Praktikum 1 zu Parallele Programmierung

Fabian Czappa

TECHNISCHE UNIVERSITÄT DARMSTADT

Wintersemester 2023/2024
31. Oktober 2023

Allgemeines

Es gibt teilweise Fußnoten, wenn Konzepte das erste Mal verwendet werden; dort können Sie sich weiter informieren. Die Standardsprache im Quellcode ist Englisch, weswegen auch die Typen/Methoden so benannt sind. *Außerdem gibt es Hinweise zu gutem Programmierstil in kursiver Schrift. Diese werden nicht bewertet.*

Bei jeder Teilaufgabe stehen die erreichbaren Punkte dabei. Dies ist so zu verstehen, dass für jeden Punkt ein vorgefertigter Test existiert; Sie erhalten den Punkt genau dann, wenn der Test durchläuft. Sollte der Test fehlschlagen oder gar nicht kompilieren, erhalten Sie dementsprechend keinen Punkt. Ein paar Tests stellen wir Ihnen bereits. Nennen sie **keine** Ihrer Dinge (z.B. Funktionen, Dateien, Klassen) mit `fabian` Bestandteil vom Namen. Wir kodieren alle unserer Testfälle, Testbilder, Testfunktionen, etc. zufällig mit `fabian` als zusätzlichen Schutz, damit sich diese nicht mit Ihren Dingen treffen.

Wegen der Restriktion von Moodle, nur Dateien bis 50 MB hochzuladen, sind die beiden Beispielbilder und deren verschlüsselten Varianten in getrennten zip-Archiven angegeben. Fügen Sie diese einfach in die passenden Ordner ein. Die Bilder sind über den gegebenen Code in der `main` verschlüsselt und dienen Ihnen als zusätzliche Hilfestellung.

ParProg
Nachname, Vorname: _____ Matrikelnummer: ⌴⌴⌴⌴⌴⌴⌴

Aufgabe 1: Bereitstellungen der notwendigen (Daten-)Klassen (Gesamt: 8 Punkte)

Die Aufgabe beschäftigt sich hauptsächlich mit den Klassen um die Daten zu halten. Die hier geforderten Funktionen werden in genau dieser Form getestet; eine solche ist durch die `Teletype` Schrift gekennzeichnet. Ihnen steht frei, weitere Funktionalität zu implementieren.

1a) Ein einfacher Pixel (1 Punkt)

Erstellen Sie die Klasse `Pixel` in der Datei /source/image/pixel.h. Die Klasse soll zur späteren Verwendung mit einem `template` für Klassen versehen werden[1]. Über diesen Typparameter ('template' übersetzt) wird später der Typ übergeben, welcher für die Speicherung der einzelnen Farbkanäle verwendet wird. Stellen Sie auch einen `Standardkonstruktor` bereit.

Stellen Sie sicher, dass jedes Attribut stets initialisiert ist. Sie können beispielsweise Standardwerte direkt in Deklaration einfügen.
Normalerweise sind die privaten Felder einer Klasse unter den öffentlichen Feldern, da erstere 'nicht einsehbar' sind.
Stellen Sie den Typparameter als dependent type innerhalb der Klasse bereit.[2]

1b) Farbkanäle eines Pixels (1 Punkt)

Erstellen Sie einen `Konstruktor` für die Klasse `Pixel`, welcher drei Werte nimmt: Die Intensität der Farbkanäle für `blau`, `grün` und `rot`. Stellen sie pro Kanal eine Methode bereit, welche die gespeicherte Intensität zurück gibt. Die Namen dieser Methoden sind `get_blue_channel`, `get_green_channel` und `get_red_channel`; sie sollten alle `const` sein. Stellen Sie das statische Attribut `channel_order` vom Typ `ChannelOrder` (weiter oben bereits definiert, nicht erweitern!) bereit, welches angibt, in welcher Reihenfolge Sie die Argumente im Konstruktor entgegen nehmen (wichtig für den korrekten Test).

