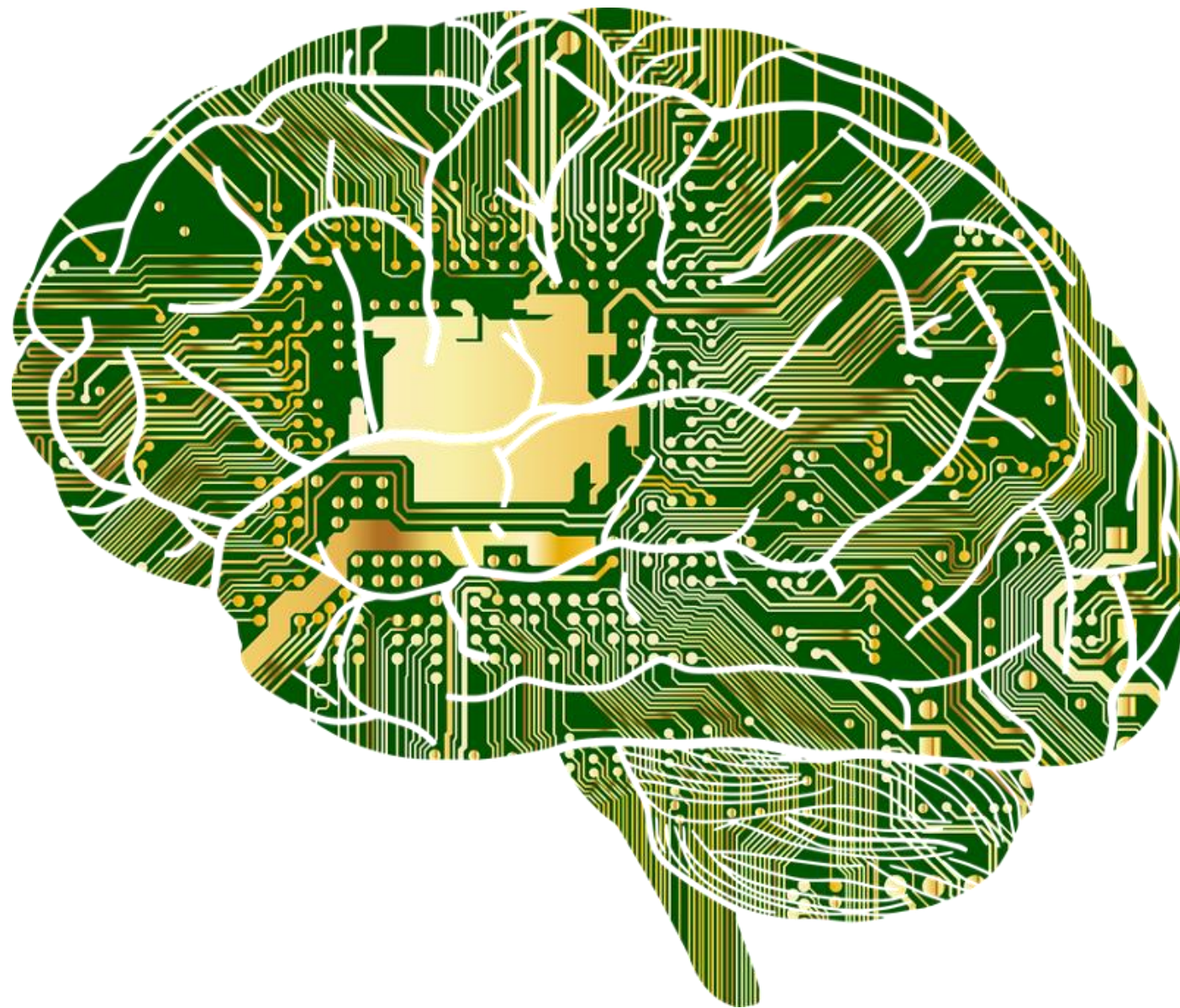


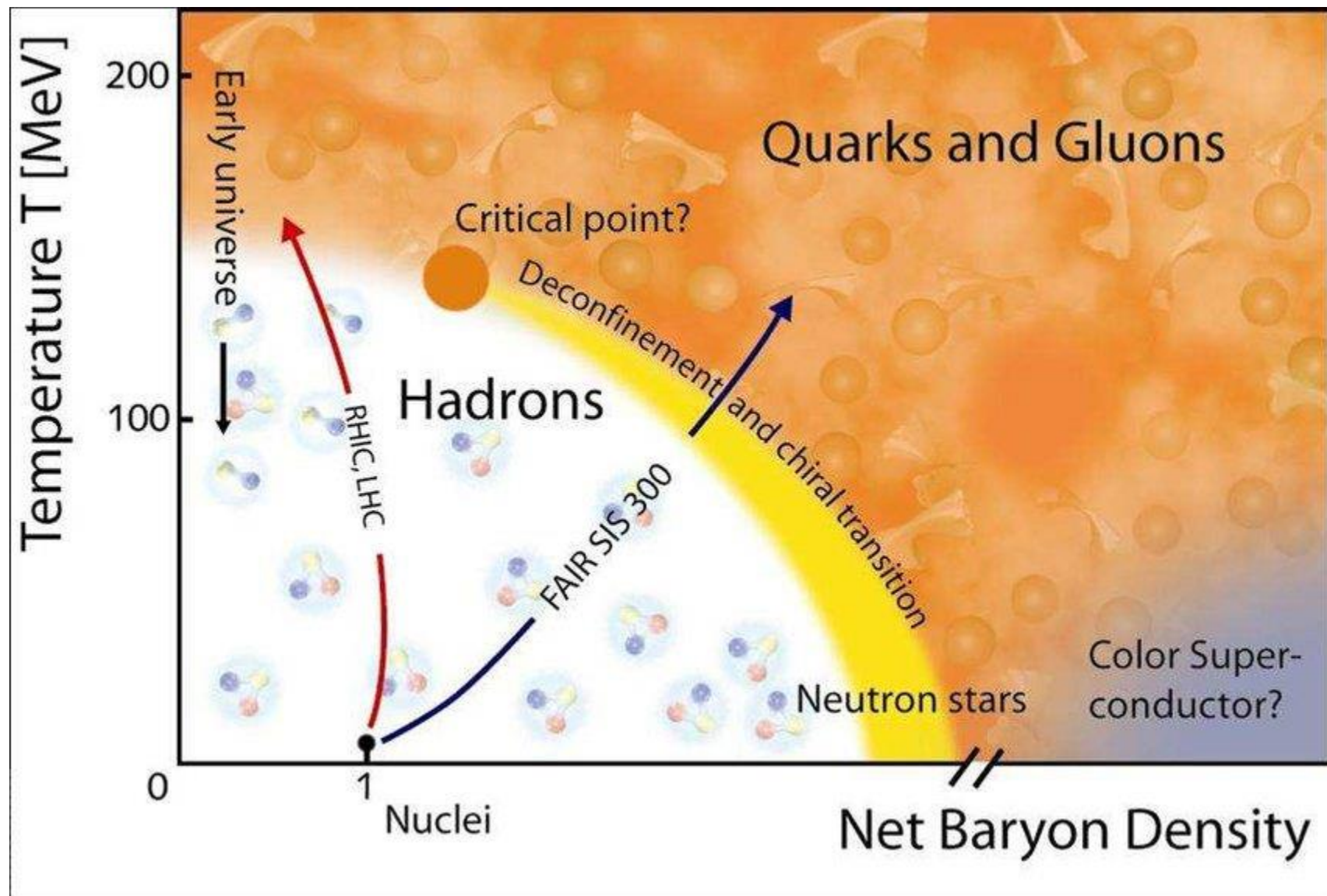


Grundlagen der Programmierung

Praktikum

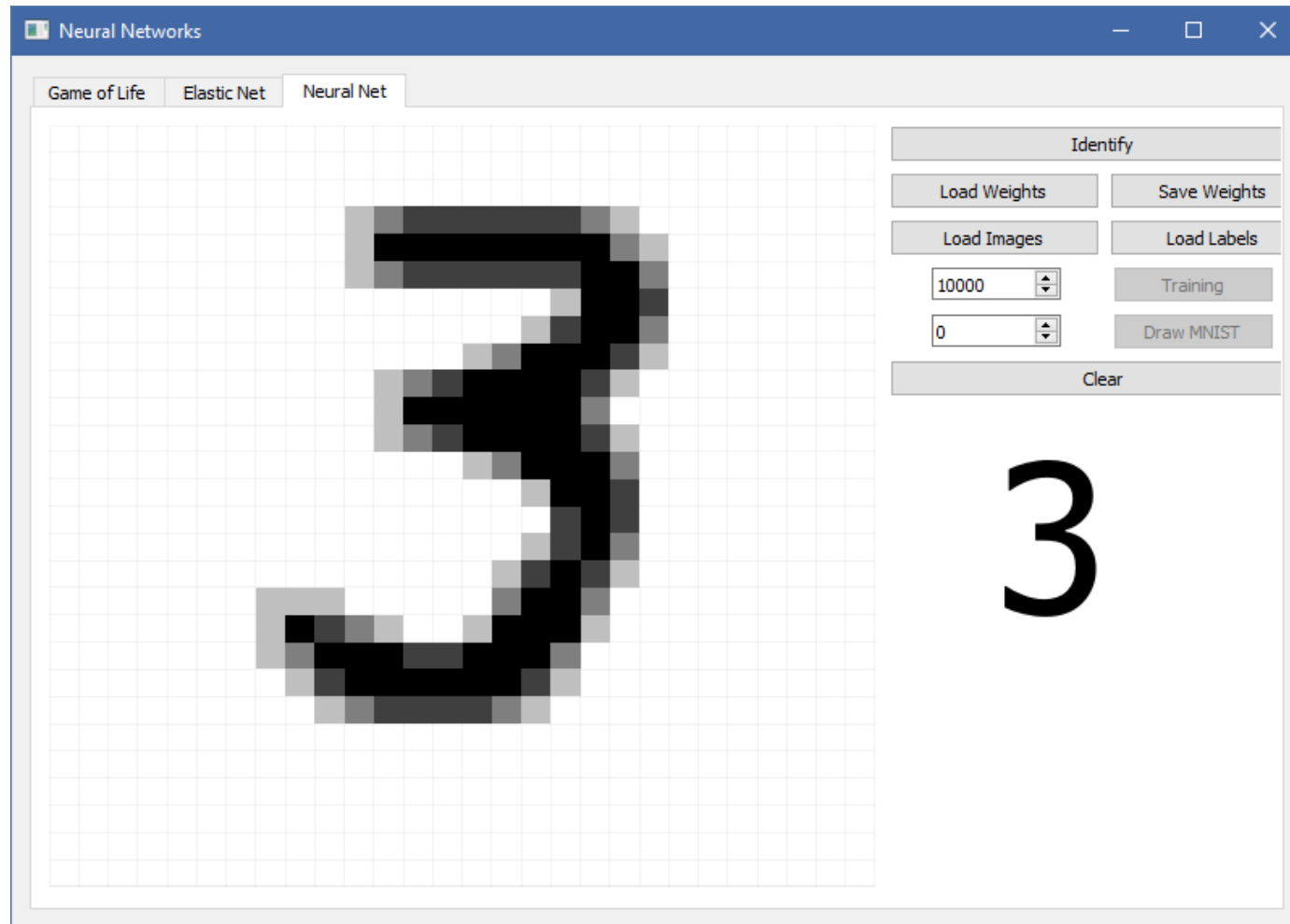
Artemiy Belousov, Ivan Kisel, Grigory Kozlov, Martin Parnet







Ziel



Prof. Dr. Ivan Kisel

- Hochleistungsrechnungen
