Solution exercice 5.1

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

#define NOM_FICHIER "fichier.txt"

int main(void) {
   FILE* f = fopen(NOM_FICHIER, "r");
   if (!f) { // Si ouverture du fichier impossible
      printf("Ouverture du fichier \"%s\" impossible.\n", NOM_FICHIER);
      return EXIT_FAILURE;
   } else {
      int n;
      while ( fscanf(f, "%d", &n) != EOF )
         printf("%d\n", n);
      // fermer le fichier
      fclose(f);
      return EXIT_SUCCESS;
   }
}
```

Autres variantes possibles :

```c
while ( fscanf(f, "%d", &n) == 1 )
   printf("%d\n", n);

while ( !feof(f) )
   if ( fscanf(f, "%d", &n) == 1 ) {
      printf("%d\n", n);

do {
   if ( fscanf(f, "%d", &n) == 1 )
      printf("%d\n", n);
} while ( !feof(f) );


// Attention! Les variantes ci-dessous fonctionnent si le fichier
// contient au moins un entier... mais pas s'il n'en contient aucun
while (1) {
   fscanf(f, "%d", &n);
   printf("%d\n", n);
   if (feof(f)) break; // pas des plus esthétiques
}

while ( !feof(f) ) {
   fscanf(f, "%d", &n);
   printf("%d\n", n);
}
```

Solution exercice 5.2

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

#define TAILLE_MAX_NOM 20
#define TAILLE_MAX_PRENOM 15

typedef char Nom[TAILLE_MAX_NOM + 1];
typedef char Prenom[TAILLE_MAX_PRENOM + 1];
typedef unsigned short Age;

typedef struct {
   Nom nom;
   Prenom prenom;
   Age age;
} Personne;

void saisie(char* chaine, size_t taille);
void clear_stdin(void);

int main(void) {
   const char* const NOM_FICHIER = "personnes.dat";
   FILE* fichier = fopen(NOM_FICHIER, "wb");
   if (!fichier) {
      printf("Desole! Le fichier \"%s\" n'a pas pu etre ouvert", NOM_FICHIER);
      return EXIT_FAILURE;
   }

   Personne p;
   printf("--- Pour finir la saisie, donnez un nom 'vide' ---\n");
   do {
      printf("\nNom: ");
      saisie(p.nom, TAILLE_MAX_NOM);
      if (strlen(p.nom) == 0)
         break;
      printf("Prenom : ");
      saisie(p.prenom, TAILLE_MAX_PRENOM);
      printf("Age: ");
      scanf("%hu", &p.age);
      clear_stdin();
      fwrite(&p, sizeof(Personne), 1, fichier);
   } while (1);

   fclose(fichier);
   return EXIT_SUCCESS;
}

void saisie(char* chaine, size_t taille) {
   fgets(chaine, (int) taille + 1, stdin);
   clear_stdin();
   for (size_t i = 0; i < taille; ++i)
      if (chaine[i] == '\n') {
         chaine[i] = '\0';
         break;
      }
}

void clear_stdin(void) {
   fseek(stdin, 0, SEEK_END);
}
```

Solution exercice 5.3

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

#define TAILLE_MAX_NOM 20
#define TAILLE_MAX_PRENOM 15

typedef char Nom[TAILLE_MAX_NOM + 1];
typedef char Prenom[TAILLE_MAX_PRENOM + 1];
typedef unsigned short Age;

typedef struct {
   Nom nom;
   Prenom prenom;
   Age age;
} Personne;

int main(void) {
   const char* const NOM_FICHIER_BINAIRE = "personnes.dat";
   FILE* fichier_binaire = fopen(NOM_FICHIER_BINAIRE, "rb");
   if (!fichier_binaire) {
      printf("Desole! Le fichier \"%s\" n'a pas pu etre ouvert\n",
             NOM_FICHIER_BINAIRE);
      return EXIT_FAILURE;
   } else {
      const char* const NOM_FICHIER_TEXTE = "personnes.txt";
      FILE* fichier_texte = fopen(NOM_FICHIER_TEXTE, "w");
      if (!fichier_texte) {
         printf("Desole! Le fichier \"%s\" n'a pas pu etre ouvert\n",
                NOM_FICHIER_TEXTE);
         fclose(fichier_binaire);
         return EXIT_FAILURE;
      } else {
         Personne p;
         fprintf(fichier_texte, "%-*s %-*s %s\n",
                 TAILLE_MAX_NOM, "Nom", TAILLE_MAX_PRENOM, "Prenom", "Age");
         while ( fread(&p, sizeof(Personne), 1, fichier_binaire) ) {
            fprintf(fichier_texte, "%-*s %-*s %3hu\n",
                    TAILLE_MAX_NOM, p.nom, TAILLE_MAX_PRENOM, p.prenom, p.age);
         }
         fclose(fichier_binaire);
         fclose(fichier_texte);
         return EXIT_SUCCESS;
      }
   }
}
```

Solution exercice 5.4

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>

#define TAILLE_MAX_NOM 20
#define TAILLE_MAX_PRENOM 15

typedef char Nom[TAILLE_MAX_NOM + 1];
typedef char Prenom[TAILLE_MAX_PRENOM + 1];
typedef unsigned short Age;

typedef struct {
   Nom nom;
   Prenom prenom;
   Age age;
} Personne;

void lire(char* chaine, size_t taille);

int main(void) {
   const char* const NOM_FICHIER = "personnes.dat";
   FILE* fichier = fopen(NOM_FICHIER, "rb");
   if (!fichier) {
      printf("Desole! Le fichier \"%s\" n'a pas pu etre ouvert\n", NOM_FICHIER);
      return EXIT_FAILURE;
   } else {
      Personne p;
      Nom nomRecherche;
      printf("Quel nom recherchez-vous ? : ");
      lire(nomRecherche, TAILLE_MAX_NOM);
      bool trouve = false;
      while ( !trouve && fread(&p, sizeof(Personne), 1, fichier) )
         trouve = strcmp(p.nom, nomRecherche) == 0;
      fclose(fichier);
      if (trouve)
         printf("%s %s, %hu ans\n", p.prenom, p.nom, p.age);
      else
         printf("Desole! Le nom \"%s\" ne figure pas dans le fichier\n",
                nomRecherche);
      return EXIT_SUCCESS;
   }
}

void lire(char* chaine, size_t taille) {
   fgets(chaine, (int) taille + 1, stdin);
   fseek(stdin, 0, SEEK_END);
   for (size_t i = 0; i < taille; ++i)
      if (chaine[i] == '\n') {
         chaine[i] = '\0';
         break;
      }
}
```