Wenn eine Methode einen Parameter nicht ändert, sollten Sie den als `const` klassifizieren – dies vermeidet Programmierfehler.[3]
Wenn eine Methode das aktuelle Objekt nicht ändert, sollten Sie diese als `const` klassifizieren. Dadurch sind die Methoden aufrufbar, auch wenn das Objekt als solches selbst `const` ist (siehe vorherigen Hinweis).[4]
Wenn eine Methode keine Exception werfen kann, sollten Sie diese als `noexcept` markieren.[5]
Wenn eine Methode einen Wert zurück gibt und ein logischer Fehler wäre, diesen nicht zu speichern, können Sie diese mit `[[nodiscard]]` markieren. Wird der Wert nicht beim Aufruf gespeichert, erzeugt dies eine Warnung.[6]

1c) Gleichheit von Pixeln (1 Punkt)

Stellen Sie den Gleichheitsoperator für `Pixel` bereit.[7] Zwei Pixel sind genau dann gleich, wenn sie die gleiche Intensität in den jeweiligen Farbkanälen haben. Der Operator sollte `const` sein.

Sie können alternativ auch den Spaceship-Operator bereitstellen: `<=>`[8]
Dieser bzw. andere können auch benutzt werden, um den Gleichheitsoperator implizit zu definieren.

[1] https://en.cppreference.com/w/cpp/language/class
[2] https://en.cppreference.com/w/cpp/language/type_alias
[3] https://en.cppreference.com/w/cpp/language/cv
[4] https://en.cppreference.com/w/cpp/language/member_functions
[5] https://en.cppreference.com/w/cpp/language/noexcept_spec
[6] https://en.cppreference.com/w/cpp/language/attributes/nodiscard
[7] https://en.cppreference.com/w/cpp/language/operators
[8] https://stackoverflow.com/questions/47466358/what-is-the-spaceship-three-way-comparison-operator-in-c

1d) Bitmap als Klasse (2 Punkte)

Erstellen Sie die Klasse `BitmapImage` in der Datei /source/image/bitmap_image.h. Diese sollte keinen expliziten Typ-/Klassenparameter benötigen. Stellen Sie auch einen `Konstruktor` bereit, welcher als erstes die Höhe und als zweites die Breite des Bildes akzeptiert. Diese sollten beide größer als 0 und kleiner oder gleich 8192 sein; sollte dies nicht der Fall sein, schmeißen Sie eine `std::exception`. Stellen Sie auch die Funktionen `get_height` und `get_width` bereit, welche das jeweilige Attribut zurück geben; sie sollten beide `const` sein. Stellen Sie als dependent type den Typ `BitmapPixel` bereit. Dieser soll gleich dem Typ `Pixel<std::uint8_t>` sein.

Die Reihenfolge der Attribute in einer Klasse beeinflusst die Ausführungszeit des Programms (minimal), neben anderen Dingen.[9]
Machen Sie sich nicht die Mühe, selbst über die Lebenszeit der Daten nachzudenken. Nutzen Sie dafür Standardcontainer.

1e) Ändern der Pixel (2 Punkte)

Stellen Sie die Funktionen `set_pixel` (drei Argumente) und `get_pixel` (zwei Argumente) für die Klasse `BitmapImage` bereit. Beide akzeptieren als erstes und zweites Argument die x- bzw. y-Position des jeweiligen Pixels (Reihenfolge der beiden Parameter Ihnen überlassen, indizieren Sie angefangen mit 0). `set_pixel` akzeptiert als drittes Argument den neuen Wert. `get_pixel` gibt den Pixel an der angegebenen Position zurück. Sollte die Position außerhalb des Bildes liegen, schmeißen Sie eine `std::exception`. Stellen Sie den Typ, welchen Sie für x- und y-Position verwenden, als dependent type `index_type` bereit.

1f) Transponieren eines Bildes (1 Punkt)

Stellen Sie die Funktion `transpose` für die Klasse `BitmapImage` bereit; diese sollte `const` sein. Sie gibt ein neues Bild zurück, für welches gilt: original[i][j] = transponiert[j][i].

In C++ ist es oftmals die beste Idee, eine neu erstelle lokale Variable als Kopie zurück zu geben.[10]

[9] `https://stackoverflow.com/questions/892767/optimizing-member-variable-order-in-c`
[10] `https://en.cppreference.com/w/cpp/language/copy_elision`

Aufgabe 2: Lesen und Schreiben von Bildern (Gesamt: 4 Punkte)

Die Aufgabe beschäftigt sich mit dem Einlesen und Ausgeben der Bilder. Die hier geforderten Funktionen werden in genau dieser Form getestet; eine solche ist durch die `Teletype` Schrift gekennzeichnet. Ihnen steht frei, weitere Funktionalität zu implementieren.

