In this document, you will find a few C program listings rendered with the color scheme used in the Visual Studio Community 2019 IDE. Well, more or less. Not every coloring is possible. Note that C++ is not covered.

For example, local variables, parameters and arguments are not colorized (i.e. black) because `listings` does no compiler parsing and does not know when to use coloring. (We could have used identifier styling, but that would affect all of the above.) Function names are partly colorized, only for most functions from the standard library. You can add your own function names using:

```
keywords=[10]{your_function_names_as_a_comma_separated_list}
```

(Really, we should do this by defining a simple control sequence and expand it in the listing setup.) Also, typedef'd types are not colorized except for the common ones in `stdint.h` and some frequently used typedef such as `FILE`, `size_t`, `time_t` and `clock_t`. You can add your own typedefs using:

```
keywords=[11]{your_typedefs_as_a_comma_separated_list}
```

Macro names are partly supported for some known macro such as `NULL` and `_MSC_VER`. You can add your own macro names using:

```
keywords=[12]{your_macros_as_a_comma_separated_list}
```

If you want to add extra system header files, use:

```
keywords=[13]{your_extra_system_header_files_as_a_comma_separated_list}
```

Escape sequences such as `\n` are not colorized, but should be. We have to find out if we can parse string contents.

The string of characters after a preprocessor pragma are not colorized.

The < and > in header file inclusion are not colorized. Then we have to use more delimiters but that messes up tests as in: `if (a < b)`.

The quote for characters and double quotes for strings are colorized the same as the string.

Line breaks in listings are shown by the ↪ symbol on the continuing line.

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

int main(void) {
    double x, y, wortel, macht;
    int i;
    printf("Voer een positief getal x in: ");
    scanf("%lf", &x);
    printf("Voer een getal y in: ");
    scanf("%lf", &y);
    for (i = 0; i < 5; i = i + 1) {
        wortel = sqrt(x);
        macht = pow(x, y);
        printf("De vierkantswortel uit %f is: %f\n", x, wortel);
        printf("%f tot de macht %f is: %f\n", x, y, macht);
        x = x + 1;
    }
    getchar();
    return 0;
}
```

```c
#include <stdio.h>

int main(void){
    int cijfer;
    char letter;
    do {
        printf("Geef je cijfer: ");
        scanf("%d", &cijfer);
    } while (cijfer < 0 || cijfer > 10);
    switch (cijfer) {
        case 10:
        case 9:
        case 8:
            letter = 'A'; break;
        case 7:
            letter = 'B'; break;
        case 6:
            letter = 'C'; break;
        case 5:
            letter = 'D'; break;
        default:
            letter = 'F'; break;
    }
    printf("Dit komt overeen met een %c.\n", letter);
    getchar();
    return 0;
}
```

```c
#include <stdio.h>

int fib(int n) {
    if (n < 2) {
        return n;
    }
    else {
      return fib(n-1) + fib(n-2);
    }
}

int main(void) {
    int getal;

    do {
        printf("Geef een getal groter dan of gelijk aan 0: ");
        scanf("%d", &getal);
    } while (getal < 0);

    printf("fib(%d) is %d\n", getal, fib(getal));

    printf("Druk op enter om het programma af te sluiten.");
    getchar();
    return 0;
}
```

```c
#include <stdio.h>

/* Dit programma demonstreert hoe een
   groot getal cijfer voor cijfer kan
   worden ingelezen */

int leesCijfer(void) {
    switch (getchar()) {
        case '0': return 0;
        case '1': return 1;
        case '2': return 2;
        case '3': return 3;
        case '4': return 4;
        case '5': return 5;
        case '6': return 6;
        case '7': return 7;
        case '8': return 8;
        case '9': return 9;
        default: return -1;
    }
}

int main(void) {
    int cijfer;
    printf("Type een groot getal:\n");
    do {
        cijfer = leesCijfer();
        if (cijfer != -1) {
            printf("%d ", cijfer);
        }
    } while (cijfer != -1);

    printf("Druk op enter om het programma af te sluiten.");
    getchar();
    return 0;
}
```

```c
#include <stdio.h>

#define AANTAL 10

int main(void) {
    double temperatuur[AANTAL];
    int i;
    double temp_acc;

    for (i = 0; i < AANTAL; i = i + 1) {
        do {
            printf("Geef temperatuur %d op: ", i + 1);
            fflush(stdin); // Oofff....
        } while (scanf("%lf", &temperatuur[i]) != 1);
    }

    temp_acc = 0.0;
    for (i = 0; i < AANTAL; i = i + 1) {
        temp_acc = temp_acc + temperatuur[i];
    }

    printf("De gemiddelde temperatuur is %.2f\n\n", temp_acc / AANTAL);

    printf("Druk op enter om het programma af te sluiten.");
    getchar();
    return 0;
}
```

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

int main(void) {

    char *ptr = malloc(1024);  /* Allocate 1024 bytes */

    if (ptr == NULL) {
        printf("Oops, no memory!\n");
        exit(-2);
    }

    printf("Allocated 1024 bytes at address %p\n", ptr);

    free(ptr); /* Free memory */

    printf("Druk op enter om het programma af te sluiten.");
    getchar();
    return 0;
}
```

```c
    char *ptrnew;

    ptrnew = realloc(ptr, 2048);
```

```c
/* myprog.c */
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {

    int i;

    printf("\nAantal argumenten: %d\n\n", argc);
    for (i = 0; i<argc; i++) {
        printf("Argument %d: %s\n", i, argv[i]);
    }

    printf("Druk op enter om het programma af te sluiten.");
    getchar();
    return 0;
}
```

```
struct artikel {
    int nummer;       // artikelnummer
    char naam[20];    // artikelnaam
    int aantal;       // aantal beschikbaar
    double prijs;     // prijs per stuk
};

enum myenum {A, B, C};

typedef enum {A, B, C} myenum_t;
```

This listing is typeset with (note the extra pair of braces):

```
\begin{lstlisting}[numbers=right,{morekeywords=[11]{artikel,myenum,myenum_t}}]
struct artikel {
int nummer;       // artikelnummer
char naam[20];    // artikelnaam
int aantal;       // aantal beschikbaar
double prijs;     // prijs per stuk
};

enum myenum {A, B, C};

typedef enum {A, B, C} myenum_t;
\end{lstlisting}
```

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

#pragma warning(disable : 4996)

int main(int argc, char *argv[]) {

    /* The input and output file handlers */
    FILE *infile, *outfile;

    /* Character to read and character count */
    int ch, count = 0;

    /* We need exactly 3 arguments */
    if (argc != 3) {
        printf("Usage: %s infile outfile\n", argv[0]);
        return -1;
    }

    /* Test if input file and output file have the same name */
    if (strcmp(argv[1], argv[2]) == 0) {
        printf("Filenames cannot be the same\n");
        return -2;
    }

    /* Open the input file, exit if error */
    infile = fopen(argv[1], "rb");
    if (infile == NULL) {
        printf("Cannot open input file %s\n", argv[1]);
        return -3;
    }

    /* Open the output file, exit if error */
    outfile = fopen(argv[2], "wb");
    if (outfile == NULL) {
        printf("Cannot open output file %s\n", argv[2]);
        fclose(infile);
        return -4;
    }

    /* Copy characters from input file to output file */
    while ((ch = fgetc(infile)) != EOF) {
        count++;
        fprintf(outfile, "%c", ch);
    }

    printf("Copied %d characters\n", count);

```

```
50      fclose(infile);
51      fclose(outfile);
52
53      return 0;
54  }
```