BÁO CÁO THU HOẠCH

BÀI TẬP BITMAP
BÀI TẬP LÝ THUYẾT 1

Thực hiện: Nguyễn Minh Trí
20120219
20120219@student.hcmus.edu.vn

I. Tổng quan

Trong đồ họa máy vi tính, BMP, còn được biết đến với tên tiếng Anh khác là Windows bitmap, là một định dạng tập tin hình ảnh khá phổ biến. Với sự hướng dẫn của thầy **Phạm Minh Hoàng**, em xin báo cáo bài tập lý thuyết 1 – bài tập về các thao tác xử lý trên file bitmap.

Chương trình có các tính năng: **giao diện và thông báo lỗi cơ bản**; **đọc / lưu ảnh bmp theo đường dẫn cho trước**; **chuyển đổi file 24 / 32 bpp sang ảnh 8 bpp theo yêu cầu đề bài**; **thu nhỏ ảnh màu 24 / 32 bpp, ảnh 8 bpp theo tỉ lệ S cho trước**.

Chương trình được chia theo cấu trúc 3 file, bao gồm: **20120219.cpp**, **Source.cpp** và **Header.h**. Trong đó file 20120219.cpp chứa các hàm nhập xuất, xử lý các thao tác cơ bản của người dùng.

```
1   #include "Header.h"
2
3   using namespace std;
4
5   //Luu tru duong dan file
6   char filename_in[512], filename_out[512];
7
8   //Ham gioi thieu chuong trinh, thanh vien
9  ⊞void welcome() { ... }
17
18  //Yeu cau nhap dia chi file doc, xuat
19 ⊞void InputFileIn() { ... }
23
24 ⊞void InputFileOut() { ... }
28
29  //Ham xu ly convert anh 24/32 BPP sang 8 BPP
30 ⊞void convertBMPto8(ifstream &fi, ofstream &fo) { ... }
69
70  //Ham zoom anh BMP theo yeu cau
71 ⊞void zoomBMP(ifstream& fi, ofstream& fo, int S) { ... }
121
122 //Main program
123⊞int main(int argc, char* argv[]) { ... }
235
```

Hình 1. Cấu trúc hàm của file 20120219.cpp

File Source.cpp chứa các hàm đọc theo các thành phần của BMP, xử lý, tính toán, xử lý và ghi vào file BMP mới để xuất file theo đường dẫn nhập trước bởi người dùng.

Hình 2. Cấu trúc hàm của file Source.cpp

File Header.h khai báo các thư viện cần thiết để chạy chương trìn, chứa các struct để lưu trữ các thuộc tính trong file BMP và khai báo các hàm trong Source.cpp.

