

Zadanie A: Konwolucje (10 punktów)

Napisz program który dla danego rysunku i kernela policzy konwolucje. Jeśli podczas obliczeń należałoby wziąć punkt spoza rysunku, to proszę przyjąć zamiast niego najbliższy punkt który się na rysunku znajduje. Np:

- $(-1, 1) \rightarrow (0, 1)$
- $(-2, -2) \rightarrow (0, 0)$

Szczegóły zadania

Powyższe zadanie należy rozwiązać w ramach projektu nad którym pracujemy na ćwiczeniach.

Program powinien obsługiwać komendę: `kernel <nazwa> <x> <y> ...` który utworzy macierz/kernel o nazwie `<nazwa>` i wymiarach $x \times y$, indeksowany od wartości ujemnej $-\frac{x-1}{2}$ do wartości dodatniej $+\frac{x-1}{2}$, oraz analogicznie na osi y . Przyjmujemy, że pierwszy indeks x odpowiada za oś poziomą, a indeks y za oś pionową. Można założyć, że podane argumenty x i y zawsze będą nieparzyste.

Po wymienionych 3 stałych argumentach wystąpi xy liczb rzeczywistych, po x liczb w każdej linijce. Na przykład:

```
kernel K 3 3
1 0 -1
1 0 -1
1 0 -1
```

Kolejna wymagana komenda to: `convolveimg <out> <in> <kernel>` która policzy konwolucję rysunku `<in>` z zadanym kernelem `<kernel>`. Każdy kanał R, G i B powinien być liczony osobno. Nowy rysunek powinien być zapisany pod nazwą `<out>`.

Uwaga! Nazwy `<in>` i `<out>` to nie nazwy plików, tylko wewnętrzne nazwy w projekcie. Zapisywanie na dysk odbywa się tylko za pomocą komendy `save`.

Program powinien też obsługiwać komendy:

- `exit`
- `load`
- `save`
- `get`

Komendy te były wyjaśnione na ćwiczeniach. Jeśli nie określono inaczej, żadna z komend nie powinna wypisywać nic na standardowe wyjście. Jedyne co powinno się pojawiać to:

- Przy uruchomieniu programu komunikat `Program GK`
- Przy wywołaniu komendy `exit` informacja `Exiting with code <code>`, gdzie `<code>` to kod błędu przekazany do komendy `exit`.
- Przy wywołaniu komendy `get` powinno pojawić się R: `<czerwony>` G: `<zzielony>` B: `<niebieski>`, gdzie wartości `<czerwony>`, `<zzielony>` i `<niebieski>` to kolory wybranego piksela, podane jako liczba rzeczywista.

Jeśli potrzeba, dodatkowe komunikaty mogą pojawiać się na standardowym wyjściu błędów (`stderr`, `std::cerr`). Wyjście błędów będzie ignorowane przez Satori, ale nadmierne wypisywanie może spowolnić pracę programu.

Szczegóły techniczne

Rozwiążanie zadania, składające się z dowolnej liczby plików `.cpp` i `.h` proszę umieścić w katalogu `src` i całość skompresować do pliku `.zip`. Tak przygotowaną paczkę można wysyłać jako rozwiązanie.

Państwa projekt zostanie rozpakowany a następnie skompilowany komenda `make`, z `Makefile`-em takim jaki został udostępniony na pierwszych ćwiczeniach (po drobnej korekcie).

Jeżeli Państwa projekt wymaga innego pliku `Makefile` (np. chce Państwo dodatkowo kompilować pliki `.c` lub `.asm`), to proszę taki plik umieścić też w zip-ie, ale na zewnątrz katalogu `src` – tak aby był on widoczny bezpośrednio po rozpakowaniu. W ten sposób `Makefile` który dostarczamy zostanie nadpisany przez Państwa.

Używanie innych języków jest dopuszczalne, ale nie daję gwarancji ani wsparcia gdyby z jakiegoś powodu w systemie Satori Państwa projekt się nie kompilował lub działał źle.

Przykład

Dla następującego wejścia:

```
load tcs1.png img1
kernel smooth 3 3
0.0625 0.125 0.0625
0.125 0.25 0.125
0.0625 0.125 0.0625
kernel edge 3 3
1 0 -1
1 0 -1
1 0 -1
convolveimg img2 img1 smooth
convolveimg img3 img2 edge
save img3 tcse.png
get img3 0 0
get img3 724 800
get img3 1451 795
get img3 1453 802
exit 0
```

Przy założeniu, że plik `tcs1.png` jest następujący:



To poprawny program wypisze na standardowe wyjście:

```
Program GK
R: -0.0083333 G: -0.00661761 B: -0.00661767
R: 0.303186 G: 0.28799 B: 0.267402
R: 0.29902 G: 0.276471 B: 0.129657
R: 0.108578 G: 0.111765 B: 0.00906863
Exiting with code 0
```

(wartości liczb rzeczywistych mogą się nieznacznie różnić)

Zaś w bieżącym katalogu pojawi się plik `tcse.png` wyglądający następująco:

