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

uint32_t rotate(uint32_t x, int n) {
    return (x << n) | (x >> (32 - n));
}

void step(uint32_t *s, int i, int j, int k, int r) {
    s[i] ^= rotate(s[j] + s[k], r);
}

void quarterround(uint32_t *s, int i0, int i1, int i2, int i3) {
  step(s, i1, i0, i3, 7);
  step(s, i2, i1, i0, 9);
  step(s, i3, i2, i1, 13);
  step(s, i0, i3, i2, 18);
}

void rowround(uint32_t *s) {
    quarterround(s, 0, 1, 2, 3);
    quarterround(s, 5, 6, 7, 4);
    quarterround(s, 10, 11, 8, 9);
    quarterround(s, 15, 12, 13, 14);
}

void columnround(uint32_t *s) {
    quarterround(s, 0, 4, 8, 12);
    quarterround(s, 5, 9, 13, 1);
    quarterround(s, 10, 14, 2, 6);
    quarterround(s, 15, 3, 7, 11);
}

void doubleround(uint32_t *s) {
    columnround(s);
    rowround(s);
}

void rounds(uint32_t *s, int nrounds) {
    uint32_t s1[16];
    int i;

    /* copiar s para s1 */
    for(i = 0; i < 16; i++)
        s1[i] = s[i];

    while(nrounds >= 2) {
        doubleround(s1);
        nrounds -= 2;
    }

    for(i = 0; i < 16; i++)
        s[i] += s1[i];
}

const uint32_t o[4] = {
    0x61707865,   /* expa */
    0x3320646e,   /* nd 3 */
```

```
    0x79622d32,   /* 2-by */
    0x6b206574,   /* te k */
};

void block(uint32_t *s, uint32_t *pos, uint32_t *nonce, uint32_t *key) {
    int i;

    /* s[::5] = o */
    s[ 0] = o[0];
    s[ 5] = o[1];
    s[10] = o[2];
    s[15] = o[3];
    /* s[1:5] = key[:4] */
    s[1] = key[0];
    s[2] = key[1];
    s[3] = key[2];
    s[4] = key[3];
    /* s[6:10] = nonce ++ pos  */
    s[6] = nonce[0];
    s[7] = nonce[1];
    s[8] = pos[0];
    s[9] = pos[1];
    /* s[11:15] = key[4:] */
    s[11] = key[4];
    s[12] = key[5];
    s[13] = key[6];
    s[14] = key[7];

    rounds(s, 20);  /* Salsa20/20 */
}

int main() {
    int i;
    uint32_t s[16];
    /* Exemplo de "The Salsa20 family of stream ciphers", Daniel J. Bernstein
     * http://cr.yp.to/snuffle/salsafamily-20071225.pdf */
    uint32_t key[8] = {
        0x04030201,
        0x08070605,
        0x0c0b0a09,
        0x100f0e0d,
        0x14131211,
        0x18171615,
        0x1c1b1a19,
        0x201f1e1d,
    };
    uint32_t nonce[] = {
        0x01040103,
        0x06020905,
    };
    uint32_t pos[] = {
        0x7,
        0x0,
    };
    block(s, pos, nonce, key);

    for(i = 0; i < 16; i++)
```

```
        printf("0x%08x\n", s[i]);
    return 0;
}
```