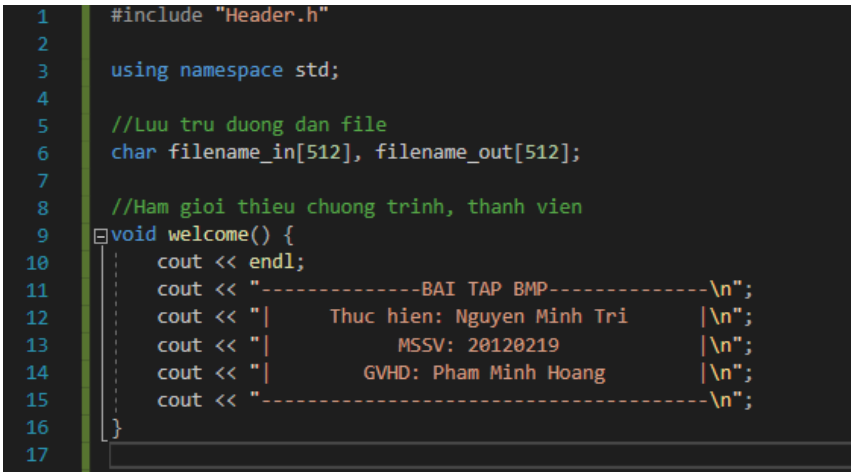
```
//Tap lenh tranh include nhieu lan
#pragma once

/*Tap lenh sap xep cac o nho trong struct
gan nhau de tinh sizeof() chinh xac hon*/
#pragma pack(1)

//Khai bao cac thu vien can thiet
#include <iostream>
#include <fstream>
#include <stdio.h>
#include <conio.h>
#include <math.h>
#include <stdlib.h>

using namespace std;

/*Struct chua cac thong tin theo tung
thanh phan cua file BMP*/
struct BmpHeader { ... };
struct BmpDib { ... };

struct Colormap8 { ... };

struct Color24 { ... };
struct Color32 { ... };

struct Bmpfile8 { ... };
struct Bmpfile24 { ... };
struct Bmpfile32 { ... };

//Ham doc cac thanh phan cua file BMP
void readBmpHeader(ifstream& f, BmpHeader& inp);
void readDib(ifstream& f, BmpDib& inp);

void readColorTable8(ifstream& f, Bmpfile8& inp);

void readPixel8(ifstream& f, Bmpfile8& inp);
void readPixel24(ifstream& f, Bmpfile24& inp);
void readPixel32(ifstream& f, Bmpfile32& inp);

//Ham xu ly file BMP doc duoc va chuyen sang 8 BPP
void Convert24_8(Bmpfile24& inp, Bmpfile8& out);
void Convert32_8(Bmpfile32& inp, Bmpfile8& out);

//Ham xu ly file BMP doc duoc va thu nho theo S
void resizeBmp8(Bmpfile8 inp, Bmpfile8& out, int S);
void resizeBmp24(Bmpfile24 inp, Bmpfile24& out, int S);
void resizeBmp32(Bmpfile32 inp, Bmpfile32& out, int S);

//Ham luu thong tin anh sau khi duoc xu ly
void writeBmp8(ofstream& f, Bmpfile8& inp);
void writeBmp24(ofstream& f, Bmpfile24& inp);
void writeBmp32(ofstream& f, Bmpfile32& inp);

//Ham giai phong tranh ro ri bo nho
void releaseBmpPixelArray8(uint8_t** pixelarray, int rowCount);
void releaseBmpPixelArray24(Color24** pixelarray, int rowCount);
void releaseBmpPixelArray32(Color32** pixelarray, int rowCount);
```

Hình 3. Cấu trúc cơ bản file Header.h

II. Giải thích chi tiết cấu trúc các file

1. File 20120219.cpp

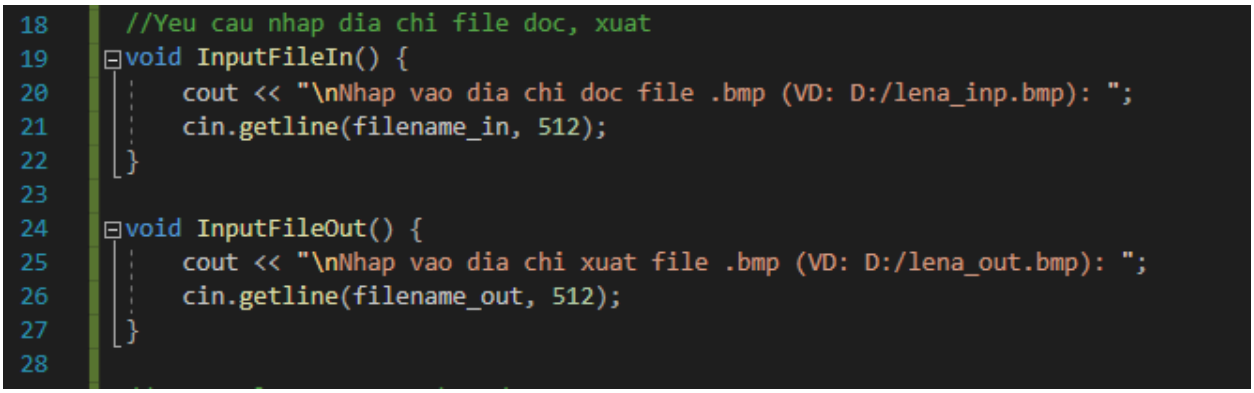
Ở đầu file, include “Header.h” để load các hàm và thư viên cần thiết. Chứa hàm welcome() để thông báo các thông tin cơ bản cho người sử dụng. Khai báo char filename_in và filename_out để lưu trữ địa chỉ file đọc, lưu mà người dùng nhập khi sử dụng console.



```
#include "Header.h"

using namespace std;

//Luu tru duong dan file
char filename_in[512], filename_out[512];

//Ham gioi thieu chuong trinh, thanh vien
void welcome() {
    cout << endl;
    cout << "--------------BAI TAP BMP--------------\n";
    cout << "|     Thuc hien: Nguyen Minh Tri      |\n";
    cout << "|           MSSV: 20120219            |\n";
    cout << "|        GVHD: Pham Minh Hoang        |\n";
    cout << "---------------------------------------\n";
}

