PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ
ESCOLA POLITÉCNICA
PROGRAMAÇÃO IMPERATIVA

O êxito está em ser bem-sucedido, e não em ter condições para o sucesso.
Qualquer terreno amplo tem condições para abrigar um palácio,
mas… onde estará o palácio se não o construírem lá.
Fernando Pessoa

Trabalho 2 – Contando objetos em imagens

A proposta deste trabalho é desenvolver um programa capaz de contar objetos em imagens pré-processadas que estão no formato PPM. O objetivo é empregar uma pilha encadeada além de utilizar conceitos de programação C como matrizes (arrays), estruturas (structs), alocação dinâmica de memória, estruturas encadeadas e leitura de arquivo binário.

Imagens digitais são normalmente armazenadas na forma de *bitmaps* (mapa de bits), em que cada bit ou grupo de bits representa um ponto elementar da imagem (pixel – abreviação de ***pic**ture **el**ement*). Assim, dependendo da densidade de bits por pixel, pode-se ter imagens monocromáticas, em escalas de cinza, ou com diferentes quantidades de cores. Esse mapa pode ser também entendido como uma matriz retangular, em que cada elemento representa um pixel.

Para representar imagens em cores, costuma-se usar 24 bits por pixel, permitindo até 16.777.216 (2^{24}) diferentes tonalidades. Esses 24 bits são agrupados em 3 bytes, que correspondem à intensidade de vermelho *R*, verde *G* e azul *B*, sendo 0 a intensidade mais baixa (ausência da cor) e 255 a mais alta (saturação total).

Formato de arquivo PPM

Imagens digitais podem ser armazenadas em diferentes formatos. O formato utilizado depende muito de sua finalidade. O formato JPEG, por exemplo, é mais adequado para fotografias, já que seu algoritmo de compressão é otimizado para isso. GIF costuma ser usado para desenhos e imagens pequenas, pois possui espectro limitado de cores, compressão sem perdas e permite animações.

O formato PPM (*portable pixel map*) tem como vantagem a simplicidade e fácil leitura. O arquivo de imagem PPM de 24 bits, formato que será usado nesse trabalho, é armazenado da seguinte forma:

- Um identificador do formato (caracteres "P6");
- Uma quebra de linha;
- As dimensões da imagem (largura e altura, nessa ordem);
- Uma quebra de linha;
- O valor máximo de intensidade (no caso desse trabalho sempre 255);
- Uma quebra de linha;
- Os valores intensidade R, G e B de cada pixel, a partir do canto superior esquerdo, da esquerda para a direita, de cima para baixo, até o canto inferior direito. Cada valor é armazenado como um byte, cujo valor corresponde à intensidade.

Por exemplo, uma imagem de tamanho 8 x 8, cujos pixels são todos brancos, é armazenada como:

```
P6
8 8
255
ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

Onde ÿ corresponde ao caractere cujo código ASCII é 255 em decimal (e FF em hexadecimal).

Algoritmo de crescimento de regiões

Para implementar o trabalho de contagem de objetos usaremos um algoritmo de segmentação por crescimento de regiões para eliminar cada região (objeto) conforme é feita a contagem.

Para isso, é preciso definir:

- A vizinhança de um pixel corresponde aos quatro pixels localizados imediatamente à direita, à esquerda, acima e abaixo, caso existam (cuidado com as bordas da imagem);
- Dois pixels pertencem a uma mesma região caso sejam vizinhos e atendam a uma determinada condição (no caso deste trabalho, seja um pixel vizinho de cor diferente do fundo, independente da cor).

Dadas as definições, o algoritmo pode ser descrito das seguinte forma:

- Define-se um ponto S, de coordenadas (x, y), denominado semente, a partir do qual se dará o crescimento de uma região;
- Marca-se de alguma forma que o pixel localizado na posição (x, y) já foi analisado;
- Verifica-se se cada um dos pixels vizinhos (x + 1, y), (x – 1, y), (x, y + 1), (x, y – 1) pertencem à mesma região de S. Cada vizinho que pertencer é considerado uma nova semente, e inicia-se outro crescimento a partir dele.

O algoritmo de crescimento de regiões pode ser implementado através de uma função recursiva (que é chamada para cada vizinho como semente) ou usando uma pilha dinâmica (na qual os pixels vizinhos são adicionados – *push*, para cada pixel analisado, passando-se para o próximo elemento da pilha em seguida - *pop*). Neste trabalho será necessário empregar uma pilha para armazenar as coordenadas do pixel vizinho, pois algumas imagens contendo objetos com muitos pixels gerariam um *stack overflow* caso a solução optasse pelo uso de função recursiva.

Para ilustrar o funcionamento do algoritmo e o emprego da pilha, o arquivo **PjBL2 - ExemploExecucao.pdf**, disponível no Canvas, apresenta um exemplo do algoritmo em funcionamento com representação visual dos vizinhos e da pilha.

Atividade

Desenvolver um programa que leia um arquivo no formato PPM de 24 bits e retorne a quantidade de objetos presentes na imagem. O programa deve:

- Pedir o nome de um arquivo de imagem;
- Caso o nome do arquivo seja inválido ou não seja um arquivo de imagem PPM, o programa deve acusar o erro e encerrar;
- Caso seja um arquivo válido (conforme especificações), a cor do primeiro pixel da imagem (canto superior esquerdo), deve ser considerado como cor de plano de fundo. Os objetos são as regiões da imagem cujos pixels possuem cor diferente do plano de fundo.

Para testar seu código, há uma série de 20 imagens de exemplo já pré-processadas, arquivo **PjBL2 – Imagens.zip**, cujos resultados são conhecidos, **PjBL2 – Contagens.pdf**, ambos disponíveis no Canvas.

Exemplos

- imagem 7 (7.ppm), contém 9 objetos, fundo branco:

- imagem 19 (19.ppm), contém 6 objetos (um deles é a grama), fundo azul: