ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA: CNTT

~~~~~~*~~~~~~

BÁO CÁO THU HOẠCH

Sinh viên thực hiện	:	**Nguyễn Thành Trí**
Lớp	:	**21CTT5**
Mã sinh viên	:	**21120575**
Giáo viên hướng dẫn	:	**Phạm Minh Hoàng**

Hồ Chí Minh ngày 18 tháng 5 năm 2022
~~~~~~

I. Tổng quan

Sau 1 khoảng thời gian tìm hiểu và thực hiện, em xin báo cáo về phần bài tập lớn cá nhân của mình.

+ Các chức năng của chương trình :
  - Đọc và ghi một ảnh bmp (8/24/32 bit) bằng được dẫn do người dùng nhập.
  - Chuyển đổi một ảnh bmp 24/32 bit sang một ảnh bmp 8 bit.
  - Phóng to một ảnh bmp (8/24/32 bit) với tỉ lệ S cho trước

+ Cấu trúc của chương trình bao gồm :
  - File header.h : Chứa các thư viện ,các hàm nhập / xuất ,giới thiêu ,đọc / ghi và giải phóng bộ nhớ

  - File 21120575.cpp : Chứa các hàm xử lí ảnh theo yêu cầu và xử lí các thao tác cơ bản của người dùng

- File header.h khai báo các thư viện cần thiết để chạy chương trình, chứa các struct để lưu trữ các thuộc tính trong file BMP và các hàm cơ bản.

```
1    #pragma once
2    #include ...
8    using namespace std;
9    #pragma pack(1)
10
11   // cau truc header
12   struct header { ... };
18
19   // cau truc dib
20   struct dib { ... };
34
35   // cau truc cua 1 pixel diem anh cua hinh bmp 8bit
36   struct pix8 { ... };
39
40   // cau truc cua 1 pixel diem anh cua hinh bmp 24bit
41   struct pix24 { ... };
46
47   // cau truc cua 1 pixel diem anh cua hinh bmp 32bit
48   struct pix32 { ... };
54
55   // bang mau cua hinh bmp 8bit
56   struct colortable8 { ... };
62
63   struct bmp { ... };
71
72   // cau truc cua 1 file bmp 8
73   struct bmp8 { ... };
81
82   // cau truc của 1 file bmp 24
83   struct bmp24 { ... };
90
91   // cau truc của 1 file bmp 32
92   struct bmp32 { ... };
99
100  char path_in[512], path_out[512];
```

Hình 1. Khai báo các cấu trúc của 1 file bmp 8/24/32 bit trong file header

```
102     // gioi thieu
103   ⊞void introduce() { ... }
111
112   ⊞void input_path(char path_in[512]) { ... }
116
117   ⊞void output_path(char path_out[512]) { ... }
121
122     // ham doc file bmp 8bit
123   ⊞int read8(const char* filename, bmp8& bmp) { ... }
167
168     // ham doc file bmp 24bit
169   ⊞int read24(char* filename, bmp24& bmp) { ... }
208
209     // ham doc file bmp 32bit
210   ⊞int read32(const char* filename, bmp32& bmp) { ... }
248
249     // ham doc file bmp
250   ⊞int readglobal(char* filename, bmp& bmp) { ... }
295
296     // ham xuat thong tin file bmp 8bit
297   ⊞void showinfo8(bmp8 a, const char* name) { ... }
317
318     // ham xuat thong tin file bmp 24bit
319   ⊞void showinfo24(bmp24 a, const char* name) { ... }
339
340     // ham xuat thong tin file bmp 32bit
341   ⊞void showinfo32(bmp32 a, const char* name) { ... }
361
362     // ham ghi file bmp 8bit
363   ⊞int write8( char* filename, bmp8 bmp) { ... }
394
395     // ham ghi file bmp 24bit
396   ⊞int write24(const char* filename, bmp24 bmp) { ... }
423
424     // ham ghi file bmp 32 bit
425   ⊞int write32(const char* filename, bmp32 bmp) { ... }
452
453     // ham giai phong du lieu mang pix 8
454   ⊞void releasepix8(pix8*& pixel) { ... }
457
458     // ham giai phong du lieu mang pix 24
459   ⊞void releasepix24(pix24*& pixel) { ... }
462
463     // ham giai phong du lieu mang pix 32
464   ⊞void releasepix32(pix32*& pixel) { ... }
```