```

Hình 4. Khai báo, xuất các thông tin cơ bản.

Các hàm dùng để thông báo người dùng nhập đường dẫn file nhập, xuất.



```
//Yeu cau nhap dia chi file doc, xuat
void InputFileIn() {
    cout << "\nNhap vao dia chi doc file .bmp (VD: D:/lena_inp.bmp): ";
    cin.getline(filename_in, 512);
}

void InputFileOut() {
    cout << "\nNhap vao dia chi xuat file .bmp (VD: D:/lena_out.bmp): ";
    cin.getline(filename_out, 512);
}

```

Hình 5.

Hàm **convertBMPto8()** dùng để thực hiện yêu cầu chuyển file BMP 24/32 BPP sang 8 BPP. Nhận 2 tham số stream fi, fo để thực hiện đọc, lưu file BMP sau khi xử lý. Ý tưởng tiếp cận của hàm là sau khi đọc phần dib thông qua hàm **readDib()**, sẽ biết được thông tin pixelSize (BPP). Vì file 8 / 24 / 32 BPP mang các thuộc tính khác nhau, sẽ được xử lý riêng trong từng trường hợp file. Sau khi đã kiểm tra được BPP, khai báo file đích **Bmpfile8 BmpOutput**, các thông tin sau khi xử lý sẽ lưu vào cấu trúc này và xuất thành file. Các hàm con được gọi xử lý sẽ được giải thích kĩ hơn ở phần sau. Nếu pixelSize không phải 8 / 24 / 32 BPP, sẽ thông báo lỗi và dừng chương trình.

```
29  //Ham xu ly convert anh 24/32 BPP sang 8 BPP
30  void convertBMPto8(ifstream &fi, ofstream &fo) {
31      /*Khai bao cac cau truc de luu tru cac
32      cau truc trong file BMP duoc nhap */
33      BmpDib Dib;
34      Bmpfile8 BmpOutput;
35
36      //Luu tru phan Dib cua file BMP
37      readDib(fi, Dib);
38
39      if (Dib.pixelSize == 24) {
40          //Khai bao cau truc file BMP 24bpp de doc
41          Bmpfile24 BmpInput;
42
43          //Doc cac thanh phan trong file
44          readBmpHeader(fi, BmpInput.header);
45          readDib(fi, BmpInput.dib);
46          readPixel24(fi, BmpInput);
47
48          //Chuyen file 24bpp xuong 8bpp
49          Convert24_8(BmpInput, BmpOutput);
50
51          //Xuat file 8bpp theo duong dan
52          writeBmp8(fo, BmpOutput);
53
54          //Giai phong bo nho da cap phat
55          releaseBmpPixelArray24(BmpInput.pixelarray, BmpInput.dib.imageHeight);
56          releaseBmpPixelArray8(BmpOutput.pixelarray, BmpOutput.dib.imageHeight);
57      }
58      //Tuong tu...
59      else if (Dib.pixelSize == 32) { ... }
72      else {
73          cout << "Bits per Pixel khong phai 24bit hoac 32bit!";
74      }
75  }
```

Hình 6.

