Solution exercice 2.1

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

#define PRINT_ADDRESS(ADR) printf("0x%" PRIxPTR "\n", (intptr_t) (ADR))

int main(void) {

   int n = 1;
   int* ptr = &n;

   printf("%d\n", *ptr);
   PRINT_ADDRESS(ptr); // PRINT_ADDRESS(&n);
   PRINT_ADDRESS(&ptr);

   // Autre variante (avec %p)
   printf("%d\n", *ptr);
   printf("%p\n", (void*) ptr); // ou printf("%p\n", ptr);
                                // ou printf("%p\n", &n);
   printf("%p\n", (void*)&ptr); // ou printf("%p\n", &ptr);

   return EXIT_SUCCESS;
}

// 1
// 0x22fe4c (si 32 ou 64 bits)
// 0x22fe40 (si 32 ou 64 bits)
// 1
// 000000000022FE4C (si 64 bits), 0022FE4C (si 32 bits)
// 000000000022FE40 (si 64 bits), 0022FE40 (si 32 bits)
```

Solution exercice 2.2

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

void carre_et_cube(double x, double* carre, double* cube);
void test(double x);

int main(void) {
   test(-1);
   test(0);
   test(1);
   test(2);
   test(2.5);
   test(DBL_MAX);

   return EXIT_SUCCESS;
}

void carre_et_cube(double x, double* carre, double* cube) {
   *carre = x * x;
   *cube = *carre * x;
}

void test(double x) {
   double carre, cube;
   carre_et_cube(x, &carre, &cube);
   printf("carre(%g) = %g, cube(%g) = %g\n", x, carre, x, cube);
}

// carre(-1) = 1, cube(-1) = -1
// carre(0) = 0, cube(0) = 0
// carre(1) = 1, cube(1) = 1
// carre(2) = 4, cube(2) = 8
// carre(2.5) = 6.25, cube(2.5) = 15.625
// carre(1.79769e+308) = inf, cube(1.79769e+308) = inf
```