Listing 1 – ex1_pf.c

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

// a function to choose an integer value
int saisie_entier_borne(int min, int max)
{
  int val = min -1;

  if(min > max)
  {
        val = max;
  }
  else
  {
                while((val < min) || (val > max))
                {
                        fprintf(stdout,"\nSaisir un entier dans l'
                            intervalle [%d,%d] :\n",min,max);
                        scanf("%d",&val);
                }
  }

  return val;
}

// display the integer (first type)
void affiche_entier(int val)
{
        fprintf(stdout,"\n%d\n",val);
}

// display the integer (second type)
void affiche_entier_fioriture(int val)
{
        fprintf(stdout,"\nOh ! Le bel entier : %d.\n",val);
}


int main()
{
        // two types of function pointers
        int (*pf_get_int) (int, int) = NULL;
        void (*pf_disp_int) (int) = NULL;
        int val = 0;

        fprintf(stdout,"\nJe pointe ici : %p\n",pf_get_int);

        pf_get_int = &saisie_entier_borne; // choose the integer
        fprintf(stdout,"\nJe pointe ici : %p\n",pf_get_int);
        val = (*pf_get_int) (0,100);

        pf_disp_int = &affiche_entier; // print the integer
        fprintf(stdout,"\nJe pointe ici : %p\n",pf_disp_int);
        (*pf_disp_int) (val);

        pf_disp_int = &affiche_entier_fioriture; // print the other
             way
        fprintf(stdout,"\nJe pointe ici : %p\n",pf_disp_int);
        (*pf_disp_int) (val);

        return 0;
```

```
60 }
```

Listing 2 – ex2_pf.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  /* retourne un entier saisie entre deux bornes */
5  int saisie_entier_borne(int min, int max)
6  {
7    int val = min -1;
8    if(min > max)
9    { val = max; }
10   else
11   {             while((val < min) || (val > max))
12                 {   printf(stdout,"\nSaisir␣un␣entier␣dans␣l'
                       intervalle␣[%d,%d]␣:\n",min,max);
13                     scanf("%d",&val);
14                 }
15   }
16   return val;
17 }
18
19 /* retourne un entier positif saisie entre deux bornes */
20 int saisie_entier_borne_abs(int min, int max)
21 {
22   int val =  saisie_entier_borne(min,max);
23   if(val < 0)   val = -val;
24   return val;
25 }
26
27 /* retourne un entier saisie par la fonction épointe par pf */
28 int saisie_entier_gen(int min, int max, int (*pf) (int, int))
29 {return (*pf) (min,max);}
30
31 /* Affiche un entier */
32 void affiche_entier(int val)
33 {fprintf(stdout,"\n%d\n",val);}
34
35 /* Affiche un entier avec fioritures */
36 void affiche_entier_fioriture(int val)
37 {fprintf(stdout,"\nOh␣!␣Le␣bel␣entier␣:␣%d.\n",val);}
38
39 /* Affiche une adresse en émmoire */
40 void affiche_adresse(void *p)
41 {fprintf(stdout,"\nJe␣pointe␣ici␣:␣%p\n",p);}
42
43
44 /* Affiche un entier avec la fonction épointe par pf */
45 void affiche_entier_gen(int val, void (*pf) (int))
46 {
47         (*pf) (val); // we exploit the function pointer!!
48 }
49
50
51
52 int main()
53 {
54         int (*pf_get_int) (int, int) = NULL; // le pointeur pour
              choisir l'entier
55         void (*pf_disp_int) (int) = NULL; // le pointeur pour
              afficher l'entier
```

```
56          int val = 0;
57
58          affiche_adresse((void *)pf_get_int); // attention! c'est un
                cast à (void*)
59
60          pf_get_int = &saisie_entier_borne;
61          affiche_adresse((void *)pf_get_int);
62
63          val = saisie_entier_gen(0,100,pf_get_int); // appelle &
                saisie_entier_borne;
64          val = saisie_entier_gen(-100,100,&saisie_entier_borne_abs);
65
66          pf_disp_int = &affiche_entier;
67          affiche_adresse((void *)pf_disp_int); //encore cast
68          affiche_entier_gen(val,pf_disp_int); // appelle &
                affiche_entier
69
70          affiche_entier_gen(val,&affiche_entier_fioriture);
71
72          return 0;
73  }
```