Hình 2 : Các file đọc / ghi , xuất thông tin , giải phóng bộ nhớ của các file bmp 8/24/32 bit trong file header

- File 21120575.cpp chứa các hàm xử lí các yêu cầu của đề như chuyển ảnh 24/32 bit sang 8 bit , thu nhỏ ảnh 8/24/32 bit và các hàm tính toán các điểm ảnh.

```
2
3     #include "header.h"
4     // ham chuyen du lieu diem anh vao du lieu pix cua hinh 8bit
5   ⊞pix8* convertDataToPixelArray8(char*& data, dib& src) { ... }
27
28    // ham chuyen du lieu diem anh vao du lieu pix cua hinh 24bit
29  ⊞pix24* convertDataToPixelArray24(char*& data, dib& src) { ... }
53
54    // ham chuyen du lieu diem anh vao du lieu pix cua hinh 32bit
55  ⊞pix32* convertDataToPixelArray32(char*& data, dib& src) { ... }
80
81    // ham chuyen du lieu pix cua hinh 8bit sang du lieu diem anh
82  ⊞char* convertPixelArray8ToData(pix8*& pixels, dib& dst) { ... }
104
105   // ham chuyen du lieu pix cua hinh 24bit sang du lieu diem anh
106 ⊞char* convertPixelArray24ToData(pix24*& pixels, dib& dst) { ... }
132
133   // ham chuyen du lieu pix cua hinh 32bit sang du lieu diem anh
134 ⊞char* convertPixelArray32ToData(pix32*& pixels, dib& dst) { ... }
162
163   // ham chuyen doi file bmp 24bit sang file bmp 8bit
164 ⊞int convert24to8bit(bmp24 src, bmp8& dst) { ... }
219
220   // ham chuyen doi file bmp 24bit sang file bmp 8bit
221 ⊞int convert32to8bit(bmp32 src, bmp8& dst) { ... }
273
274   // ham tinh trung binh mang pixel theo ti le s cua file bmp 8bit
275 ⊞void average8(pix8* srcpixels, pix8*& dstpixels, bmp8 dst, int s) { ... }
304
305   // ham tinh trung binh mang pixel theo ti le s cua file bmp 24bit
306 ⊞void average24(pix24* srcpixels, pix24*& dstpixels, bmp24 dst, int s) { ... }
342
343   // ham tinh trung binh mang pixel theo ti le s cua file bmp 24bit
344 ⊞void average32(pix32* srcpixels, pix32*& dstpixels, bmp32 dst, int s) { ... }
385
386   // ham thu nho file bmp 8bit
387 ⊞int resize8(bmp8 src, bmp8& dst, int s) { ... }
425
426   // ham thu nho file bmp 24bit
427 ⊞int resize24(bmp24 src, bmp24& dst, int s) { ... }
465
466   // ham thu nho file bmp 32bit
467 ⊞int resize32(bmp32 src, bmp32& dst, int s) { ... }
505
506 ⊞int main(int argc, char* argv[]) { ... }
```

Hình 3 : Các hàm chuyển đổi , xử lí cơ bản trong file 21120575.cpp

II. Chi tiết

1. File header.h

- Đầu tiên sẽ include các thư viện cần thiết để chạy được các hàm. Tiếp theo sẽ đến khai báo các dữ liệu cấu trúc của các thành phần cần thiết trong 1 file bmp

```
#pragma once
#include<iostream>
#include<string.h>
#include<fstream>
#include<math.h>
#include<stdio.h>
#include<conio.h>
using namespace std;
#pragma pack(1)

// cau truc header
struct header {
    char type[2];
    uint32_t size;
    uint32_t reserved;
    uint32_t dataoffset;
};

// cau truc dib
struct dib {
    uint32_t size;
    int32_t  width;
    int32_t  height;
    uint16_t planes;
    uint16_t bpp;

    uint32_t compression;
    uint32_t imagesize;
    int32_t  xpb;
    int32_t  ypb;
    uint32_t colorused;
    uint32_t colorimportant;
};
```

Hình 4

```
// cau truc cua 1 pixel diem anh cua hinh bmp 8bit
struct pix8 {
    uint8_t color;
};

// cau truc cua 1 pixel diem anh cua hinh bmp 24bit
struct pix24 {
    uint8_t b;
    uint8_t r;
    uint8_t g;
};

// cau truc cua 1 pixel diem anh cua hinh bmp 32bit
struct pix32 {
    uint8_t a;
    uint8_t b;
    uint8_t r;
    uint8_t g;
};

// bang mau cua hinh bmp 8bit
struct colortable8 { ... };

struct bmp {
    header header;
    dib dib;
    pix24 pix;
    colortable8 colortable[256];
    char* pdibreversed;
    char* pimagedata;
};
```

Hình 5

```
100     // gioi thieu
101     void introduce() {
102         cout << "------------BTL BMP----------" << endl;
103         cout << "|Ho va Ten: Nguyen Thanh Tri |" << endl;
104         cout << "|MSSV : 21120575             |" << endl;
105         cout << "|Lop : 21CTT5                |" << endl;
106         cout << "|GVHD : Pham Minh Hoang      |" << endl;
107         cout << "-----------------------------" << endl;
108     }
```

Hình 6 : Giới thiệu thông tin cơ bản

```
109
110     char path_in[512], path_out[512];
111
112     void input_path(char path_in[512]) {
113         cout << "\nNhap dia chi doc file (VD : D:\\pic_in.bmp ) : ";
114         cin.getline(path_in, 512);
115     }
116
117     void output_path(char path_out[512]) {
118         cout << "\nNhap dia chi xuat file (VD : D:\\pic_out.bmp ) : ";
119         cin.getline(path_out, 512);
120     }
```

Hình 7 : Hàm nhập các đường dẫn / xuất của file ảnh bmp cần thao tác

```
// ham doc file bmp 8bit
int read8(const char* filename, bmp8& bmp) {
    ifstream f(filename, ios::binary);
    // kiem tra file mo duoc khong
    if (!f.is_open()) {
        cout << "Khong the mo file ";
        return 0;
    }
    // dua con tro den vi tri dau file
    f.seekg(0, f.beg);

    // doc header
    f.read((char*)&bmp.header, 14);

    // doc dib
    f.read((char*)&bmp.dib, 40);

    // kiem tra xem co phan du khong
    if (bmp.dib.size > 40) {
        // tinh kich thuoc phan du
        int t = bmp.dib.size - 40;
        // cap phat vung nho phan du
        bmp.pdibreversed = new char[t];
        // doc phan du
        f.read(bmp.pdibreversed, t);
    }
```

- Hàm đọc file nhận vào 2 tham số là đường dẫn filename và biến cấu trúc bmp. Đầu tiên sẽ kiểm tra đường dẫn file hợp lệ hay không. Tiếp theo sẽ gán con trỏ về vị trí đầu file bằng hàm seekg. Lần lượt đọc header , dib , phần dư (nếu có) , bảng màu (chỉ có ở bmp 8bit) và dữ liệu điểm ảnh vào từng biến cấu trúc phù hợp

Hình 8 : Hàm đọc file

```
// doc bang mau cua file bmp 8bit (chi danh cho 8bit)
if (bmp.dib.bpp == 8) {
    f.read((char*)&bmp.colortable, sizeof(colortable8) * 256);
}

// cap phat vung nho cua diem anh
bmp.pimagedata = new char[bmp.dib.imagesize];

// doc du lieu diem anh
f.read(bmp.pimagedata, bmp.dib.imagesize);

f.close();

return 1;
```

Hình 9 : Hàm đọc file

- Tương tự như vậy với các hàm đọc file bmp 24bit và bmp 32bit

```
// ham ghi file bmp 8bit
int write8( char* filename, bmp8 bmp) {
    ofstream f(filename, ios::binary);

    // kiem tra
    if (!f) {
        cout << "Khong the ghi file ";
        return 0;
    }

    // ghi header vao file
    f.write((char*)&bmp.header, 14);

    // ghi dib vao file
    f.write((char*)&bmp.dib, 40);
```

- Hàm ghi file nhận vào 2 tham số là đường xuất filename và biến cấu trúc bmp. Đầu tiên sẽ kiểm tra đường dẫn file hợp lệ hay không. Tiếp theo lần lượt ghi header , dib , phần dư (nếu có) , bảng màu (chỉ có ở bmp 8bit) và dữ liệu điểm ảnh vào file bmp.