Tương tự với hàm **convertBMPto8()**, hàm **zoomBMP()** cũng đọc phần Dib của file và xử lý bằng các hàm với cấu trúc riêng biệt cho từng BPP. Các hàm con được gọi sẽ giải thích kĩ ở phần sau.

```
70  //Ham zoom anh BMP theo yeu cau
71  void zoomBMP(ifstream& fi, ofstream& fo, int S) {
72      BmpDib Dib;
73      readDib(fi, Dib);
74
75      if (Dib.pixelSize == 8) {
76          //Khai bao cau truc file 8bpp de doc va xuat
77          Bmpfile8 BmpInput, BmpOutput;
78
79          //Doc cac cau truc file 8bpp
80          readBmpHeader(fi, BmpInput.header);
81          readDib(fi, BmpInput.dib);
82          readPixel8(fi, BmpInput);
83
84          //Ham xu ly, resize anh 8bpp theo yeu cau
85          resizeBmp8(BmpInput, BmpOutput, S);
86
87          //Viet cau truc xuong duong dan
88          writeBmp8(fo, BmpOutput);
89
90          //Giai phong bo nho da cap phat
91          releaseBmpPixelArray8(BmpInput.pixelarray, BmpInput.dib.imageHeight);
92          releaseBmpPixelArray8(BmpOutput.pixelarray, BmpOutput.dib.imageHeight);
93      }
94      // Tuong tu...
95      else if (Dib.pixelSize == 24) { ... }
109     else if (Dib.pixelSize == 32) { ... }
123     else {
124         cout << "Bits per Pixel khong phai 8bit, 24bit hoac 32bit!";
125     }
126 }
127
```

Hình 7.

Hàm **main()** dùng để xử lý thao tác người dùng, chứa các câu lệnh thông báo để người dùng dễ sử dụng hơn, cũng như nhận biết người dùng đang sử dụng console hoặc thông qua tham số dòng lệnh để có cách xử lý khác nhau.

Trường hợp **argc == 1**, chương trình sẽ hiểu đang được chạy ở dạng console. Biến choice sẽ nhận yêu cầu của người dùng, gọi các hàm phù hợp hoặc dừng chương trình. Thông báo sau khi xử lý xong.

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

    //Dung de chay chuong trinh console
    if (argc == 1) {
        //Menu chuong trinh
        cout << "\nMenu chuong trinh:\n";
        cout << "1. Convert anh 24/32 BPP sang anh 8 BPP.\n";
        cout << "2. Thu nho anh 8/24/32 BPP theo thong so S.\n";
        cout << "0. Ngung chuong trinh.\n";
        cout << "\nXin moi ban nhap lua chon: ";

        int choice = -1; cin >> choice;
        while (choice < 0 && choice > 2) {
            cout << "Lua chon khong hop le. Xin nhap lai.\n";
            cin >> choice;
        }

        if(choice == 0)
            return 0;

        cin.ignore();
        InputFileIn();
        InputFileOut();

        ifstream fi(filename_in, ios::binary);
        ofstream fo(filename_out, ios::binary);
```

```
        //Error handler
        if (fi.fail() || fo.fail()) {
            cout << "Loi doc/mo file! Vui long kiem tra.\n";
            cout << filename_in << " " << filename_out;
            return 0;
        }

        if (choice == 1) {
            convertBMPto8(fi, fo);
        }

        if (choice == 2) {
            cout << "\nNhap vao S la thong so zoom: ";
            int S; cin >> S;
            zoomBMP(fi, fo, S);
        }

        cout << "\nDa thuc hien xong yeu cau. Thoat chuong trinh...\n";
    }
```

Hình 8, 9.

Tạo ra trường hợp khi người dùng nhập sai số argc và thông báo lỗi. Khi **argc == 4**, sẽ kiểm tra đã đọc đúng tham số dòng lệnh hay không. Tiến hành đọc file và thông báo nếu có bất kì lỗi nào. Sau đó sẽ gọi **convertBMPto8()** để thực hiện yêu cầu **-conv** như đề bài đã cho.