- Teilchenphysik
- Hochenergiephysik
- Hochleistungsrechnerarchitektur
Praktikum
- Sprechstunde Mi 16-17 Fischerräume
- FIAS (Frankfurt Institute for Advanced Studies)
- GSI Helmholtzzentrum für Schwerionenforschung

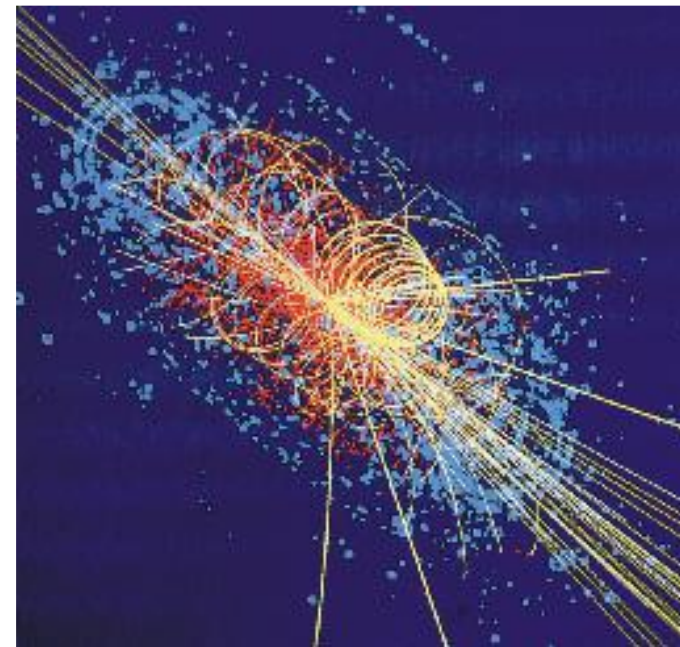
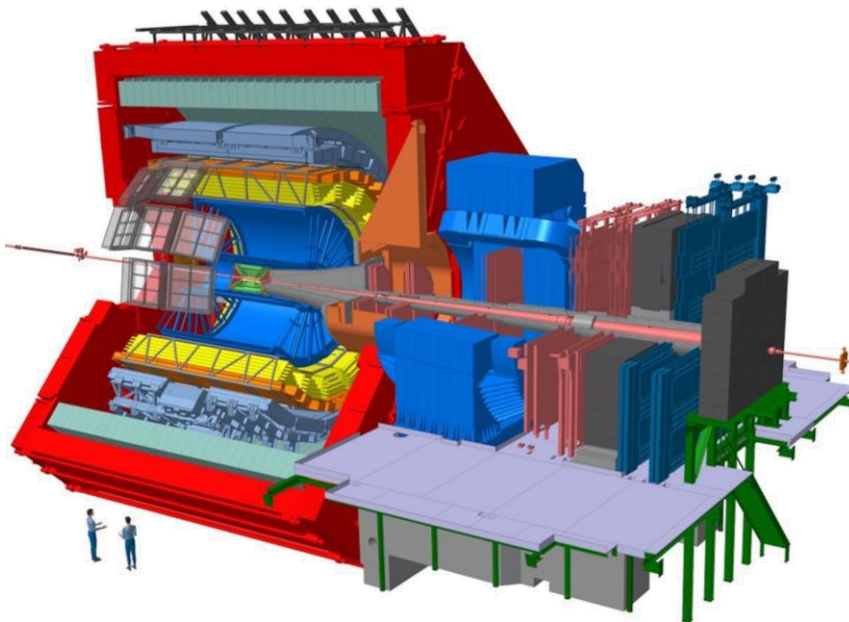


