```c
/*@author Michal Pawlowski*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <stdbool.h>

#define BUF_SIZE 10

typedef struct RGB {
    unsigned int R;
    unsigned int G;
    unsigned int B;
}RGB;

typedef struct element {
    struct element* pPrev;
    struct element* pNext;
    struct RGB color;
}element;

/** Funkcja generuje wartości RGB do momentu zapełnienia buforu
@date 2020-04-02
@param buf wskaźnik na bufor
@param bufferLength aktualna liczba elementów bufora
@param writeIndex aktualna pozycja zapisu
*/
void makeRGB(RGB* buf, int* bufferLength, int* writeIndex) {

    srand(time(NULL));

    while (true) {

        if ((*bufferLength) == BUF_SIZE) break;

        buf[*writeIndex].R = rand() % 255;
        buf[*writeIndex].G = rand() % 255;
        buf[*writeIndex].B = rand() % 255;
        (*bufferLength)++;
        (*writeIndex)++;

        if ((*writeIndex) == BUF_SIZE) (*writeIndex) = 0;
    }
}

/** Funkcja odczytuje i kopiuje wartości z bufora do tablicy
@date 2020-04-02
@param buf wskaźnik na bufor
@param tab tablica do której zostaną wczytane wartości
@param bufferLength aktualna liczba elementów bufora
@param writeIndex aktualna pozycja odczytu
*/
void readRGB(RGB* buf, RGB* tab, int* bufferLength, int* readIndex) {
```

```
54
55      while (true) {
56
57          if ((*bufferLength) == 0) break;
58
59          tab[*readIndex] = buf[*readIndex];
60
61          (*bufferLength)--;
62          (*readIndex)++;
63
64          if ((*readIndex) == BUF_SIZE) (*readIndex) = 0;
65      }
66  }
67
68  /** Funkcja generuje wartości RGB do bufufora cyklicznego a następnie
      przepisuje wartości do tablicy
69  @date 2020-04-02
70  @param tab tablica docelowa
71  */
72  void circularBufferToTable(RGB tab[]) {
73
74      struct RGB* circularBuffer = malloc(sizeof(RGB) * BUF_SIZE);
75      int bufferLength = 0;
76      int readIndex = 0;
77      int writeIndex = 0;
78
79      makeRGB(circularBuffer, &bufferLength, &writeIndex);
80      readRGB(circularBuffer, tab, &bufferLength, &readIndex);
81
82      free(circularBuffer);
83  }
84
85  /** Funkcja dodaje element do listy niepustej
86  @date 2020-04-02
87  @param pHead wskaźnik na początek listy
88  @param color struktura zawierająca wartości RGB
89  */
90  void addElem(element** pHead, RGB color) {
91
92      element* pHelp = *pHead;
93
94      while (pHelp->pNext) pHelp = pHelp->pNext;
95
96      pHelp->pNext = (element*)malloc(sizeof(element));
97      pHelp->pNext->color = color;
98      pHelp->pNext->pNext = NULL;
99      pHelp->pNext->pPrev = pHelp;
100 }
101
102 /** Funkcja zlicza ilość ellementów o większej zadanje wartości R
103 @date 2020-04-02
104 @param pHead wskaźnik na początek listy
105 @param value zadana wartosć
```

```
106 */
107 unsigned int countR(element* pHead, unsigned int value) {
108
109     unsigned int number = 0;
110
111     do {
112         if (pHead->color.R > value) number++;
113         pHead = pHead->pNext;
114     } while (pHead);
115     return number;
116 }
117
118 /** Funkcja usuwa liste
119 @date 2020-04-02
120 @param pHead wskaźnik na początek lsty
121 */
122 void deleteList(element** pHead) {
123
124     if (*pHead) {
125         deleteList((*pHead)->pNext);
126         (*pHead)->pPrev->pNext = NULL;
127         free(*pHead);
128         *pHead = NULL;
129     }
130 }
131
132 /** Funkcja tworzy listę dwukierunkową z tablicy kolorów a następnie zlicza
      wartości większe od zadanej składowej
133 @date 2020-04-02
134 @param tab wskaźnik na tablice z danymi
135 */
136 void makeListAndCount(RGB* tab) {
137
138     element* pHead = NULL;
139     pHead = (element*)malloc(sizeof(element));
140     pHead->color = tab[0];
141     pHead->pNext = NULL;
142     pHead->pPrev = NULL;
143
144     for (int i = 1; i < BUF_SIZE; i++) {
145         addElem(&pHead, tab[i]);
146     }
147
148     printf("\nIlosc elementow o skladowej R wiekszej niz %d: %d", 128,
        countR(pHead, 128));
149
150     deleteList(&pHead);
151 }
152
153 /** Funkcja tworzy listę dwukierunkową z tablicy kolorów a następnie zlicza
      wartości większe od zadanej składowej, wersja z arytmetyką wskaźników
154 @date 2020-04-02
155 @param tab wskaźnik na tablice z danymi
```

```
156 */
157 void makeListAndCount2(RGB* tab) {
158
159     RGB* ptr = tab;
160     element* pHead = NULL;
161     pHead = (element*)malloc(sizeof(element));
162     pHead->color = *(ptr++);
163     pHead->pNext = NULL;
164     pHead->pPrev = NULL;
165
166     for (int i = 1; i < BUF_SIZE; i++) {
167         addElem(&pHead, *ptr++);
168     }
169
170     printf("\nIlosc elementow o skladowej R wiekszej niz %d: %d", 128,
      countR(pHead, 128));
171 }
172
173 int main(int argc, char* argv[]) {
174
175     RGB tab[BUF_SIZE];
176
177     circularBufferToTable(tab);
178
179     for (int i = 0; i < BUF_SIZE; i++) {
180         printf("%d,%d,%d\n", tab[i].R, tab[i].G, tab[i].B);
181     }
182
183     makeListAndCount2(tab);
184 }
185
```