeAction» Fonction qui supprime une action en connaissant */
228 /*» » » La date et L'heure» » » » */
229 /*» » » » » » » » */
230 /*En entrée:» » liste_actions Structure de type actions_t» */
231 /*» » » val» » Entier représentant Le jour et L'heure» */
232 /*» » » » » » » » */
233 /*En sortie:» » La fonction retourne une nouvelle liste chaînée ne */
234 /*» » contenant plus l'action à supprimer si elle existe» » */
235 /*-----
  • --*/
236
237 void SupprimeAction (actions_t * liste_actions, char val [3])
238 {
239     ..int»» i;
240     ..actions_t » * cour = liste_actions;
241     ..actions_t » ** ptrAction;
242     ..
243     ..ptrAction = Recherche (cour, val);
244     ..
245     ..for (i=0; i<3 ; i++)
246     ..{
247     ....if ((* ptrAction) != NULL)
248     ....{
249     .....if (val[i] == (*ptrAction)->suiv->jour_heure[i])
250     .....{
251     » LibérerAction( * ptrAction);» /*on supprime un élément*/
252     .....}
253     .....else

```

```
254     .....{
255     » printf("La plage horaire ne contient aucun élément, il n'y a pas d'act:
    •     supprimer"); .....
256     .....}
257     ....}
258     ....else
259     ....{
260     .....printf("La plage horaire ne contient aucun élément, il n'y a pas
    •     d'action à supprimer");
261     ....}
262     ..}
263 }
264
265
266
267
```