Комп’ютерний практикум № 02

Комплексні типи даних: масиви, рядки, структури, об’єднання.

Хід роботи

**Завдання 1**

*Розробити структуру даних для збереження інформації про 1 піксель з використанням об’єднань та структур. Розробити структуру даних для збереження інформації про зображення (в динамічній купі оперативної пам’яті). Створити наступні функції:*

*- create\_white - функція, що створює біле полотно заданого розміру;*

*- create\_black - функція, що створює чорне полотно заданого розміру;*

*- create\_color - функція, що створює полотно, де колір кожного каналу є різницею індексів пікселя.*

*В звіті навести скріншот отриманого за допомогою функції create\_color зображення.*

**Структура даних, для збереження інформації про один піксель:**

union Pixel{

    uint32\_t argb;

    struct {

    char r;

    char g;

    char b;

    char a;

    };

};

**Структура даних, для збереження інформації про зображення:**

struct Image {

UINT width;//width

UINT height;//height

char type[3];//magic number

unsigned char Maxval;//The maximum color value (Maxval) char beacose <=255

Pixel \*\*canvas; //value of color

void setPixel(int i, int j, Pixel p){ canvas[i][j] = p;}

Pixel getPixel(int i, int j ){return canvas[i][j]; }

Image(char \* t, UINT color,const int w, const int h):

Maxval(color),width(w), height(h){

strcpy(type, t);

canvas = new Pixel\*[height];

for (int i = 0; i < height; i++)

{

canvas[i] = new Pixel[width];

}

}

//Конструктор копирования

Image(const Image & img) {

strcpy(type,img.type);

Maxval = img.Maxval;

width = img.width;

height = img.height;

canvas = new Pixel\*[height];

for (int i = 0; i <height; i++)

{

canvas[i] = new Pixel[width];

}

//copy all values

for (int i = 0; i < height; i++) {

for (int j = 0; j < width; j++) {

canvas[i][j].argb =img.canvas[i][j].argb;

}

}

}

~Image() {

for (int i = 0; i < height; i++)

{

delete[] canvas[i];

}

delete[] canvas;

}

};

**Функція створення полотна, заданого варіантом:**

void create\_color(Image \* variant) {

for (int i = 0; i < variant->height; i++) {

for (int j = 0; j < variant->width; j++) {

variant->canvas[i][j].r = j;

variant->canvas[i][j].g = j;

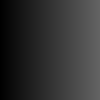
variant->canvas[i][j].b = j;

variant->canvas[i][j].a = 0;

}

}

}



**Завдання 2**

*Реалізувати функції збереження/завантаження зображення до/з файлу в форматі ​PPM​. В звіті навести зміст отриманого PPM файлу для зображення розміру 5 х 5 пікселів.*

**Функція збереження зображення в PPM файл:**

bool save\_picture(Image \*myImage, string fileName) {

ofstream picture( fileName, ios::out | ios::binary);

if (!picture.is\_open()) {

cout << "Can't open file. Check it please!" << endl;

return false;

}

picture << myImage->type<<'\n' << myImage->width << " " << myImage->height << "\n" <<(int)myImage->Maxval<< "\n";

for (int i = 0; i < myImage->height; i++) {

for (int j = 0; j < myImage->width; j++) {

picture<<myImage->canvas[i][j].r<<myImage->canvas[i][j].g <<myImage->canvas[i][j].b;

}

//picture << endl;

}

picture.close();

return true;

}

**Функція завантаження зображення в PPM файл:**

bool upload\_picture(Image \*& myImage, string fileName) {

char line[MAX];

char \* ptr;

int temp;

int sizeX = 0;

int sizeY = 0;

int colorMax = 0;

ifstream picture(fileName, ios::binary);

if (!picture.is\_open()) {

cout << "Can't open file. Check it please!" << endl;

return false;

}

cout << "File opened!" << endl;

#pragma region Read Head

picture.read(line, MAX);

if (line[0] != 'P'&& line[1] != '6')//read magic number

{

cout << line << "\nThis is not .PPM file!" << endl;

return false;