A Large Ion Collider Experiment

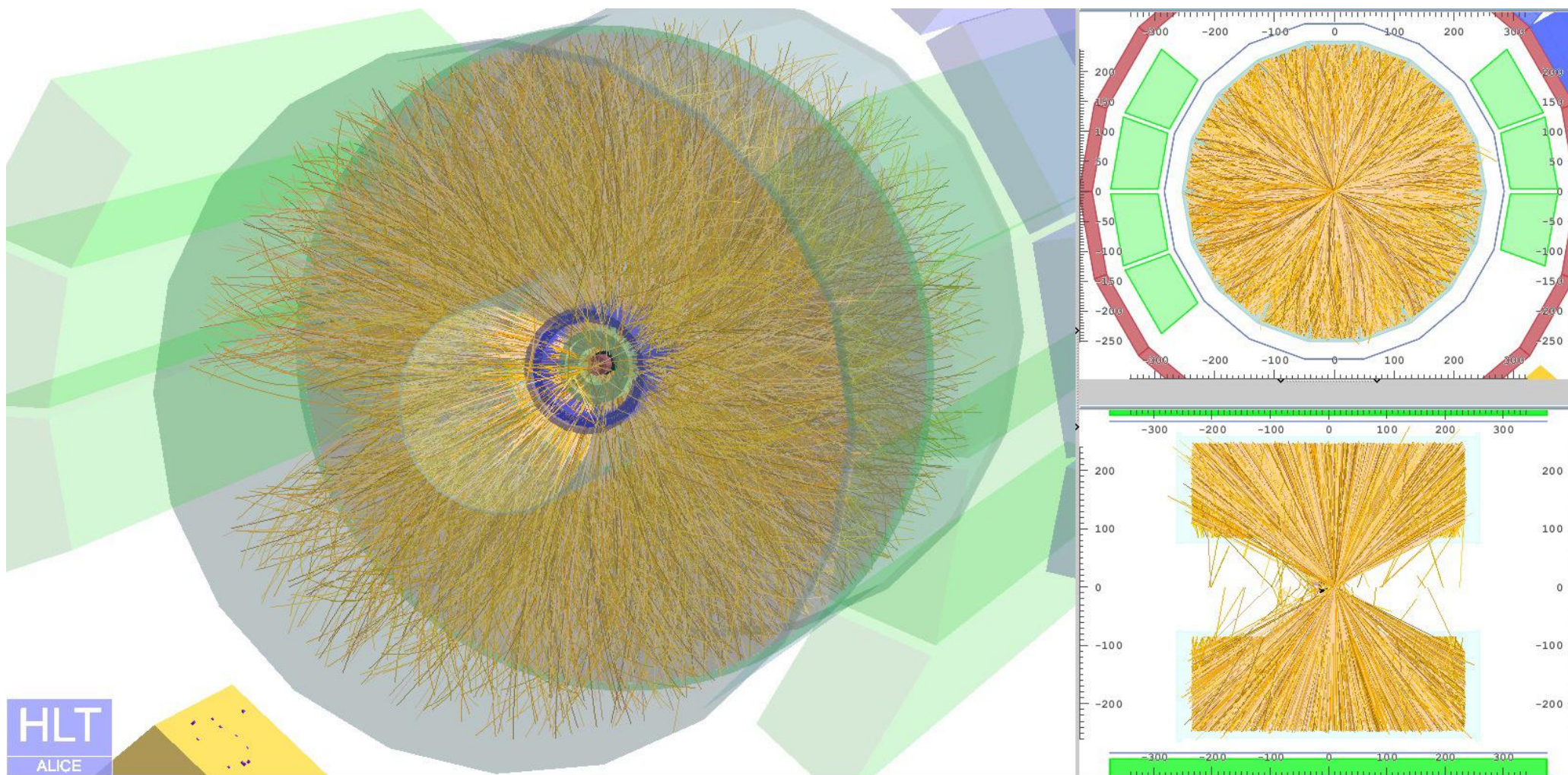


A Large Ion Collider Experiment

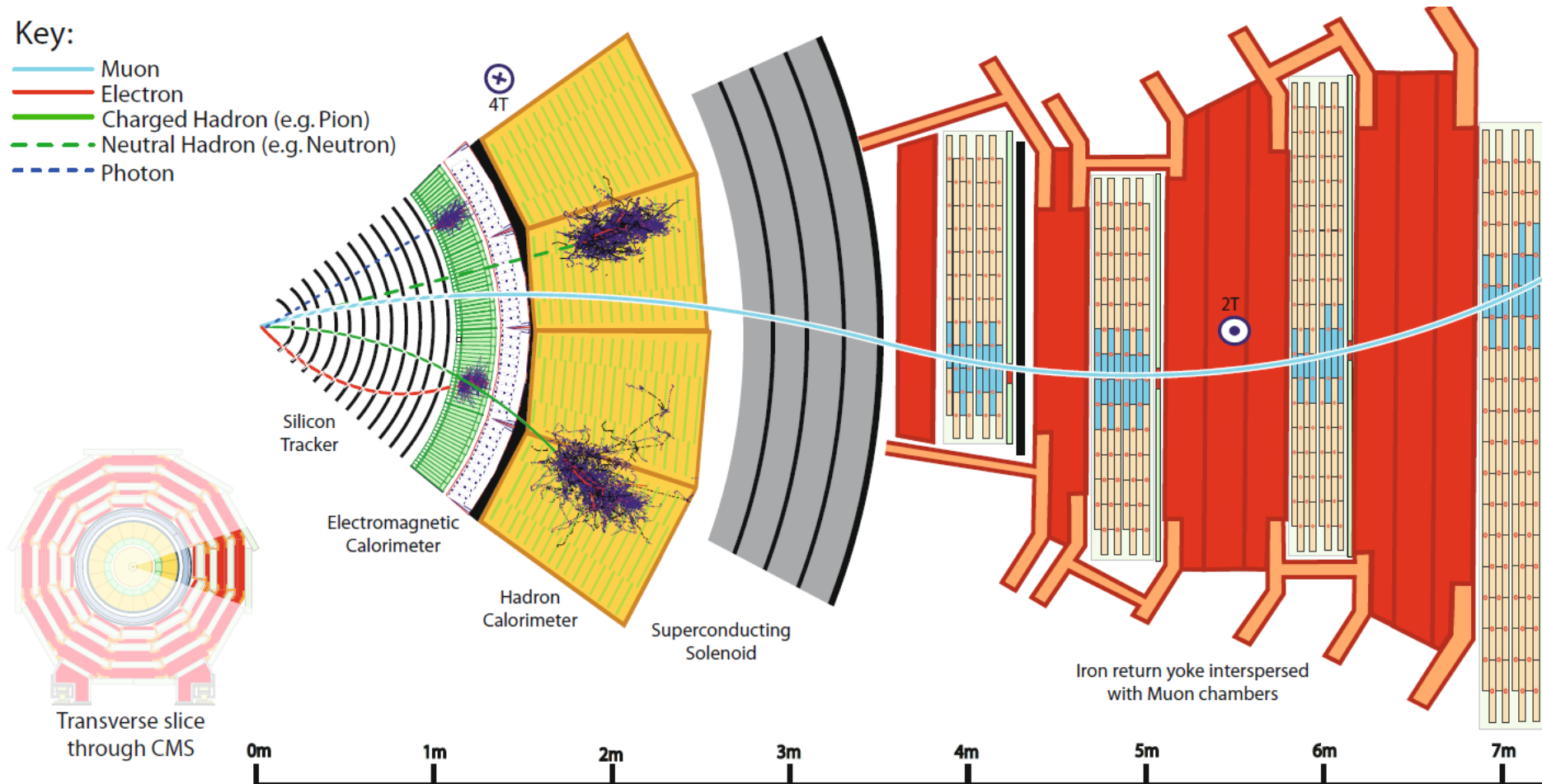
- 500 Kollisionen in einer Sekunde
- Wie sieht die Trajektorie eines Teilchen aus?
- 600 Millionen elektrische Auslesekanäle

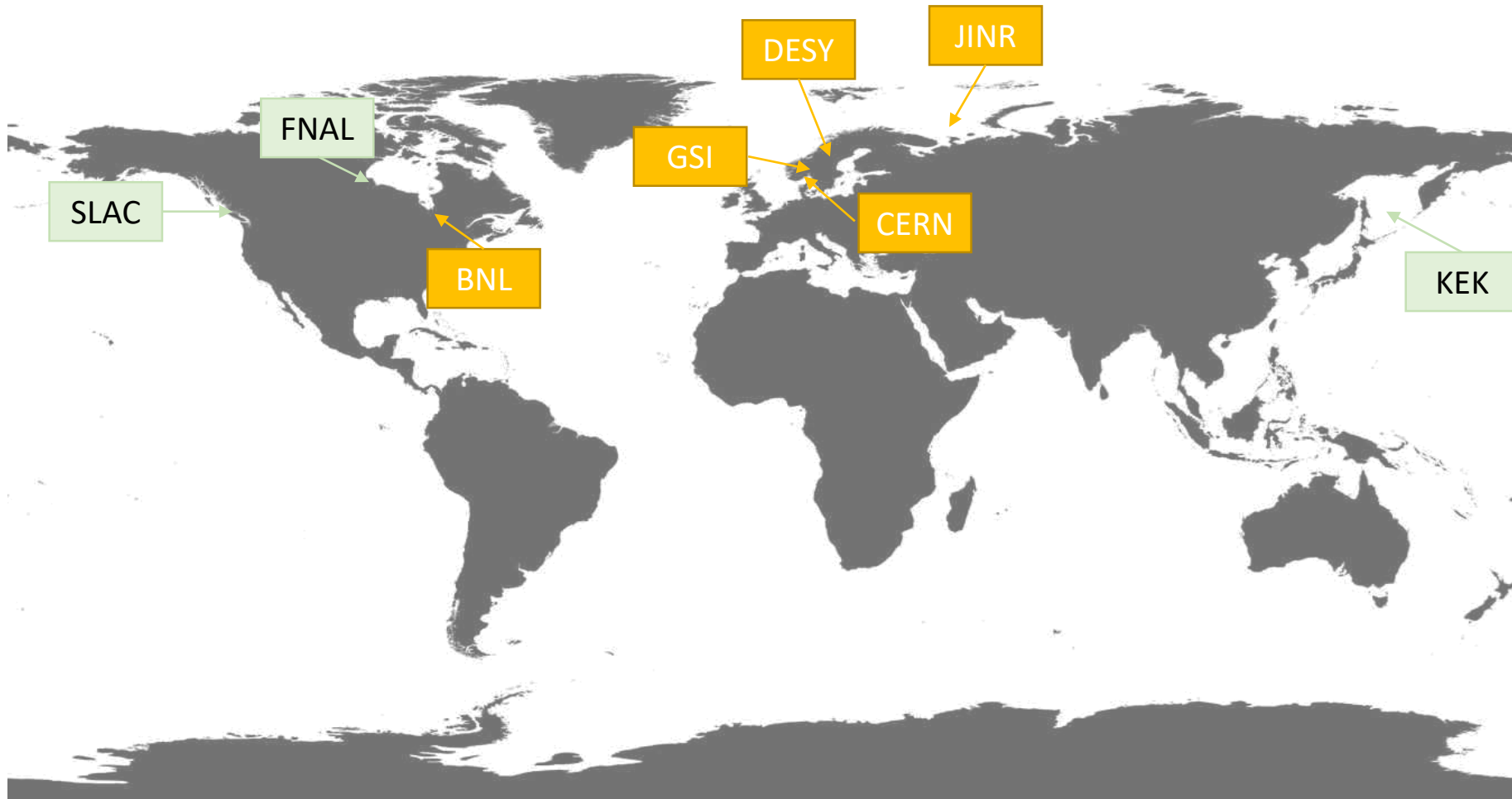


A Large Ion Collider Experiment



Teilchendetektoren







Organisatorisches

Organisation

- Vertiefung der Programmierkenntnisse
- Größeres Projekt
- Teamarbeit
- Komplexere Aufgaben lösen
- Entwicklungsumgebungen nutzen
- Versionsverwaltung kennen lernen
- Dokumentation

Organisation

- Zeitaufwand 8 CP (60h + 180h)
- Bachelor Informatik und Bioinformatik
 - Nicht benotet
- Lehramt Haupt und Realschule, Förderschule
 - $\frac{3}{4}$ Zeitaufwand, aber benotet
- Lehramt Gymnasium (SPoL)
 - 6 CP, $\frac{3}{4}$ Zeitaufwand, benotet

Voraussetzung: (Prüfungsordnung beachten!)