2a) Laden eines Bitmap (2 Punkte)

Erstellen Sie die Klasse `ImageParser` in /source/io/image_parser.h. Stellen Sie die statische Methode `read_bitmap` bereit, welche ein `BitmapImage` zurückgibt, und als Argument ein `std::filesystem::path` akzeptiert. Geben Sie das spezifizierte Bild geladen zurück.

Stellen Sie sicher, dass Sie korrekt die Höhe und Breite des Bildes verarbeiten können, ebenso wie `bfOffBits`.[11] Bzgl. der anderen Attribute können Sie annehmen, dass diese gleich der von uns gestellten Bilder sind. Sollte etwas schief gehen (z.B. der übergebene Pfad nicht auf eine gültige Datei zeigen), schmeißen Sie eine `std::exception`.

C++ bietet die Möglichkeit viele verschiedene Typen zu verwenden. Nutzen Sie für jeden Zweck einen möglichst spezifischen (z.B. `std::filesystem::path`), damit Sie Programmierfehler direkt umgehen können.
Wir stellen Ihnen einige Bilder direkt zur Verfügung, welche Sie ohne Probleme einlesen können sollten.

2b) Speichern eines Bitmap (2 Punkte)

Stellen Sie die statische Methode `write_bitmap` bereit, welche zwei Argumente akzeptiert: Als erstes ein `std::filesystem::path`, als zweites ein `BitmapImage`. Speichern Sie das Bild an die spezifizierte Stelle.

Sollte etwas schief gehen (z.B. der übergebene Pfad nicht auf eine gültige Datei zeigen), schmeißen Sie eine `std::exception`.

Wenn Sie ein Bild laden, speichern und wieder laden, sollte zwischen den geladenen Versionen kein Unterschied bestehen. Gleiches gilt für speichern, laden, nochmal speichern; die gespeicherten Dateien sollten gleich sein.

[11] `https://de.wikipedia.org/wiki/Windows_Bitmap`

Aufgabe 3: Verschlüsseln von Bildern (Gesamt: 8 Punkte)

Die Aufgabe beschäftigt sich mit dem Verschlüsseln von Bildern mit **F**abian's **E**ncryption **S**cheme. Die hier geforderten Funktionen werden in genau dieser Form getestet; eine solche ist durch die `Teletype` Schrift gekennzeichnet. Ihnen steht frei, weitere Funktionalität zu implementieren.

Konzeptionelle Einführung:
Das Bild wird in **Blöcke** aufgeteilt, jeder Block ist 3x1=3 Pixel hoch und 3x16=48 Pixel breit. Eine Ansammlung von 1x16 Pixeln nennen wir **Reihe**. Der Zusammenschluss von Pixeln zu Reihen ist in Abb. 1 schematisch abgebildet; der Zusammenschluss von Reihen zu Blöcken in Abb. 2. Die Aufteilung startet bei Pixel (x=0, y=0) im Bild (und setzt sich dementsprechend fort bei (x=48, y=0) bzw. (x=0, y=3)).

Sie können davon ausgehen, dass es keine unvollständigen Reihen gibt; es kann aber durchaus unvollständige Blöcke geben. In einem solchen Fall müssen Sie die Randbehandlung von weiter unten durchführen.

Aufteilung der Blöcke:
Für jeden Block ist der **vorherigen Block** definiert (dieser bezieht sich immer auf das gleiche Bild). Hat ein Block den Startpixel (x=48+k, y) (es gab schon einen Block vorher auf gleicher Höhe), so ist der vorherige Block der mit Startpixel (x=k, y) (48 Pixel nach vorne geschoben). Eine Darstellung dessen ist in Abb. 3 zu sehen.

Hat ein Block den Startpixel (x=0, y), so ist der vorherige Block durch den Schlüssel zu berechnen. Jede Reihe in dem Block ist dabei gleich. Der Schlüssel enthält 48 Byte an Daten, nummeriert als [0], [1], [2], ... Die Reihe wird nun gebildet als 16 Pixel mit den Werten (b: [2], g: [1], r: [0]), (b: [5], g: [4], r: [3]), ...

Verschlüsselung der Blöcke:
Der Block an Stelle (x, y) wird nun mit dem vorherigen Block verschlüsselt und beim verschlüsselten Bild an Stelle (x, y) gespeichert. Achten Sie hier ggf. auf die Randbehandlung!