}

char type[3];

strncpy(type, line, 2);

type[2] = '\0';

cout << type << endl;

ptr = line + 2;

while (!colorMax)

{

temp = 0;

if (\*ptr == '#')while (\*ptr != '\n')ptr++;//without comment

while (\*ptr == ' ')ptr++; //

if (\*ptr == '\n' || \*ptr == '\r' || \*ptr == '\t')

ptr++;

temp = atoi(ptr);// read number

if (temp)

{

if (!sizeX)

sizeX = temp;

else

if (!sizeY)

sizeY = temp;

else

if (!colorMax)

colorMax = temp;

int delta = temp \* 10;

while (delta = delta / 10)// how many digit next

{

ptr++;

}

}

}

#pragma endregion

picture.seekg(ptr - line+1, picture.beg);//set position after maxcolor

myImage =new Image(type,colorMax,sizeX, sizeY);

for (int i = 0; i < sizeY; i++) {

for (int j = 0; j < sizeX; j++) {

picture.read(&myImage->canvas[i][j].r, 1);

picture.read(&myImage->canvas[i][j].g, 1);

picture.read(&myImage->canvas[i][j].b, 1);

//myImage->canvas[i][j].a = 0;

}

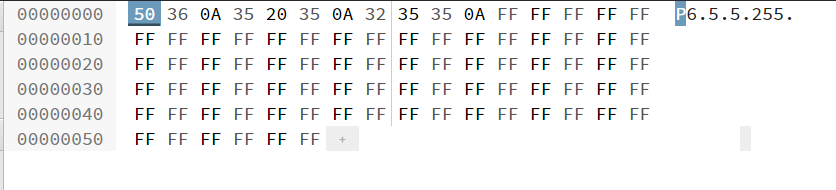
}

picture.close();

return true;

}

**Зміст отриманого PPM файлу для зображення розміру 5 х 5 пікселів чорного кольору.**



**Завдання 3**

*Реалізувати виділення пам’яті для зображень з ​пулу пам’яті фіксованого розміру. Розробити набір функцій для виділення пам’яті під зображення (​pool\_new, за заданими розмірами повертає покажчик на виділену пам’ять або nullptr, якщо це неможливо) та звільнення пам’яті (​pool\_delete, за заданим покажчиком звільняє пам’ять). Враховувати випадки, коли пам’яті недостатньо для створення нового зображення, або розмір зображення не вміщується в доступні ділянки пам’яті в пулі. В звіті навести блок-схеми і програмний код функцій pool\_new та pool\_delete.*

**Pool\_new:**

|  |  |
| --- | --- |
| Image \* Memory\_Pool::pool\_new(char \* type, char color, const int x, const int y)  {  for (size\_t i = 0; i < BLOCK\_COUNT; i++)  {  if (pool[i].ptr\_image == nullptr && sizeof(Image) + x\*y < BLOCK\_SIZE)  {  pool[i].ptr\_image = new(pool[i].allocate(sizeof(\*new Image(type, color, x, y)))) Image(type, color, x, y);  return pool[i].ptr\_image;  }  }  return nullptr;  }  void | D:\Studing\Пары\Програмування -2. Алгоритми і структури даних\Звіт\pool_new.png |

**Pool\_delete:**

|  |  |
| --- | --- |
| Memory\_Pool::pool\_delete(Image\* ptr)  {  for (size\_t i = 0; i < BLOCK\_COUNT; i++)  {  if (pool[i].ptr\_image == ptr)  {  pool[i].ptr\_image = nullptr;  pool[i].ptr\_last = pool[i].memory;  return;  }  };  } | **D:\Studing\Пары\Програмування -2. Алгоритми і структури даних\Звіт\pool_delete.png** |

**Makefile:**

all: solve

solve: main.o source.o

g++ main.o source.o -o main

main.o: main.cpp

g++ -c main.cpp

pool.o: source.cpp

g++ -c source.cpp

clean:

del -rf \*.o \*.exe

**Висновок:** під час виконання практикуму, я повторила типи даних, вивчені у минулому семестрі, такі комплексні типи даних: масиви, рядки, структури, об’єднання.