Hình 10 : Ghi file

```
    if (bmp.dib.size > 40) {
        int t = bmp.dib.size - 40;
        // ghi phan du vao file
        f.write(bmp.pdibreversed, t);
    }

    // ghi bang mau cua file bmp 8bit vao file
    if (bmp.dib.bpp == 8) {
        f.write((char*)&bmp.colortable, sizeof(colortable8) * 256);
    }
    // ghi du lieu diem anh vao file
    f.write(bmp.pimagedata, bmp.dib.imagesize);

    f.close();
    return 1;
}
```

Hình 11 : Ghi file

- Tương tự như vậy với các hàm ghi file bmp 24bit và bmp 32bit

- Cuối cùng là các hàm giải phóng bộ nhớ các mảng động của điểm ảnh

```
// ham giai phong du lieu mang pix 8
void releasepix8(pix8*& pixel) {
    delete[] pixel;
}

// ham giai phong du lieu mang pix 24
void releasepix24(pix24*& pixel) {
    delete[] pixel;
}

// ham giai phong du lieu mang pix 32
void releasepix32(pix32*& pixel) {
    delete[] pixel;
}
```

Hình 12 : Hàm giải phóng bộ nhớ

2. File 21120575.cpp

- Về các bước chuyển đổi cơ bản của mảng dữ liệu điểm ảnh, em đã tạo ra 2 hàm để chuyển đổi qua lại giữa mảng dữ liệu điểm ảnh và mảng dữ liệu pixel để có thể dễ dàng thao tác trên các mảng.

- **Hàm chuyển đổi từ mảng dữ liệu điểm ảnh sang mảng dữ liệu pixel**

- Đầu tiên là hàm chuyển đổi từ mảng dữ liệu điểm ảnh sang mảng dữ liệu pixel. Hàm trả về một mảng động pixel 8/24/32 , nhận 2 tham số là mảng dữ liệu điểm ảnh của hình và phần dib của file. Sau khi tính kích thước của hình và padding, em cấp phát một mảng động có kiểu dữ liệu pix 8/24/32 với kích thước là kích thước của hình (sau khi chạy thì visual studio có warning em là nên cấp phát với kích thước là sizearray*2 nhưng em cũng không biết tại sao). Sau đó em tạo một con trỏ temp và đọc các điểm ảnh vào từng ô pixel, mỗi khi kết thúc 1 hàng thì con trỏ sẽ bỏ qua padding. Hàm trả về mảng pixel sau khi đọc xong

```
// ham chuyen du lieu diem anh vao du lieu pix cua hinh 8bit
pix8* convertDataToPixelArray8(char*& data, dib& src) {
    // khai bao kich thuoc cua hinh
    int sizearray = src.width * src.height;

    // cap phat vung nho cho mang du lieu pix8
    pix8* pixels = new pix8[sizearray * 2];

    // tinh papdding
    int padding = (4 - (src.width * src.bpp / 8) % 4) % 4;

    char* temp = data;

    // ghi cac du lieu diem anh vao mang du lieu pix8
    for (int i = 0; i < src.height; i++) {
        for (int j = 0; j < src.width; j++) {
            pixels[i * src.width + j].color = *(temp++);
        }
        // bo qua cac padding
        temp += padding;
    }
    return pixels;
}

// ham chuyen du lieu diem anh vao du lieu pix cua hinh 24bit
pix24* convertDataToPixelArray24(char*& data, dib& src) { ... }

// ham chuyen du lieu diem anh vao du lieu pix cua hinh 32bit
pix32* convertDataToPixelArray32(char*& data, dib& src) { ... }
```