Die Berechnung des verschlüsselten Blocks (**e**) auf Basis des aktuellen und vorherigen Blocks (aktuell, **a**; vorherig, **v**) läuft reihenweise ab, siehe Abb. 4. Für jede Reihe (x=0, 1, 2; y = 0, 1, 2) wird die jeweilige Reihe von **a** mit der jeweiligen Reihe von **v** kombiniert. Die resultierende Reihe wird mit allen bereits verschlüsselten Reihen kombiniert. Dabei gilt die Reihenfolge (mit + als Zeichen für die Kombination):
(((((**a**[x, y] + **v**[x, y]) + **e**[0, 0]) + **e**[1, 0]) + **e**[2, 0]) + **e**[0, 1]) + ... Abb. 5 zeigt einige Musterberechnungen.

Kombinieren von Reihen:
Das Kombinieren von zwei Reihen erfolgt für jeden Pixel getrennt. Gegeben zwei Pixel $p_1 = (b_1, g_1, r_1)$ und $p_2 = (b_2, g_2, r_2)$ ist der kombinierte Pixel $p := (b_1 \oplus b_2, g_1 \oplus g_2, r_1 \oplus r_2)$, wobei $\oplus$ für XOR steht.

Randbehandlung:
Es können folgende Fälle eintreten (siehe Abb. 6):

- Der Block liegt vollständig im Bild: Indizieren Sie die Reihen normal.
- Der Block liegt bzgl. seiner Breite vollständig im Bild, bzgl. seiner Höhe nicht: Indizieren Sie die Reihen normal. Nehmen Sie fehlende Reihen als 0 an.
- Der Block liegt bzgl. seiner Höhe vollständig im Bild, bzgl. seiner Breite nicht: Indizieren Sie die Reihen transponiert. Nehmen Sie fehlende Reihen als 0 an.
- Der Block liegt weder bzgl. seiner Höhe noch seiner Breite vollständig im Bild. Es liegen zwei Reihen breit im Bild: Verschieben Sie beim Indizieren die fehlende Spalte ans Ende. Nehmen Sie fehlende Reihen als 0 an.
- Der Block liegt weder bzgl. seiner Höhe noch seiner Breite vollständig im Bild. Es liegt nur eine Reihe breit im Bild: Transponieren Sie die Indizes und überspringen Sie fehlende Spalten (eine Kombination aus Varianten 2 und 4).

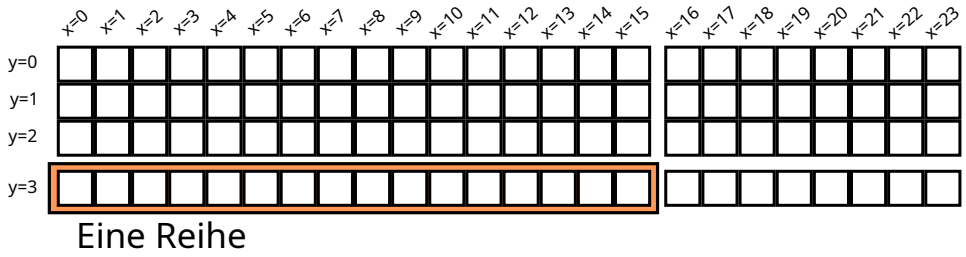

Abbildung 1: Eine Reihe ist hervorgehoben: Sie beinhaltet die Pixel an den Stellen x=0, ..., 15 und y=3. Angedeutet sind weitere Reihen.