Solution exercice 5.5

```c
#include <inttypes.h>
#include <stdio.h>
#include <stdlib.h>

#define TAILLE_MAX_NOM 20
#define TAILLE_MAX_PRENOM 15

typedef char Nom[TAILLE_MAX_NOM + 1];
typedef char Prenom[TAILLE_MAX_PRENOM + 1];
typedef unsigned short Age;

typedef struct {
   Nom nom;
   Prenom prenom;
   Age age;
} Personne;

int main(void) {
   const char* const NOM_FICHIER = "personnes.dat";
   FILE* fichier = fopen(NOM_FICHIER, "rb"); // accès direct en lecture seule
   size_t nbEnregistrements;
   int rang; // rang de la personne recherche dans le fichier

   if (!fichier) {
      printf("Desole! Le fichier \"%s\" n'a pas pu etre ouvert\n", NOM_FICHIER);
      return EXIT_FAILURE;
   } else {
      // Déterminer combien d'enregistrements contient le fichier
      fseek(fichier, 0, SEEK_END);
      nbEnregistrements = (size_t)ftell(fichier) / sizeof(Personne);
      if (nbEnregistrements == 0)
         printf("Le fichier \"%s\" est vide\n", NOM_FICHIER);
      else {
         printf("Quel rang recherchez-vous ? : ");
         scanf("%d", &rang);
         printf("Le fichier contient %" PRIuMAX " enregistrement(s).\n",
                (uintmax_t) nbEnregistrements);
         if (rang <= 0 || (size_t) rang > nbEnregistrements)
            printf("Desole! Aucune personne ayant ce rang "
                   "ne figure dans le fichier.\n");
         else {
            Personne p;
            // Se positionner sur l'enregistrement concerné
            fseek(fichier, (rang - 1) * (int) sizeof(Personne), SEEK_SET);
            // Lire les infos correspondantes
            fread(&p, sizeof(Personne), 1, fichier);
            // Afficher le résultat à l'écran
            printf("La personne de rang %d est : ", rang);
            printf("%s %s, %hu ans\n", p.prenom, p.nom, p.age);
         }
      }
      fclose(fichier);
      return EXIT_SUCCESS;
   }
}
```

Solution exercice 5.7

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

int main(void) {

   const char* const NOM_FICHIER = "Ex5-7.c";
   FILE* f = fopen(NOM_FICHIER, "r");

   if (!f) { // Si ouverture du fichier impossible
      printf("Ouverture du fichier \"%s\" impossible.\n", NOM_FICHIER);
      return EXIT_FAILURE;
   } else {
      // Récupérer la taille en octets du fichier
      fseek(f, 0, SEEK_END);
      const size_t NB_OCTETS = (size_t) ftell(f);
      // Créer dynamiquement un buffer dans lequel sera stocké le contenu
      // du fichier
      char* buffer = (char*) calloc(NB_OCTETS + 1, sizeof(char));
      if (!buffer) {
         printf("Memoire insuffisante pour creer le buffer.\n");
         fclose(f);
         return EXIT_FAILURE;
      } else {
         // Se repositionner au début du fichier
         rewind(f); // ou fseek(f, 0, SEEK_SET);
         // Lire le contenu du fichier d'un seul tenant et le stocker dans buffer
         fread(buffer, NB_OCTETS, 1, f);
         // Afficher buffer à l'écran
         printf("%s\n\n", buffer);
         // Récupérer la mémoire allouée pour buffer
         free(buffer);
         fclose(f);
         return EXIT_SUCCESS;
      }
   }
}
```