- 2011 Erfolgreicher Abschluss von B-PRG-1 oder B-PRG-2
- 2019 Erfolgreicher Abschluss von B-EPI, B-PDB, B-PPDC oder B-MOD

Teamarbeit

- 3 – 4 Personen
- Gemischte Teams
- Registrierung bald möglich
- Nicht Tutorien übergreifend
- Probleme in Teams lösen
- Individuelle Leistung muss deutlich abgrenzbar und bewertbar sein

Tutorien



	Wochentag	Zeit	Raum	Tutor
1	Montag	12:00 – 15:00	Fischerräume	Robin Lakos
2	Montag	15:00 – 18:00	Fischerräume	Roman Staehle
3			RBI 26	Lisa Hornung
4	Dienstag	14:00 – 17:00	Fischerräume	Devansh Rastogi
5			RBI 26	Samiyah Farooq
6	Mittwoch	12:00 – 15:00	Fischerräume	Abdelkhalek Kamkoun
7			RBI 26	Nadir Mokhtari
8	Donnerstag	14:00 – 17:00	Fischerräume	Enes Hafut Erdag
9			RBI 26	Sami Chaabi
10	Freitag	14:00 – 17:00	Fischerräume	Martin Parnet
11			RBI 26	Emal Jar

Tutorien beginnen ab 22.10.2019

Fischerräume: RBI 22, 23, 25

Kriterien für das Bestehen

- Kriterien werden für jeden Milestone festgelegt
- Milestones werden gewichtet nach Bearbeitungszeit und Schwierigkeitsgrad
- Tutor_innen nehmen Milestones in einem persönlichen Gespräch ab
- Bestanden, wenn mindestens 60% erreicht
 - 30 % mindestens pro Milestone
- Lehramt-Studierende erhalten benotete Scheine

< 60 %	5.0	<i>entspricht 42% der Gesamtpunktzahl</i>
> 98 %	0.7	<i>entspricht 72% der Gesamtpunktzahl</i>
Linear interpoliert		<i>für reine Lehramtgruppen</i>

Modalitäten

- Regelmäßige Teilnahme an den Besprechungen
- Abgabe muss fristgerecht erfolgen
- Keine Möglichkeit zur Nachbesserung
- Dokumentation muss vorhanden sein
- Quelltext vollständig kommentiert
- Autorenschaft Nachvollziehbar
- Readme
 - Bekannte Fehler
 - Zeiterfassung
- Abgabe wurde vorgeführt

Plagiate



- Austausch von Programmcode zwischen Projektgruppen ist **nicht** gestattet
- Verwendung von Quellcode aus dem Internet nur mit Quellenangabe
- Offensichtliche Plagiate werden als Betrugsversuch gewertet

	Abgabe
Einführung in C++ <ul style="list-style-type: none">• Game of Life• Visuelle Kryptographie• QT	13.11.2019 (4 Wochen)
<hr/>	
Quark Gluonen Plasma (QGP)	
• Fully Connected Neural Network <ul style="list-style-type: none">• Training• Auswertung	11.12.2019 (4 Wochen)
• Convolutional Neural Network <ul style="list-style-type: none">• Vergleich mit FC NN	22.01.2020 (3 Wochen)

WS19.20: Praktikum-Grundlagen der
Programmierung [PRG-PR]

WS19.20: Praktiku

Startseite

Einschreibung

Tutorien

Tutorium 1

Tutorium 2

Tutorium 3

Tutorium 4

Tutorium 5

Tutorium 6

Tutorium 7

Tutorium 8

Tutorium 9

Tutorium 10

Tutorium 11

Materialien

Plenartermine

Milestones

Bewertung

Forum

Literatur

E-Mail an Teilnehm

E-Mail an Tutoren

E-Mail an Veranstal

Gruppen

Forum

Sie dürfen Beiträge lesen, Beiträge schreiben, und Moderieren.

Das Forum ist noch leer.