Hình 13 : Hàm chuyển đổi từ mảng dữ liệu điểm ảnh sang mảng dữ liệu pixel

- Tương tự như vậy với hàm chuyển đổi từ mảng dữ liệu điểm ảnh sang mảng dữ liệu pixel của hình bmp 24 và 32 bit

- **Hàm chuyển đổi từ mảng dữ liệu pixel sang mảng dữ liệu điểm ảnh**

- Tiếp theo là hàm chuyển đổi từ mảng dữ liệu pixel sang mảng dữ liệu điểm ảnh. Hàm trả về một mảng động điểm ảnh, nhận 2 tham số là mảng dữ liệu pixel 8/24/32 bit của hình và phần dib của file. Sau khi tính kích thước của hình và padding, em cấp phát một mảng động có kiểu dữ liệu char với kích thước là kích thước của hình. Sau đó em tạo một con trỏ temp và đọc các ô pixel vào từng điểm ảnh, mỗi khi kết thúc 1 hàng thì các điểm ảnh ở phần padding sẽ được gán bằng 0. Hàm trả về mảng điểm ảnh sau khi đọc xong

```
// ham chuyen du lieu pix cua hinh 8bit sang du lieu diem anh
char* convertPixelArray8ToData(pix8*& pixels, dib& dst) {
    // tinh padding
    int padding = (4 - (dst.width * dst.bpp / 8) % 4) % 4;

    // tinh kich thuoc cua mang du lieu diem anh
    int size = dst.width * dst.height * (dst.bpp / 8) + padding * dst.height;

    // cap phat vung nho cho mang du lieu diem anh
    char* data = new char[size];
    char* temp = data;

    // ghi cac du lieu pix vao mang du lieu diem anh
    for (int i = 0; i < dst.height; i++) {
        for (int j = 0; j < dst.width; j++) {
            *(temp++) = pixels[i * dst.width + j].color;
        }
        for (int k = 0; k < padding; k++) {
            *(temp++) = 0;
        }
    }
    return data;
}

// ham chuyen du lieu pix cua hinh 24bit sang du lieu diem anh
char* convertPixelArray24ToData(pix24*& pixels, dib& dst) { ... }

// ham chuyen du lieu pix cua hinh 32bit sang du lieu diem anh
char* convertPixelArray32ToData(pix32*& pixels, dib& dst) { ... }
```

Hình 14 : Hàm chuyển đổi từ mảng dữ liệu pixel sang mảng dữ liệu điểm ảnh

- Tương tự như vậy với hàm chuyển đổi từ mảng dữ liệu pixel sang mảng dữ liệu điểm ảnh của hình bmp 24 và 32 bit

- **Hàm chuyển đổi từ ảnh 24/32 bit sang 8bit**

- Hàm nhận 2 tham số được truyền vào là một biến dữ liệu của ảnh bmp 24bit và một biến dữ liệu của ảnh bmp 8bit. Đầu tiên em sẽ kiểm tra bit per pixel của ảnh src , nếu không phải 24/32 bit thì hàm sẽ kết thúc. Sau đó em sẽ gán header , dib và phần dư từ biến của hình 24bit sang biến của hình 8bit. Sau đó gán bit per pixel của biến của hình 8bit bằng 8. Lần lượt tính padding và kích thước của hình. Sau đó lần lượt gán bảng màu cho hình 8bit.

```
int convert24to8bit(bmp24 src, bmp8& dst) {
    // kiem tra bpp cua file src
    if (src.dib.bpp != 24 || src.dib.bpp == NULL)
        return 0;

    // gan header,dib,reveresed cua file source cho destination
    dst.header = src.header;
    dst.dib = src.dib;
    dst.pdibreversed = src.pdibreversed;

    // gan bpp cua file destinaton bang 8
    dst.dib.bpp = 8;

    // tinh padding
    int padding = (4 - (dst.dib.width * dst.dib.bpp / 8) % 4) % 4;

    // tinh kich thuoc cua anh
    dst.dib.imagesize = dst.dib.width * dst.dib.height * dst.dib.bpp / 8 + padding * dst.dib.height;

    // gan mau cho bang mau cua file bmp 8bit
    for (int i = 0; i < 256; i++) {
        dst.colortable[i].b = i;
        dst.colortable[i].g = i;
        dst.colortable[i].r = i;
        dst.colortable[i].reserved = 0;
    }
```