Listing 3 – ex3_pf.c

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   /* retourne un entier saisie entre deux bornes */
5   int saisie_entier_borne(int min, int max)
6   {
7     int val = min -1;
8     if(min > max)
9     {     val = max; }
10    else
11    {             while((val < min) || (val > max))
12                  {
13                          fprintf(stdout,"\nSaisir␣un␣entier␣dans␣l'
                                intervalle␣[%d,%d]␣:\n",min,max);
14                          scanf("%d",&val);
15                  }
16    }
17    return val;
18  }
19
20  /* retourne un entier positif saisie entre deux bornes */
21  int saisie_entier_borne_abs(int min, int max)
22  {
23    int val =  saisie_entier_borne(min,max);
24    if(val < 0)   val = -val;
25    return val;
26  }
27
28  /* retourne un entier saisie par la fonction épointe par pf */
29  int saisie_entier_gen(int min, int max, int (*pf) (int, int))
30  {return (*pf) (min,max);}
31
32  /* retourne un pointeur sur une fonction de saisie */
33  int (* choix_saisie_entier(char c)) (int,int)
34  {
35          if(c != 'a')    return &saisie_entier_borne;
36          else                    return &saisie_entier_borne_abs;
37  }
```

```c

/* Affiche un entier */
void affiche_entier(int val)
{       fprintf(stdout,"\n%d\n",val);}

/* Affiche un entier avec fioritures */
void affiche_entier_fioriture(int val)
{       fprintf(stdout,"\nOh ! Le bel entier : %d.\n",val);}

/* Affiche un entier avec la fonction épointe par pf */
void affiche_entier_gen(int val, void (*pf) (int))
{       (*pf) (val);}

/* Affiche une adresse en émmoire */
void affiche_adresse(void *p)
{       fprintf(stdout,"\nJe pointe ici : %p\n",p);}


int main(){

        int (*pf_get_int) (int, int) = NULL;
        void (*pf_disp_int) (int) = NULL;
        int val = 0;

        affiche_adresse((void *)pf_get_int);

        pf_get_int =  choix_saisie_entier('b');
        affiche_adresse((void *)pf_get_int);
        val = saisie_entier_gen(0,100,pf_get_int);

        val = saisie_entier_gen(-100,100,&saisie_entier_borne_abs);

        pf_disp_int = &affiche_entier;
        affiche_adresse((void *)pf_disp_int);
        affiche_entier_gen(val,pf_disp_int);

        affiche_entier_gen(val,&affiche_entier_fioriture);

        return 0;
}
```

Listing 4 – ex4_pf.c

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

typedef int (*pf_saisie) (int, int);

/* retourne un entier saisie entre deux bornes */
int saisie_entier_borne(int min, int max)
{
  int val = min -1;
  if(min > max){  val = max;  } else
  {     while((val < min) || (val > max))
        {       fprintf(stdout,"\nSaisir un entier dans l'
            intervalle [%d,%d] :\n",min,max);
                scanf("%d",&val);
        }
  }
  return val;
}

```

```c
/* retourne un entier positif saisie entre deux bornes */
int saisie_entier_borne_abs(int min, int max)
{
  int val =  saisie_entier_borne(min,max);
  if(val < 0) val = -val;
  return val;
}

/* retourne un entier saisie par la fonction épointe par pf */
int saisie_entier_gen(int min, int max, pf_saisie pf)
{       return (*pf) (min,max);}

/* retourne un pointeur sur une fonction de saisie */
pf_saisie choix_saisie_entier(char c)
{       if(c != 'a')
                return &saisie_entier_borne;
        else
                return &saisie_entier_borne_abs;
}

/* Affiche un entier */
void affiche_entier(int val)
{       fprintf(stdout,"\n%d\n",val);}

/* Affiche un entier avec fioritures */
void affiche_entier_fioriture(int val)
{       fprintf(stdout,"\nOh ! Le bel entier : %d.\n",val);}

/* Affiche un entier avec la fonction épointe par pf */
void affiche_entier_gen(int val, void (*pf) (int))
{       (*pf) (val);}

/* Affiche une adresse en émmoire */
void affiche_adresse(void *p)
{       fprintf(stdout,"\nJe pointe ici : %p\n",p);}


int main(){

        pf_saisie pf_get_int = NULL;            // use typedef type
        void (*pf_disp_int) (int) = NULL;
        int val = 0;

        affiche_adresse((void *)pf_get_int);

        pf_get_int =  choix_saisie_entier('b');
        affiche_adresse((void *)pf_get_int);
        val = saisie_entier_gen(0,100,pf_get_int);

        val = saisie_entier_gen(-100,100,&saisie_entier_borne_abs);

        pf_disp_int = &affiche_entier;
        affiche_adresse((void *)pf_disp_int);
        affiche_entier_gen(val,pf_disp_int);

        affiche_entier_gen(val,&affiche_entier_fioriture);

        return 0;
}
```

