```
// Program to convert text file containing doubles to binary.

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

#define ERROR 1
#define ENOTOPEN 2

typedef struct
{
    unsigned long count;
    double *values;
} doubles;

char data_ext[] = ".data";
char usage_info[] =
    "<filename> is the name of the input file, a plain text file with double-\n"
    "precision floating-point values.\n"
    "[max] is the maximum number of values to put in the output binary file.\n";

char *strip_extension(char *filename)
{
    for (int i = strlen(filename); i-- >= 0;) {
        if (filename[i] == '.') {
            filename[i] = '\0';
            break;
        }
    }
    return filename;
}

int usage(char *prog)
{
    printf("Usage: %s <filename> [max]\n\n%s\n\n", prog, usage_info);
    return ERROR;
}

unsigned long count_doubles(FILE *INPUT)
{
    unsigned long count = 0;
    rewind(INPUT);
    double d;
    while (1) {
        fscanf(INPUT, "%lf", &d);
        if (feof(INPUT))
            break;
        count++;
    }
    printf("%ld doubles in file.\n", count);
    return count;
}

doubles read_doubles(FILE *INPUT)
{
    doubles retval = {count_doubles(INPUT), 0};
    rewind(INPUT);
    retval.values = (double*)malloc(retval.count * sizeof(double));
    for (int i = 0; i < (int)retval.count; i++) {
        if (1 != fscanf(INPUT, "%lf", retval.values+i)) {
            perror("Failed to read double: ");
        }
    }
    return retval;
}

void write_doubles(FILE *OUTPUT, doubles ds)
{
```

Q: use a project header

BS: needless preprocessor directives

NAME: start types with a capital

C style struct definition

ROC

SF global public data. Use, e.g., an anonymous namespace

WIM?

MFIF

C: use C++'s string

SLV

Type: negative lengths?

TC: use C++'s

NAMING: use lc filesystem members

C: use streams

LEAK

C

SLV

PP

NMN: use a bool value for clarity

see above

C NAMING

C casts

C! return value not checked.