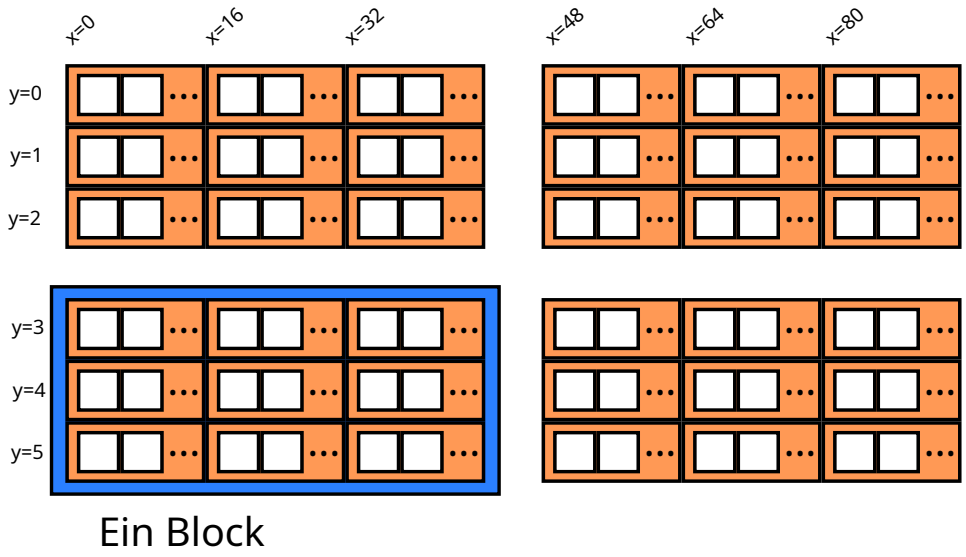

Abbildung 2: Ein Block ist hervorgehoben: Er beinhaltet 3x3 Reihen, welche insgesamt von x=0 bis x=47 und y=3 bis y=5 gehen. Andere Reihen sind auch zu Blöcken gruppiert.

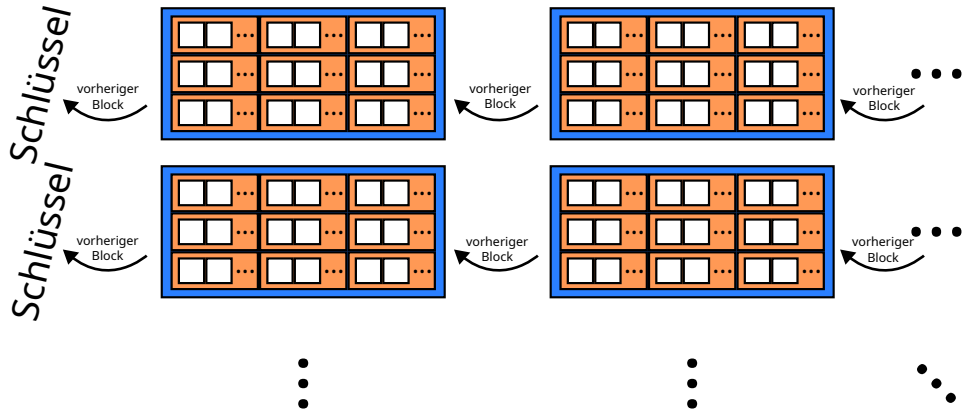

Abbildung 3: Das Verhältnis von Blöcken zu den jeweils vorherigen Blöcken ist durch Pfeile gekennzeichnet. Die ersten Blöcke in einer (Block-)Zeile berechnen diesen auf Basis des Schlüssels. Schematisch gehen die Blöcke nach rechts und unten weiter.