Listing 5 – ex5_pf.c

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

#define TAILLE 10

typedef void* (*pf) (void* arg);

/* la struct éutilise pour les entiers */
typedef struct
{       int* tab;
        int taille;
} une_table;

/* retourne un entier saisie entre deux bornes */
void* saisie_entier_borne(void* arg)
{
        int val, min, max;

        /* Cast de l'argument d'éentre pour éércuprer les arguments */
        int* bornes = (int*)arg;
        min = bornes[0];
        max = bornes[1];
        /* Fin cast */

        val = min -1;

  if(min > max)
  {    val = max;  }
  else {
                while((val < min) || (val > max))
                {
                        fprintf(stdout,"\nSaisir␣un␣entier␣dans␣l'intervalle␣[%d,%d]␣:\t",min,max);
                        scanf("%d",&val);
                }
  }

  /* Cast de la  valeur de retour (possible car val est un entier) */
  return (void*)val;
}

void* saisie_tab_entier(void *arg)
{
  /* cast du void* a une_table* */
  une_table* T1 = (une_table*) arg;
  int i;

  printf("\n");
  for(i=0;i<T1->taille;i++)
  {
    printf("Veuillez␣saisir␣un␣nombre␣entier␣:\t");
    scanf("%d",(T1->tab)+i);
  }

  return NULL;
}

/* retourne un entier positif saisie entre deux bornes */
void* saisie_entier_borne_abs(void* arg)
{
```

```c
  /* Cast du void * éretourne en int */
  int val =  (int) saisie_entier_borne(arg);

  if(val < 0)   val = -val;

  return (void*)val;
}

/* retourne un double */
void* saisie_reel(void* arg){

        double* n = (double *) arg;;

        printf("\nVeuillez saisir un érel :\t");
        scanf("%lf",n);

  return NULL;
}

/* retourne qqch saisie par la fonction épointe par pf */
void* saisie_gen(void* (*pf) (void* arg), void* arg){
        void* res = (*pf)(arg);
        return res;
}

/* Affiche un entier */
void affiche_entier(int val)
{
        fprintf(stdout,"\n%d\n",val);
}

/* Affiche un entier avec fioritures */
void affiche_entier_fioriture(int val)
{
        fprintf(stdout,"\nOh ! Le bel entier : %d.\n",val);
}

/* Affiche un entier avec la fonction épointe par pf */
void affiche_entier_gen(int val, void (*pf) (int))
{
        (*pf) (val);
}

int main()
{
        pf pf_get = NULL;
        void (*pf_disp_int) (int) = NULL;
        int i;

        double reel = 0.0;
        int val = 0;

        une_table T;
        T.tab = (int *) malloc(TAILLE * sizeof(int));
        T.taille = TAILLE;

        int bornes[2] = {0,100};

        pf_get = &saisie_entier_borne;
        val = (int) saisie_gen(pf_get, (void*) bornes);