```
//Error handler - Xu ly khi nhap sai
if (argc > 1 && argc < 4 || argc > 5) {
    cout << "Lenh Nhap Vao Khong Hop Le!\n";
    cout << "Cau truc cau lenh: 20120219.exe -conv <Input Path> <Output Path>\n";
    cout << "20120219.exe -zoom <Input Path> <Output Path> <ratio>\n";
    return 0;
}

//Ham xu ly argv -conv
if (argc == 4) {

    //Error handler - Khu vuc xu ly loi

    //Kiem tra argv co dung yeu cau hay khong
    if (strcmp(argv[1], "-conv") != 0) {
        cout << "Lenh Nhap Vao Khong Hop Le !\n";
        cout << "Cau truc cau lenh: 20120219.exe -conv <Input Path> <Output Path>\n";
        return 0;
    }

    //Tao cac luong doc viet file BMP
    ifstream fi(argv[2], ios::binary);
    ofstream fo(argv[3], ios::binary);

    //Error handler
    if (fi.fail() || fo.fail()) {
        cout << "Loi doc/mo file! Vui long kiem tra.";
        return 0;
    }

    convertBMPto8(fi, fo);
}
```

Hình 10.

Tương tư, **argc == 5** sẽ kiểm tra, đọc và thông báo nếu có bất kì lỗi nào. Nếu không gặp lỗi, sẽ tạo biến **int S** chuyển **char argv[4]** sang số nguyên tương ứng với S trong đề bài, gọi hàm **zoomBMP()** để thực hiện yêu cầu -zoom như đề bài đã cho.

```
//Ham xu ly argv - zoom
if (argc == 5) {

    //Error handler - Khu vuc xu ly loi

    //Kiem tra argv co dung yeu cau hay khong
    if (strcmp(argv[1], "-zoom") != 0) {
        cout << "Lenh Nhap Vao Khong Hop Le !\n";
        cout << "Cau truc cau lenh: 20120219.exe -zoom <Input Path> <Output Path> S\n";
        return 0;
    }

    //Tao cac luong doc viet file BMP
    ifstream fi(argv[2], ios::binary);
    ofstream fo(argv[3], ios::binary);

    //Error handler
    if (fi.fail() || fo.fail()) {
        cout << "Loi doc/mo file! Vui long kiem tra.";
        return 0;
    }

    //Chuyen S tu char sang int
    int S = atoi(argv[4]);

    zoomBMP(fi, fo, S);
}

return 0;
```

Hình 11.

2. File Source.cpp

Hàm **readBmpHeader()** và **readDib()** nhận vào stream đọc **f**. Sử dụng **seekg()** để di chuyển con trỏ đến đúng nơi cần đọc và dùng **read()** để đọc block thông tin và chuyển thông tin vào các biến trong các struct dành riêng cho các phần thông tin trong file BMP.

```
//Doc header BMP
void readBmpHeader(ifstream& f, BmpHeader& inp)
{
    //Error handler
    if (f.fail())
        return;
    //Di chuyen con tro den dau file
    f.seekg(0, f.beg);
    /*Doc tat ca bit trong phan BMP Header
    va chuyen vao cau truc BmpHeader inp*/
    f.read((char*)&inp, sizeof(BmpHeader));
}

//Doc dib BMP
void readDib(ifstream& f, BmpDib& inp)
{
    //Error handler
    if (f.fail())
        return;
    //Di chuyen con tro den sau vi tri BmpHeader
    f.seekg(sizeof(BmpHeader), f.beg);
    /*Doc tat ca bit trong phan BMP Dib
    va chuyen vao cau truc BmpDib inp*/
    f.read((char*)&inp, sizeof(BmpDib));
}
```

Hình 12.

Hàm **readColorTable8()** dùng để đọc bảng màu file 8bpp. Dùng để giữ bảng pallete cũ sau khi resize file 8bpp. Tuy nhiên, do đề bài chỉ yêu cầu thu nhỏ ảnh 8 bpp (không có màu) nên em không sử dụng hàm này nữa.