Hình 15 : Hàm chuyển đổi ảnh 24bit sang 8bit

Tiếp theo em sẽ chuyển dữ liệu điểm ảnh của hình 24 bit sang mảng dữ liệu pixel bằng hàm convertDataToPixelArray24/32 , khai báo mảng động dữ liệu pixel cho hình 8bit . Lần lượt tính các giá trị trung bình của 1 pixel bằng 2 vòng for như hình. Sau khi tính giá trị trung bình thì em sẽ chuyển đổi mảng pixel của hình 8bit về mảng dữ liệu điểm ảnh bằng hàm convertPixelArray24/32toData.

```
// tao mot mang srcpixel chua cac du lieu pix cua file bmp 24bit
pix24* srcpixels = convertDataToPixelArray24(src.pimagedata, src.dib);

// cap phat bo nho cho mang du lieu pix cua file bmp 8bit
// tam thoi luu pix cua file bmp8bit vao trong mang pix24
pix24* dstpixels = new pix24[dst.dib.width * dst.dib.height];


char ave;
// tinh gia tri trung binh cua 1 diem anh pix va gan vao pix cua destination
for (int i = 0; i < dst.dib.height; i++) {
    for (int j = 0; j < dst.dib.width; j++) {
        int index = i * dst.dib.width + j;
        ave = (char)((srcpixels[index].r + srcpixels[index].g + srcpixels[index].b) / 3);
        dstpixels[index].r = dstpixels[index].g = dstpixels[index].b = ave;
    }
}

// mang pix cua destination khi vao trong ham convert se chuyen thanh du lieu diem anh cua file bmp8bit
dst.pimagedata = convertPixelArray24ToData(dstpixels, dst.dib);


// giai phong bo nho cho 2 mang pix cua src va dst file
releasepix24(srcpixels);
releasepix24(dstpixels);

return 1;
```

Hình 16 : Hàm chuyển đổi ảnh 24bit sang 8bit

- Tương tự như vậy với hàm chuyển đổi ảnh từ 32bit sang 8bit

- **Hàm thu nhỏ ảnh 8/24/32 bit theo tỉ lệ S**

- Hàm sẽ nhận 3 tham số được truyền vào là 2 biến dữ liệu của ảnh 8/24/32 bit và 1 tỉ lệ s . Đầu tiên em sẽ kiểm tra bit per pixel của ảnh. Sau đó em sẽ gán biến dữ liệu source cho biến dữ liệu destination. Lần lượt tính padding và kích thước của mảng động. Chuyển dữ liệu điểm ảnh của biến src sang dữ liệu pixel và tính trung bình tỉ lệ S bằng hàm averag. Tính lại chiều dài , chiều rộng , padding và kích thước của biến destination . Chuyển dữ liệu pixel của biến destination sang dữ liệu điểm ảnh.

```
// ham thu nho file bmp 8bit
int resize8(bmp8 src, bmp8& dst, int s) {
    // kiem tra bpp
    if (src.dib.bpp != 8)
        return 0;

    // gan du lieu cua src vao dst
    dst = src;

    // tinh padding
    int paddingsrc = (4 - (src.dib.width * src.dib.bpp / 8) % 4) % 4;

    // chuyen du lieu diem anh cua file src vao du lieu pix cua file src
    pix8* srcpixels = convertDataToPixelArray8(src.pimagedata, src.dib);

    // cap phat bo nho
    pix8* dstpixels = new pix8[dst.dib.width * dst.dib.height];

    // ham xu li
    average8(srcpixels, dstpixels, dst, s);

    // tinh lai chieu dai , chieu rong
    dst.dib.height = (dst.dib.height / s);
    dst.dib.width = (dst.dib.width / s);

    // tinh lai padding anh
    int paddingdst = (4 - (dst.dib.width * dst.dib.bpp / 8) % 4) % 4;
    // tinh lai kich thuoc anh
    dst.dib.imagesize = dst.dib.height * dst.dib.width * dst.dib.bpp / 8 + paddingdst * dst.dib.height;

    // chuyen du lieu pix cua hinh 8bit sang du lieu diem anh
    dst.pimagedata = convertPixelArray8ToData(dstpixels, dst.dib);

    // giai phong bo nho cho 2 mang pix cua src va dst file
    releasepix8(srcpixels);
    releasepix8(dstpixels);

    return 1;
}
```

Hình 17 : Hàm thu nhỏ ảnh 8bit với tỉ lệ S cho trước

- Tương tự như vậy với hàm thu nhỏ ảnh 24/32 bit với tỉ lệ S cho trước.