        affiche_entier_gen(val,&affiche_entier_fioriture);
```

```
122
123         pf_get = &saisie_tab_entier;
124         saisie_gen(pf_get, (void*) &T);
125
126         for(i=0;i<T.taille;i++)
127         {               printf("\nT[%d] = \t%d\n",i,T.tab[i]);   }
128
129         pf_get = &saisie_reel;
130         saisie_gen(pf_get, (void*) &reel);
131
132         printf("\nLe reel vaut \t%lf\n",reel);
133
134         return 0;
135 }
```

Listing 6 – ex1_t.c

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <pthread.h>
4
5  int i; //global integer
6
7  typedef struct
8  {
9          int inc;
10         int* ret;
11 } arg_str;
12
13 void* addition(void* arg)
14 {
15         /* On èèrcupre les arguments */
16         arg_str* inout = (arg_str*) arg; //cast
17         int inc = inout->inc;
18         int* ret = inout->ret;
19         /* fin recup */
20
21         i = i + inc;
22         printf("Hello, ici thread fils PID %d. Pour info, i = \t%d\
               n",getpid(),i);
23         i = i + 2*inc;
24         printf("Hello, ici thread fils PID %d. Pour info, i = \t%d\
               n",getpid(),i);
25
26         *ret = i;
27
28         pthread_exit((void*) ret);
29 }
30
31 int main()
32 {
33
34         pthread_t num_thread;
35         arg_str my_arg;
36         int val,val2;
37         int* retour;
38
39         my_arg.inc = 10;
40         my_arg.ret = &val;
41         i = 0;
42
43         if(pthread_create(&num_thread, NULL, &addition, (void *) &
```

```
            my_arg) != 0)
44              perror("Pb. pthread_create()\n");
45
46          i += 1000;
47          printf("Hello, ici thread père PID %d. Pour info, i = \t%d\
                n",getpid(),i);
48          i += 2000;
49          printf("Hello, ici thread père PID %d. Pour info, i = \t%d\
                n",getpid(),i);
50
51          pthread_join(num_thread,(void *) &retour);
52          val2 = *retour;
53
54          if(val == val2)
55          {
56                  printf("Ca tombe bien !\n");
57          }
58
59          return 0;
60  }
```

Listing 7 – test_thread.c

```
1   #include <pthread.h>
2   #include <stdio.h>
3   #include <time.h>
4   #include <stdlib.h>
5   #include "basic_func.h"
6
7   int glob;
8   pthread_mutex_t my_mutex = PTHREAD_MUTEX_INITIALIZER;
9
10  int main(int argc, char *argv[]) {
11
12    long result;
13    pthread_t pthread_id[2];
14    incStruct toto = { 0, 0, 0, NULL};
15    time_t T[3];
16
17    if(argc != 4){
18      printf("usage : ./test_thread val_init increment
            nombre_iterations\n");
19      return(1);
20    }
21
22    glob = 0;
23
24    toto.val_init = atoi(argv[1]);
25    toto.inc = atoi(argv[2]);
26    toto.N = atoi(argv[3]);
27    toto.res = &result;
28
29    result = toto.val_init;
30
31    T[1] = time(T);
32    pthread_create(&pthread_id[0], NULL, &somme_inc_Ntimes,(void *) &
          toto);
33    pthread_create(&pthread_id[1], NULL, &somme_inc_Ntimes,(void *) &
          toto);
```

```
34
35    glob +=4;
36
37    pthread_join(pthread_id[0],NULL);
38    pthread_join(pthread_id[1],NULL);
39    T[2] = time(T);
40
41    printf("Result : %ld\t Temps en s. : %ld\t Valeur glob : %d\n",
          result,T[2]-T[1],glob);
42
43    return(0);
44
45 }
```

Listing 8 – basic_func.h

```
1  #ifndef BASIC_FUNC
2  #define BASIC_FUNC
3  #include <pthread.h>
4
5  extern int glob;
6  extern pthread_mutex_t my_mutex;
7
8  typedef struct  {
9          long val_init;
10         long inc;
11         unsigned long N;
12         long *res;
13 } incStruct;
14
15 void *somme_inc_Ntimes(void *arg);
16
17 #endif
```

Listing 9 – basic_func.c

```
1  #include <stdio.h>
2  #include <unistd.h>
3  #include "basic_func.h"
4
5  void *somme_inc_Ntimes(void *arg) {
6
7    incStruct *var =(incStruct *) arg;
8    int i = 0;
9
10   pthread_mutex_lock(&my_mutex);
11
12   for(i = 0; i < var->N; i++) {
13                 sleep(0.5);
14                 *(var->res) += var->inc;
15   }
16   pthread_mutex_unlock(&my_mutex);
17
18   glob += 5;
19
20   pthread_exit(NULL);
21 }
```