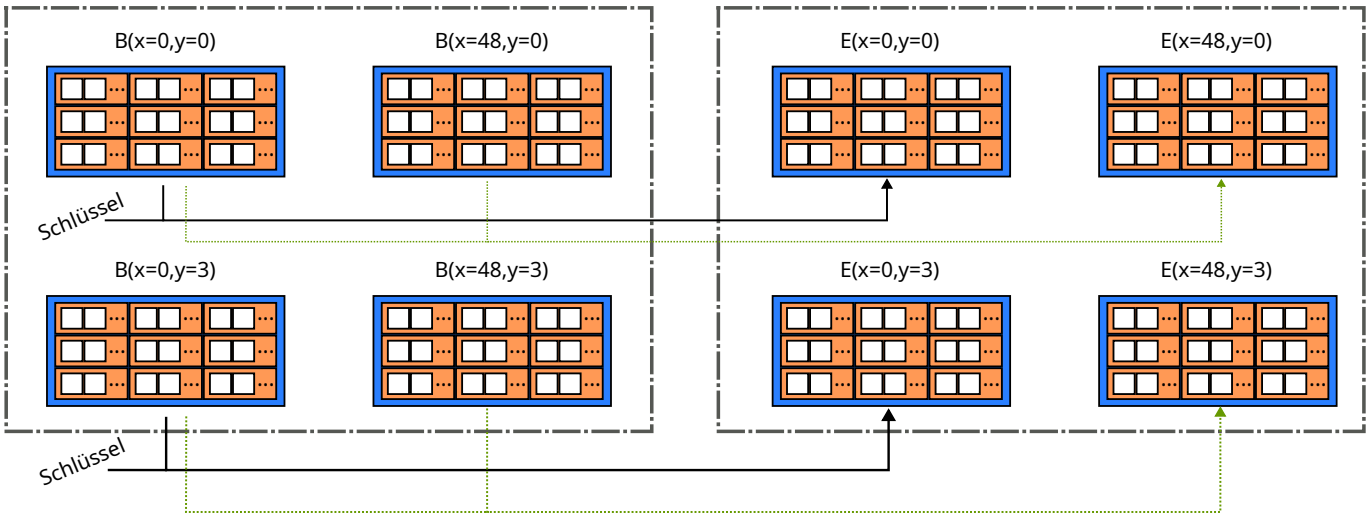

Abbildung 4: Eine schematische Betrachtung, wie ein Bild (links, “B”) verschlüsselt wird (rechts, “E”). In Klammern ist jeweils der erste Pixel des Blocks angegeben. Die erste Berechnung pro Reihe ist durch einen durchgehenden, schwarzen Pfeil angedeutet. Die zweite Berechnung pro Reihe ist durch einen gestrichelten, grünen Pfeil angedeutet. Weitere Berechnungen sind der Deutlichkeit halber ausgelassen.

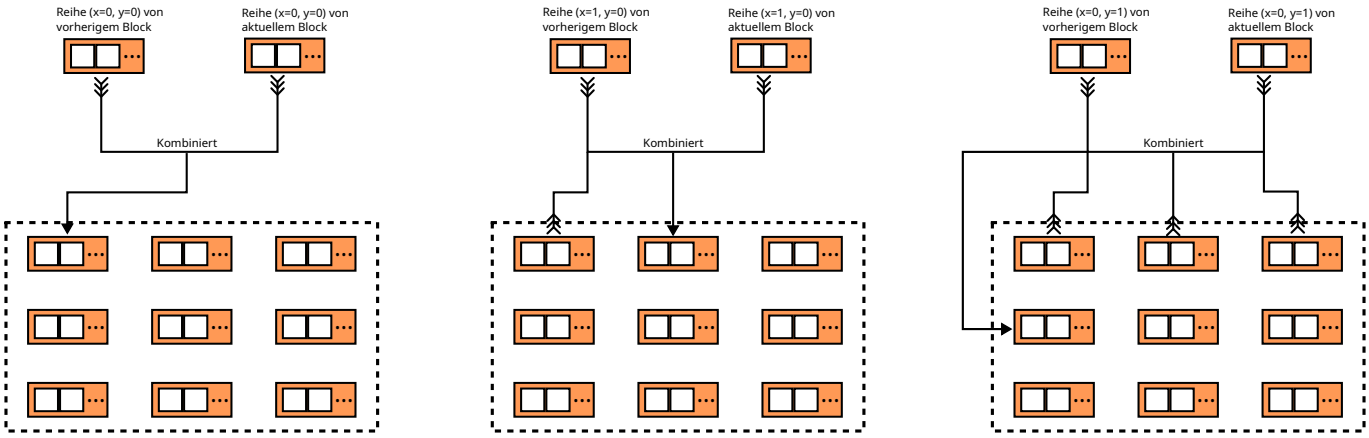

Abbildung 5: Berechnungsschema für ausgewählte Reihen im verschlüsselten Block (unten, gestrichelt gruppiert). Es sind jeweils die beiden benutzen Reihen vom aktuellen und vom vorherigen Block abgebildet; die anderen sind ausgelassen. Die Reihen sind innerhalb des Blocks von links nach rechts (x) und oben nach unten (y) nummeriert. Reihen, welche nicht an der Stelle (0, 0) stehen, hängen auch von den bereits vorher verschlüsselten Reihen ab.