- **Hàm tính trung bình các điểm ảnh theo tỉ lệ S**

- Hàm nhận vào 4 tham số truyền vào , 2 mảng dữ liệu pixel src và dst, biến dữ liệu của hình bmp 8/24/32 bit và tỉ lệ S. Tiếp theo em sẽ khai báo một biến để tính tổng các điểm pixel.
- 2 vòng for đầu tiên sẽ tạo 1 vòng lặp chạy qua từng ô có kích thước S*S của hình.
- 2 vòng for tiếp theo sẽ tạo 1 vòng lặp chạy qua từng điểm pixel có trong ô của 2 vòng for đầu tiên.

- Ở trong 4 vòng for sẽ là các bước xử lí (tính tổng các pixel theo các màu B,G,R đối với từng hình 8/24/32 bit khác nhau và ép kiểu)

```
// ham tinh trung binh mang pixel theo ti le s cua file bmp 8bit
void average8(pix8* srcpixels, pix8*& dstpixels, bmp8 dst, int s) {
    unsigned int color;

    int squarex, squarey, index = 0;

    // hai vong for dau tien - chay qua tung o vuong co kich thuoc s*s trong mang du lieu diem anh
    // hai vong for tiep theo - chay qua tung pixel trong cac o vuong cua 2 vong for dau tien
    for (int i = 0; i <= dst.dib.height - s; i += s) {
        for (int j = 0; j <= dst.dib.width - s; j += s) {
            color = 0;
            squarey = 0;
            for (int k = i; squarey < s; k++) {
                squarex = 0;
                for (int l = j; squarex < s; l++) {
                    // tinh tong cac pix của file bmp 8bit trong mang pix
                    color += srcpixels[k * dst.dib.width + l].color;
                    squarex++;
                }
                squarey++;
            }
            // chia trung binh tong cac pix trong mang
            color = (char)(color / (s * s));
            // gan vao mang pix cua file destination
            dstpixels[index].color = (char)color;
            index++;
        }

    }
}
```

Hình 18 : Hàm thu nhỏ ảnh 8bit với tỉ lệ S cho trước

- **Hàm main**

- Đầu tiên em sẽ tạo 1 biến global để đọc bit per pixel của hình được truyền vào , với bit per pixel phù hợp hình sẽ được xử lí theo đúng yêu cầu của người dùng.

- Lần lượt là các hàm introduce , nhập path_in / path_out và các lựa chọn cho người dùng

```
int main(int argc, char* argv[]) {
    // tao mot bien global dang bmp de kiem tra xem file input co bpp bang bao nhieu
    bmp global;

    bmp24 src24;
    bmp32 src32;
    bmp8 src8;
    bmp8 dst;

    introduce();
    if (argc == 1) {
        // Menu
        cout << endl << "Menu" << endl;
        cout << "1. Convert anh 24/32 bit sang anh 8 bit" << endl;
        cout << "2. Thu nho anh 8/24/32 bit theo ti le S " << endl;
        cout << "0. Ngung chuong trinh" << endl;
        cout << "Nhap lua chon : ";


        int selection = -1;
        cin >> selection;

        while (selection < 0 || selection > 2) {
            cout << "Lua chon khong hop le. Moi ban nhap lai : ";
            cin >> selection;
        }

        if (selection == 0)
            return 0;

        cin.ignore();
        input_path(path_in);
        output_path(path_out);

        if (selection == 1) { ... }
        if (selection == 2) { ... }
    }

    if (argc == 4) { ... }

    if (argc == 5) { ... }

    _getch();
    return 1;

}
```

Hình 18 : Hàm main