Use C++'s new

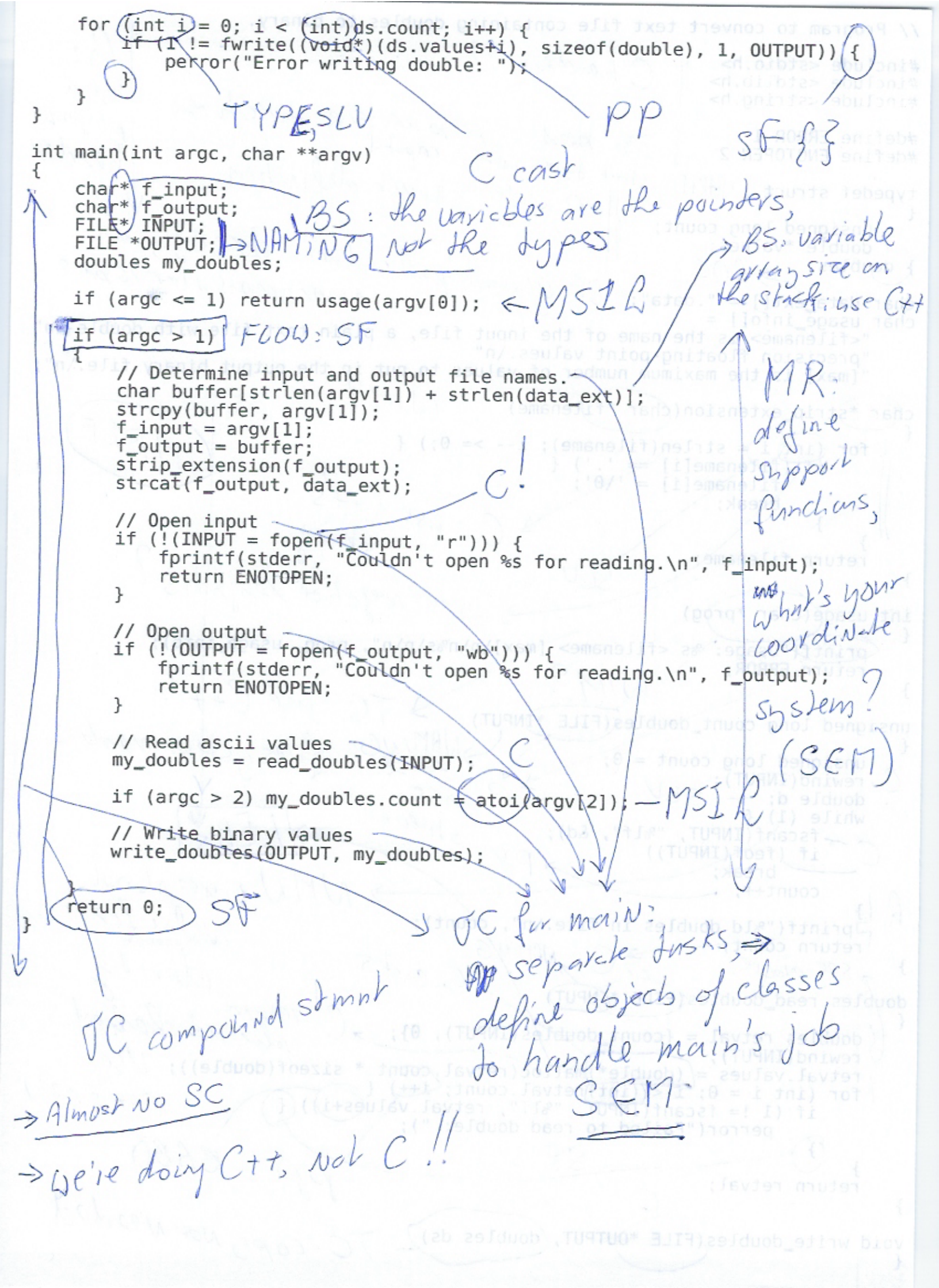
BABO

SF

PP

C

TC: copy not needed


for (int i = 0; i < (int)ds.count; i++) {
if (1 != fwrite((void*)(ds.values+i), sizeof(double), 1, OUTPUT)) {
perror("Error writing double: ");
}
}
}
TYPESLV
PP
SF {}
C cast
int main(int argc, char **argv)
{
char* f_input;
char* f_output;
FILE* INPUT;
FILE *OUTPUT;
doubles my_doubles;
BS: the varicbles are the pointers,
not the types
→NAMING
←MSIC
BS: variable
array size on
the stack: use C++
if (argc <= 1) return usage(argv[0]);
if (argc > 1)
FLOW: SF
{
// Determine input and output file names.
char buffer[strlen(argv[1]) + strlen(data_ext)];
strcpy(buffer, argv[1]);
f_input = argv[1];
f_output = buffer;
strip_extension(f_output);
strcat(f_output, data_ext);
C!
// Open input
if (!(INPUT = fopen(f_input, "r"))) {
fprintf(stderr, "Couldn't open %s for reading.\n", f_input);
return ENOTOPEN;
}
// Open output
if (!(OUTPUT = fopen(f_output, "wb"))) {
fprintf(stderr, "Couldn't open %s for reading.\n", f_output);
return ENOTOPEN;
}
// Read ascii values
my_doubles = read_doubles(INPUT);
C
if (argc > 2) my_doubles.count = atoi(argv[2]);
MSIC
// Write binary values
write_doubles(OUTPUT, my_doubles);
}
return 0;
}
SF
MR:
define
support
fundions,
what's your
coordinate
system?
(SEM)
OC for main:
separate tasks,
define objects of classes
to handle main's job
SEM
OC compound stmnt
→ Almost no SC
→ We're doing C++, not C !!