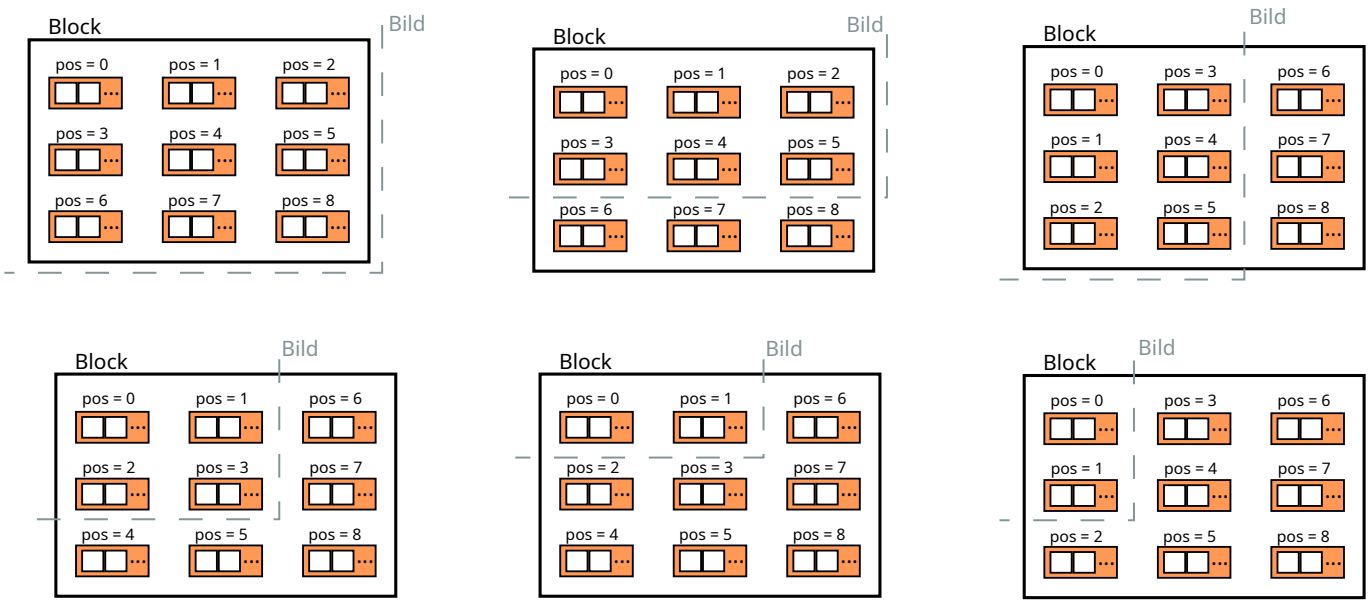

Abbildung 6: Die Randbehandlung für verschiedene Varianten, wie ein Block über die Grenzen des Bildes herausragen kann. “pos = ...” gibt dabei an, in welcher Reihenfolge die Reihen innerhalb des Blocks verschlüsselt werden. Reihen außerhalb des Bildes können Sie als 0 annehmen.

3a) Typen (1 Punkt)

Erstellen Sie die Klasse FES in /source/encryption/FES.h. Stellen Sie als dependent type die Typen `key_type` (soll gleich dem Type `Key::key_type` sein), `row_type` (soll gleich dem Typen `std::array<BitmapImage::BitmapPixel, 16>` sein) und `block_type` (soll gleich dem Typen `std::array<std::array<row_type, 3>, 3>` sein) bereit.

3b) Konvertierung des Schlüssels (1 Punkt)

Stellen Sie die statischen Methoden `convert_key_to_block` und `convert_key_to_row` bereit, welche jeweils einen `key_type` akzeptieren und einen `block_type` bzw. `row_type` zurück geben. Die Methoden berechnen eine Reihe bzw. einen Block auf Basis des übergebenen Schlüssels wie vorgegeben.

3c) Kombinieren von Reihen (1 Punkt)

Stellen Sie die statische Methode `combine_rows` bereit, welche zwei Parameter von Typ `row_type` akzeptiert und einen `row_type` zurück gibt. Die Methode kombiniert zwei Reihen wie vorgegeben.

3d) Verschlüsseln von Blöcken (3 Punkte)

Stellen Sie die statische Methode `encrypt_block` bereit, welche zwei Parameter von Typ `block_type` akzeptiert (als erstes den aktuellen, als zweites den vorherigen) und einen `block_type` zurück gibt. Die Methode verschlüsselt den Block nach dem angegebenen Schema.

3e) Verschlüsseln von Bildern (2 Punkte)

Stellen Sie die statische Methode `encrypt` bereit, welche zwei Parameter akzeptiert: Als erstes einen von Typ `BitmapImage`, als zweites einen vom Typ `key_type`. Sie gibt ein `BitmapImage` zurück.