```
//Ham doc bang mau
void readColorTable8(ifstream& f, Bmpfile8& inp)
{
    //Error handler
    if (f.fail())
        return;
    //Di chuyen con tro den sau vi tri BmpHeader
    f.seekg(sizeof(BmpHeader), f.beg);
    //Di chuyen con tro den vi tri cua colormap (chi o file 8bpp)
    f.seekg(inp.dib.dibSize, f.cur);

    //Doc va ghi vao inp.colormap
    for (int i = 0; i < 256; i++)
    {
        f.read((char*)&inp.colormap[i], sizeof(Color32));
    }
}
```

Hình 13.

Hàm **readPixel8(), readPixel24() và readPixel32()** đều dùng để đọc thông tin pixel có trong ảnh, tuy nhiên do cấu trúc file khác nhau nên chia ra 3 hàm riêng biệt.

```
//Ham doc pixel file 8bpp
void readPixel8(ifstream& f, Bmpfile8& inp)
{
    //Error handler
    if (f.fail())
        return;
    //Tinh padding, rieng anh 32bpp khong co padding
    char paddingCount = (4 - inp.dib.imageWidth % 4) % 4;
    //Di chuyen tro chuot den phan luu tru pixel
    f.seekg(inp.header.dataOffset, f.beg);
    //Khoi tao mang dong theo chieu cao
    inp.pixelarray = new uint8_t * [inp.dib.imageHeight];
    //Doc pixel theo tung hang
    for (int i = 0; i < inp.dib.imageHeight; i++)
    {
        /*Khoi tao pixel o cac cot, vi 8bpp nen chi
        dung 1 byte mau; 24bpp dung 3 byte; 32bpp dung 4 byte.*/
        inp.pixelarray[i] = new uint8_t[inp.dib.imageWidth];

        for (int j = 0; j < inp.dib.imageWidth; j++)
            //Doc thuoc tinh pixel
            f.read((char*)&inp.pixelarray[i][j], sizeof(uint8_t));
        //Cach mot doan padding
        f.seekg(paddingCount, f.cur);
    }
}
//Tuong tu...
void readPixel24(ifstream& f, Bmpfile24& inp){ ... }
//Tuong tu...
void readPixel32(ifstream& f, Bmpfile32& inp){ ... }
```

Hình 14.

Hàm **Convert24_8()** và **Convert32_8()** nhận vào cấu trúc dữ liệu file đọc và file sẽ ghi để xử lý và ghi vào cấu trúc file mới.

```
void Convert24_8(Bmpfile24& inp, Bmpfile8& out)
{
    //Tam thoi gan header va dib cua file cu sang file moi
    out.header = inp.header;
    out.dib = inp.dib;
    //Tao color table cho file 8bpp
    for (int i = 0; i < 256; i++) {
        out.colormap[i].red = i;
        out.colormap[i].green = i;
        out.colormap[i].blue = i;
        out.colormap[i].reverse = 0;
    }
    //Khoi tao mang dong theo chieu cao
    out.pixelarray = new uint8_t * [inp.dib.imageHeight];
    //Doc pixel theo tung hang
    for (int i = 0; i < inp.dib.imageHeight; i++) {
        //Khoi tao pixel o cac cot, vi 8bpp nen chi dung 1 byte luu du lieu
        out.pixelarray[i] = new uint8_t[inp.dib.imageWidth];
        for (int j = 0; j < inp.dib.imageWidth; j++) {
            //Tinh mau pixel moi
            uint8_t _gray_adjust = (inp.pixelarray[i][j].blue + inp.pixelarray[i][j].red + inp.pixelarray[i][j].green) / 3;
            //Ghi mau vao mang pixel cho file BMP moi
            out.pixelarray[i][j] = _gray_adjust;
        }
    }
}
//Tuong tu...
void Convert32_8(Bmpfile32& inp, Bmpfile8& out){ ... }
```

Hình 15.

Hàm **calcAverage8()**, **calcAverage24()** và **calcAverage32()** nhận vào thông tin pixel và toạ độ cần tính (top x, top y, bottom x, bottom y) để tính ra giá trị màu trung bình. Trường hợp file 8bpp có màu, khi tính sẽ ra giá trị trung bình sẽ tạo ra colortable mới, dẫn đến sai màu. Nhưng do trong bài không yêu cầu resize ảnh 8bpp, ảnh trắng đen vẫn có thể scale chuẩn. *Ngoài ra, em bị lỗi tính màu trung bình cho ảnh 32bpp sẽ ra ảnh trong suốt hiện tại chưa sửa lỗi được.*