Nachname, Vorname: _____ Matrikelnummer: ␣␣␣␣␣␣␣␣

Hinweise zur Abgabe

Hier ein paar zusätzliche Formalitäten zu Ihrer Abgabe. Falls Sie diese nicht beachten, ist es möglich, dass Sie keine Punkte für eine oder alle Aufgaben erhalten.

Angabe der Autorschaft

Geben Sie in der Datei /source/authors.h an, wer von Ihnen an welcher Teilaufgabe mitgearbeitet hat. Mehrere Namen pro Teilaufgabe sind ok, trennen Sie diese z.B. mit Komma. Wir fordern eine Teilnahme von allen! Sollten Sie nicht an dem Praktikum mitgearbeitet haben (basierend auf den Angaben), erhalten Sie 0 Punkte auf das ganze Praktikum.

Hochladen des Quellcodes

Zippen Sie folgende Dateien und laden Sie das zip-Archiv in Moodle hoch:

- Den Ordner /cmake/
- Den Ordner /source/
- Die Datei CMakeLists.txt

Sie sind alle für die korrekte Abgabe verantwortlich. Zu spät eingereichte Dateien, Dateien mit fehlenden Lösungen, etc., liegen in allein Ihrer Verantwortung.

Hinweise zum Lichtenberg

Der Lichtenberg-Hochleistungsrechner ist aufgeteilt in sogenannte Loginknoten und Rechenknoten, wobei sich alle das gleiche Dateisystem teilen. Erstere sind mit **ssh** und **scp** zu erreichen und werden benutzt um Rechenaufgaben für letztere zu kreieren. Wenn Sie eine sinnvolle Linux-Distribution verwenden, kann /bin/bash nativ **ssh** und Sie können das Dateisystem des Lichtenberg direkt in Ihrem Explorer (z.B. Nemo) einbinden. Wenn Sie Windows benutzen, empfiehlt sich das Terminal bzw. PuTTy für **ssh** und WinSCP für **scp**.

Sobald Sie eingeloggt sind, sollten Sie zusätzliche Softwarepakete laden. Das erreichen Sie beispielsweise mit **module load git/2.40.0 cmake/3.26.1 gcc/11.2.0 cuda/11.8 boost/1.81.0**. Kopieren Sie die Dateien auf den Hochleistungsrechner und kompilieren Sie dort das Programm.

Wenn Sie eine Rechenaufgabe lösen möchten, müssen Sie dafür SLURM benutzen. Effektiv führen Sie den Befehl **sbatch <filename>** aus, wobei <filename> hier eine Datei ist, die, sobald Ihr Programm Zeit und Resourcen zugeteilt bekommen hat, auf den Rechenknoten ausgeführt wird. Ein beispielhaftes Skript sieht so aus:

```
#!/bin/bash

#SBATCH -J parprog
#SBATCH -e /home/kurse/kurs00072/<TUID>/stderr/stderr.parprog.%j.txt
#SBATCH -o /home/kurse/kurs00072/<TUID>/stdout/stdout.parprog.%j.txt
#SBATCH -C avx512
#SBATCH -n 1
#SBATCH --mem-per-cpu=1024
#SBATCH --time=5
#SBATCH --cpus-per-task=1
#SBATCH -A kurs00072
#SBATCH -p kurs00072
#SBATCH --reservation=kurs00072

module load git/2.40.0 cmake/3.26.1 gcc/11.2.0 cuda/11.8 boost/1.81.0

echo "This is job $SLURM_JOB_ID"
```

Der Kurs ist ab dem 01. November 2023 freigeschaltet. Alle Informationen erhalten Sie auch hier: `https://www.hrz.tu-darmstadt.de/hlr/nutzung_hlr/zugang_hlr/loginknoten_hlr/index.de.jsp`

Zusätzlich: Optimierung

Wir laden Sie ein diverse Optimierungen in Ihren Code einzubauen! Es steht Ihnen frei, beispielsweise `encrypt_block_fast` zu implementieren und innerhalb `encrypt` zu verwenden, ebenso andere Typen, solange Sie die Vorgaben der Aufgaben nicht verletzen.

Wir stellen Ihnen eine Vergleichsmöglichkeit in Moodle bereit, in der wir auch unseren Lösungsvorschlag “mitkämpfen” lassen. Nutzen Sie zur Leistungsmessung bitten den Lichtenberg, ihr Laptop hat bereits einen intrinsischen Nachteil.