```
/*
Tinh ra mau trung binh dua tren so pixel tu
hinh chu nhat tao boi top x, top y, bottom x, bottom y.

Vi cau truc cac file BMP voi BPP khac nhau se khac nhau
nen tao ra 3 ham rieng biet de xu ly va tra ve dung
cau truc cho tung BPP.
*/

uint8_t calcAvarage8(uint8_t** pixelarray, int tx, int ty, int bx, int by) {
    //Pixel dich se tra ve
    uint8_t res;
    res = 0;

    //Bien tong
    int res_sum = 0;

    //Cong tat ca gia tri pixel lai va tinh trung bình
    for (int i = tx; i <= bx; i++)
        for (int j = ty; j <= by; j++) {
            res_sum += pixelarray[i][j];
        }
    res = res_sum / ((bx - tx + 1) * (by - ty + 1));
    //Tra ve pixel dich
    return res;
}
//Tuong tu...
Color24 calcAvarage24(Color24** pixelarray, int tx, int ty, int bx, int by){ ... }
//Tuong tu...
Color32 calcAvarage32(Color32** pixelarray, int tx, int ty, int bx, int by){ ... }
```

Hình 16.

```
Color32 calcAvarage32(Color32** pixelarray, int tx, int ty, int bx, int by) {
    Color32 res;
    int res_sum[4];
    res_sum[0] = res_sum[1] = res_sum[2] = res_sum[3] = 0;

    for (int i = tx; i <= bx; i++)
        for (int j = ty; j <= by; j++) {
            res_sum[0] += pixelarray[i][j].blue;
            res_sum[1] += pixelarray[i][j].green;
            res_sum[2] += pixelarray[i][j].red;
            res_sum[3] += pixelarray[i][j].alpha;
        }

    res.blue = res_sum[0] / ((bx - tx + 1) * (by - ty + 1));
    res.green = res_sum[1] / ((bx - tx + 1) * (by - ty + 1));
    res.red = res_sum[2] / ((bx - tx + 1) * (by - ty + 1));
    res.alpha = res_sum[3] / ((bx - tx + 1) * (by - ty + 1));

    return res;
}
```

Hình 17. Hàm tính trung bình màu file 32bpp.

Hàm **resizeBmp8(), resizeBmp24() và resizeBmp32()** đều nhận cấu trúc file đọc, file đích và S để xử lý. Do cấu trúc các file khác nhau nên cần 3 hàm khác nhau. Do để yêu cầu resize ảnh 8bpp không màu nên em dùng colortable trắng đen. *Hiện tại hàm resizeBmp32() hoạt động không đúng (xuất ra file trong suốt).*

```
//Ham resize anh 8bpp
void resizeBmp8(Bmpfile8 inp, Bmpfile8& out, int S) {
    // Tam thoi luu lai thong tin Header va Dib cu
    out.header = inp.header;
    out.dib = inp.dib;
    // width va height cua hinh moi
    int old_w = inp.dib.imageWidth;
    int old_h = inp.dib.imageHeight;
    int new_w = (inp.dib.imageWidth + S - 1) / S;
    int new_h = (inp.dib.imageHeight + S - 1) / S;
    //Tao color table (chi co o anh 8bpp)
    for (int i = 0; i < 256; i++) {
        out.colormap[i].red = i;
        out.colormap[i].green = i;
        out.colormap[i].blue = i;
        out.colormap[i].reverse = 0;
    }
    // Thay doi thong tin dib
    out.dib.imageWidth = new_w;
    out.dib.imageHeight = new_h;
    out.pixelarray = new uint8_t * [new_h];
    // Down sampling
    for (int i = 0; i < new_h; i++) {
        out.pixelarray[i] = new uint8_t[new_w];
        for (int j = 0; j < new_w; j++) {
            int tx = i * S, ty = j * S; // top left of current kernel
            int bx = min(tx + S, old_h) - 1, by = min(ty + S, old_w) - 1; // bottom right of current kernel
            // update new pixel value by average of old pixel
            out.pixelarray[i][j] = calcAvarage8(inp.pixelarray, tx, ty, bx, by);
        }
    }
}
//Tuong tu
void resizeBmp24(Bmpfile24 inp, Bmpfile24& out, int S) { ... }
//Tuong tu
void resizeBmp32(Bmpfile32 inp, Bmpfile32& out, int S) { ... }
```

Hình 18.

Hàm **writeBmp8(), writeBmp24(), writeBmp32()** nhận vào stream xuất file và cấu trúc file đích để ghi file ra đường dẫn người dùng đã nhập. Do cấu trúc file khác nhau nên sử dụng 3 hàm riêng biệt.

```
void writeBmp8(ofstream& f, Bmpfile8& inp)
{
    //Ghi lai tat ca thong tin dib cho file bmp moi
    inp.dib.dibSize = sizeof(inp.dib);
    inp.dib.pixelSize = 8;
    inp.dib.compressMethod = 0;
    inp.dib.horizontalResolution = 0;
    inp.dib.verticalResolution = 0;
    //Tinh padding count
    char paddingCount = (4 - inp.dib.imageWidth % 4) % 4;
    inp.dib.bitmapByteCount = inp.dib.imageHeight * (inp.dib.imageWidth + paddingCount);
    inp.header.fileSize = sizeof(BmpHeader) + sizeof(BmpDib) + inp.dib.bitmapByteCount;
    //Di chuyen con tro den dau file
    f.seekp(0, f.beg);
    //Xuat cac thanh phan cua file BMP
    f.write((char*)&inp.header, sizeof(BmpHeader));
    f.write((char*)&inp.dib, sizeof(BmpDib));
    //Xuat colortable chi danh cho file 8bpp
    f.write((char*)&inp.colormap, sizeof(inp.colormap));

    //Ghi pixel data vao file BMP moi
    for (int i = 0; i < inp.dib.imageHeight; i++)
    {
        for (int j = 0; j < inp.dib.imageWidth; j++)
            f.write((char*)&inp.pixelarray[i][j], sizeof(uint8_t));
        f.write("\0\0\0\0", paddingCount);
    }
}
 //Tuong tu...
void writeBmp24(ofstream& f, Bmpfile24& inp){ ... }
 //Tuong tu...
void writeBmp32(ofstream& f, Bmpfile32& inp){ ... }
```

Hình 20.

Sau khi hoàn thành tất cả các bước, sẽ gọi hàm giải phóng bộ nhớ cho từng cấu trúc file.

```
 //Giai phong bo nho cho mang dong
void releaseBmpPixelArray8(uint8_t** pixelarray, int rowCount)
{
    for (int i = 0; i < rowCount; i++)
        delete[]pixelarray[i];

    delete[]pixelarray;
}
 //Tuong tu
void releaseBmpPixelArray24(Color24** pixelarray, int rowCount){ ... }
 //Tuong tu
void releaseBmpPixelArray32(Color32** pixelarray, int rowCount){ ... }
```

Hình 21.