Neues Thema

Abonnieren

auch bei
Materialien



Milestone 1

Abgabefrist

23:59 - 13.11.2019

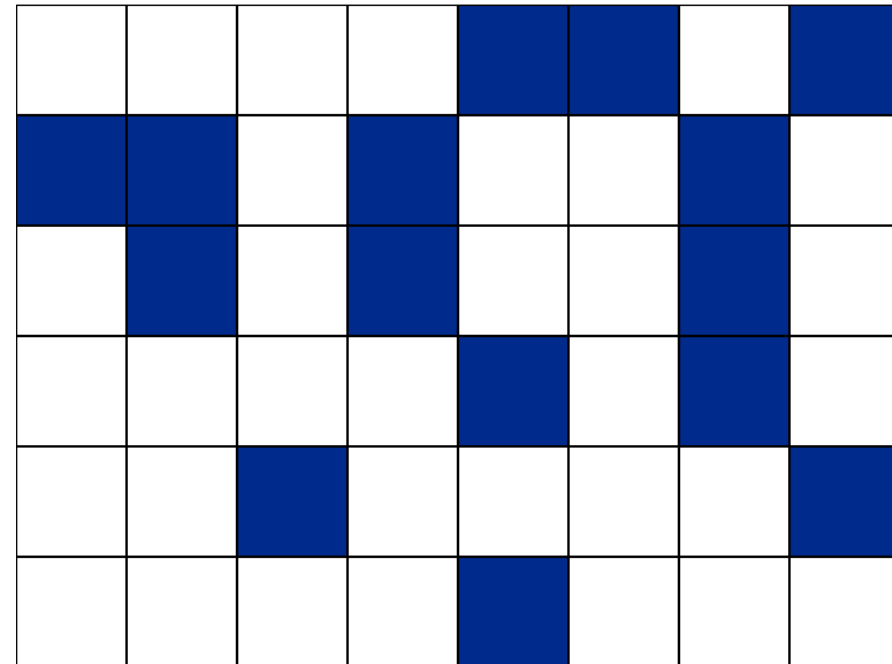
Zellulärer Automat

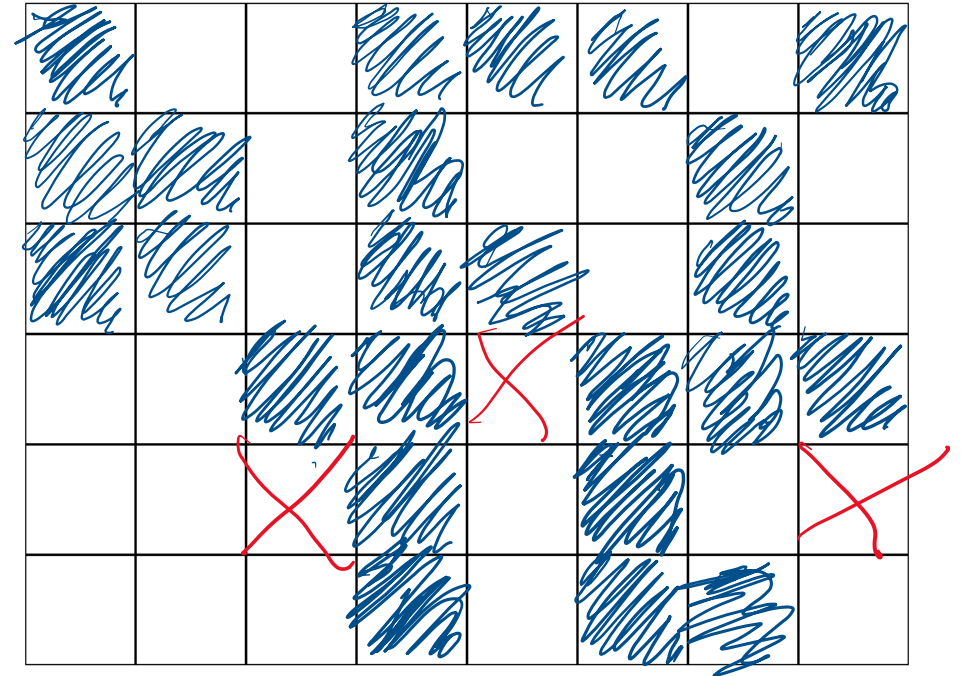
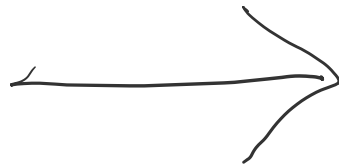
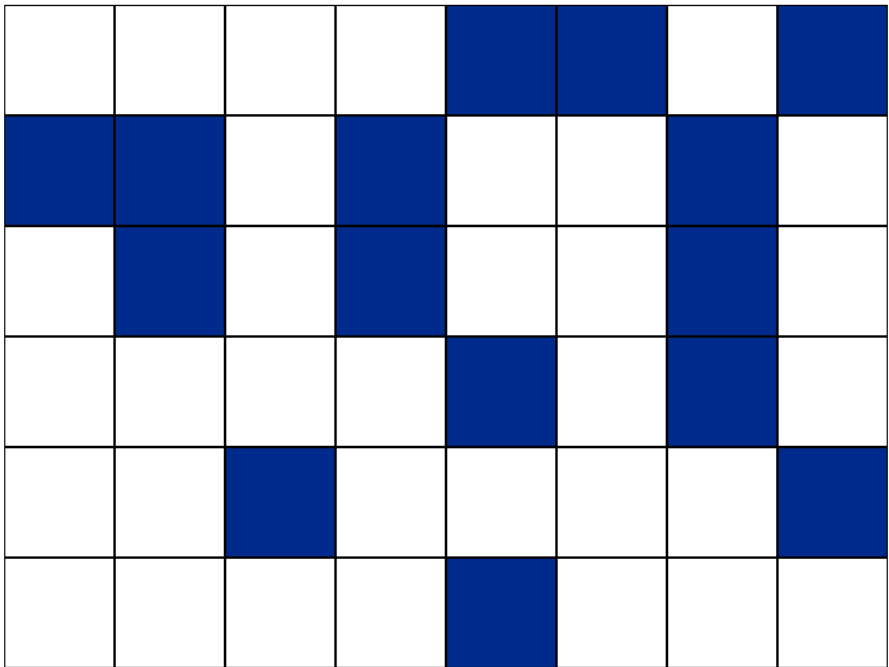
- Mathematisches Modell für dynamisches Modell
 - Biologie, Organismen, Künstliches Leben
- Diskretisierung
 - Raum: Gitter
 - Zeit: Zeitschritte
 - Zustände: endlich
- Transformationsvorschrift für Folgezustand
 - Lokal beschränkt

Game of Life



- Feld mit $n \cdot m$ Zellen $n, m \in \mathbb{N}$
- Zustand der Zellen lebend oder tot
- Transformationsvorschrift für Moore Nachbarschaft
- Folgezustand hängt nur vom aktuellen Zustand ab

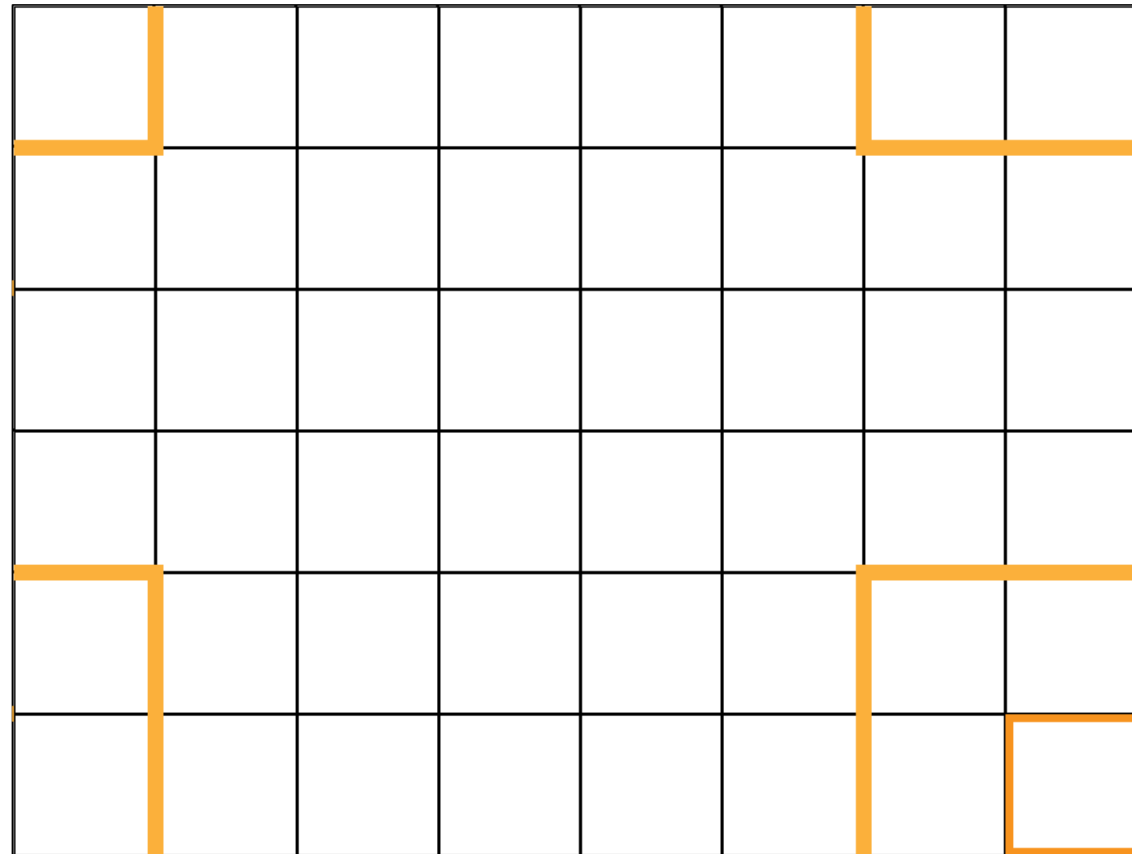




Game of Life

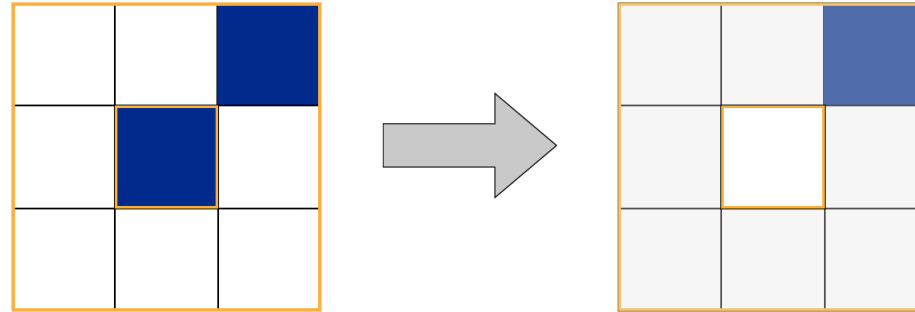


- 1 - Moore Nachbarschaft

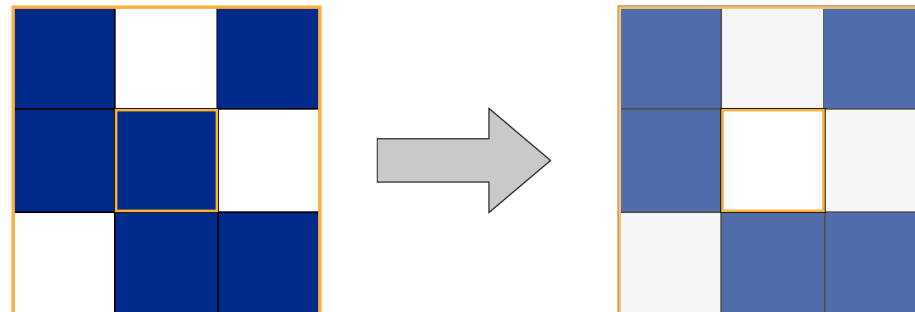


Game of Life – Transformation

- Tod – Vereinsamung (E)
 - Lebende Zelle mit weniger als 2 Nachbarn stirbt

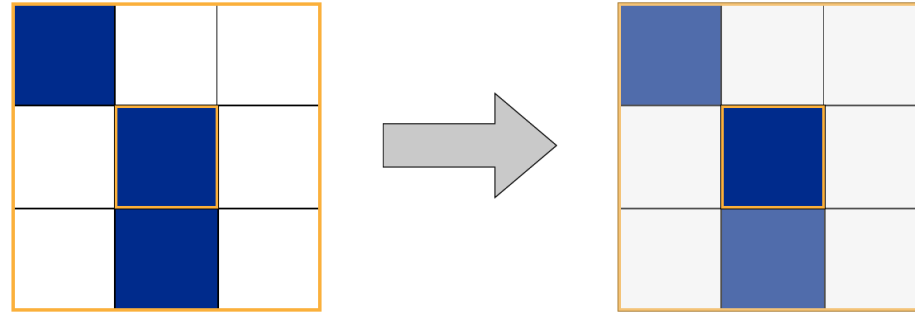


- Tod – Verhungern (V)
 - Lebende Zelle mit mehr als 3 Nachbarn stirbt

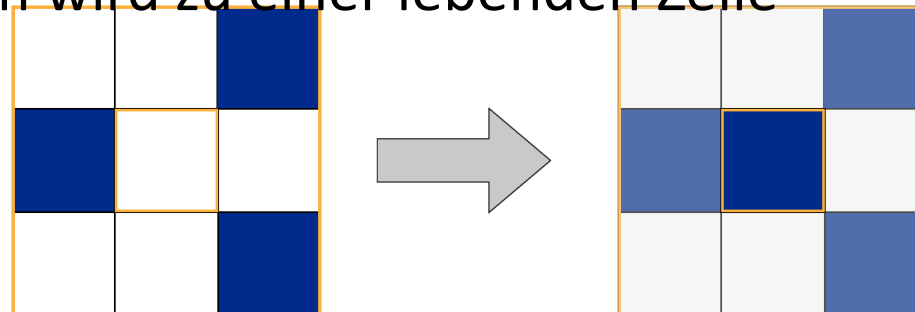


Game of Life – Transformation

- Überleben (u)
 - Lebende Zelle mit 2 oder 3 Nachbarn verbleibt lebend

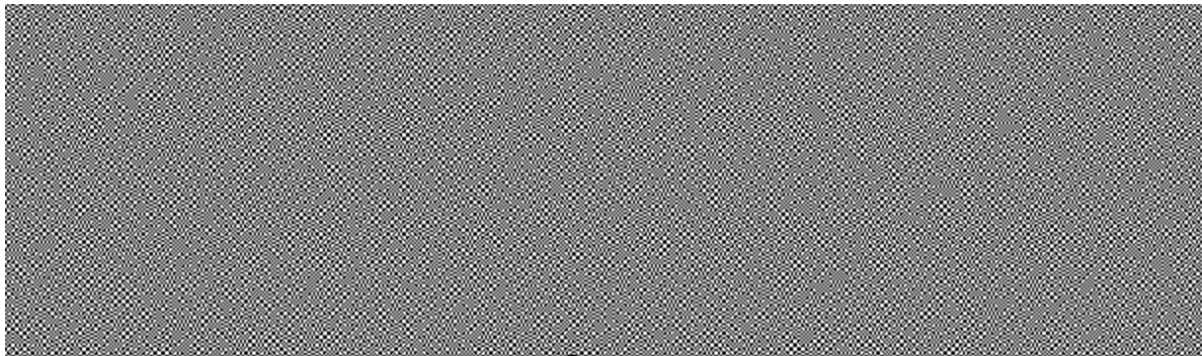


- Geburt (G)
 - Tote Zelle mit genau 3 Nachbarn wird zu einer lebenden Zelle

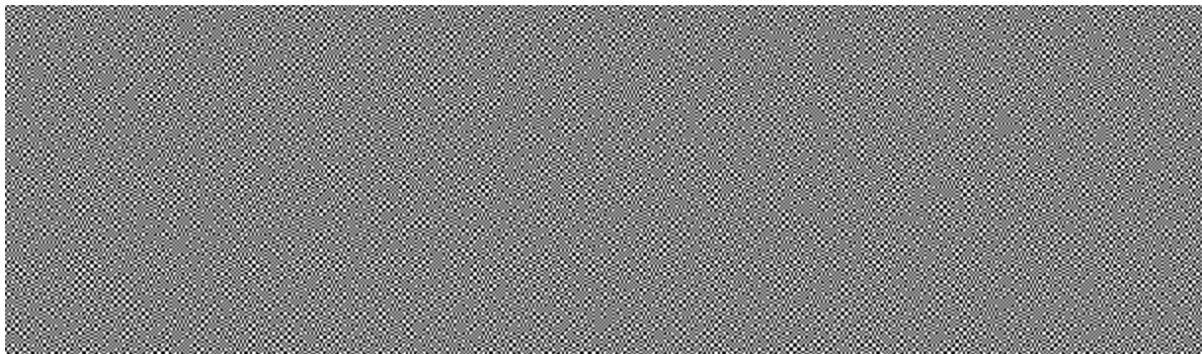


Visuelle Kryptographie

- Entschlüsselung durch optische Wahrnehmung
- Verschlüsselung mit One-Time-Pad; Schlüssel:



Informationen

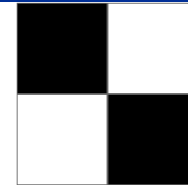
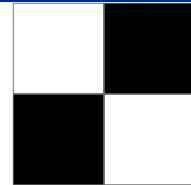


Ergebnis

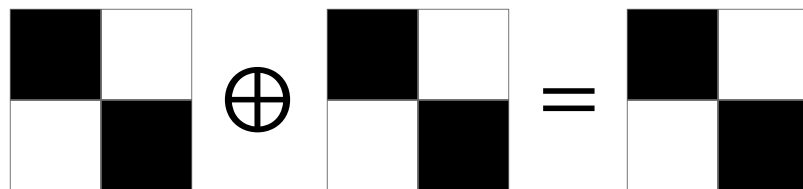
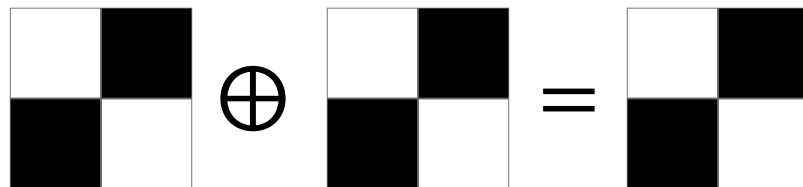


Visuelle Kryptographie (Umkehrung)

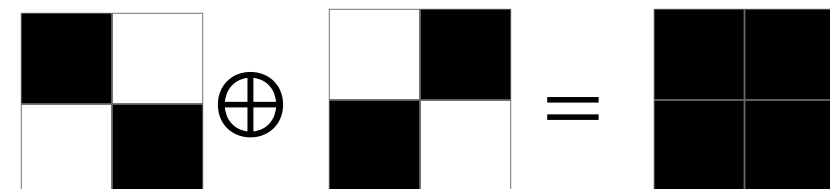
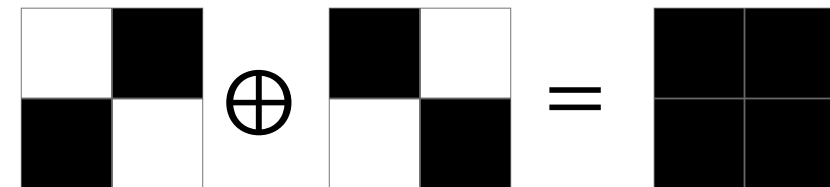
- Elemente A
- Schlüssel alle Bildpunkte rein zufällig aus A und B
- Operationen



Verschlüsselt Schlüssel Weißes Pixel



Verschlüsselt Schlüssel Schwarzes Pixel





Visuelle Kryptographie (Anwendung)

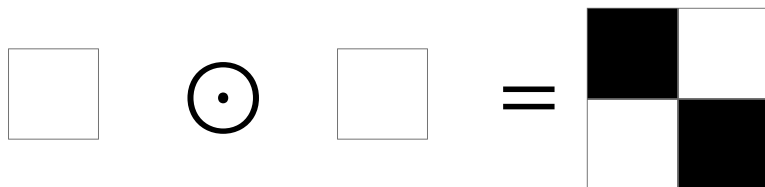
- Elemente A
- Aus einem Bildpunkt werden im neuen Bild 4
- Operationen



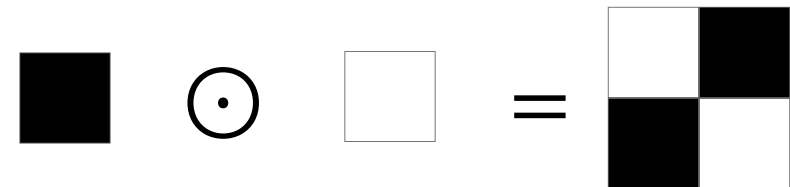
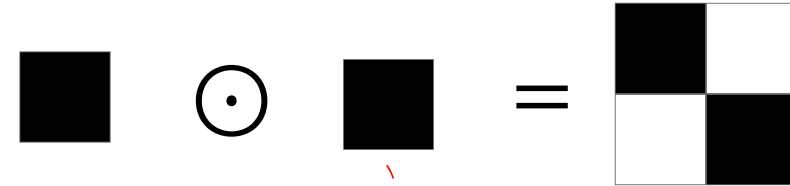
Visuelle Kryptographie (alternative)

- Elemente A  
- Leichter zu Programmieren
- Operationen

Bildpunkt Schlüssel Verschlüsselt



Bildpunkt Schlüssel Verschlüsselt





C++ Grundlagen

Hello World



main.cpp

```
#include <iostream>

int main(int argn, char ** argv) {
    std::cout << "Hello World!" << std::endl;
    return 0;
}
```

```
bash> g++ main.cpp -o main.o
```

```
bash> . main.o
```

```
Hello World!
```



Datentypen - Zahlen

- Ganze Zahlen
 - char, short, **int**, long, long long, **size_t**
 - signed und unsigned Varianten
 - Wertebereiche Compilerabhängig
- Gleitkommazahlen
 - float, **double**, long double
 - Wissenschaftliche Notation möglich
- Automatische Typkonvertierung (Promotion)

Operatoren



Fließkommazahlen, Integer

Operator	Bezeichnung
+	Addition
-	Subtraktion
*	Multiplikation
/	Division

Nur Integer

Operator	Bezeichnung
%	Modulo
++a	Präinkrement
a++	Postinkrement
--a	Prädecrement
a--	Postdecrement

Vergleiche

Operator	Bezeichnung
==	Gleich
!=	Ungleich
<=	Kleiner gleich
>=	Größer gleich
<	Kleiner
>	Größer

Datentypen - Zahlen

```
int ia(20);    // Varianten
int ib{20};

int dec = 123;  // Zahlensysteme
int oct = 0173;
int hex = 0x7B;

signed int a = -2147483648; // Datentypen
long l = -2147483649L;
unsigned short int c = 3u;

int b = 100;   // Rechnung
auto d = b / c;

std::cout << b*c << std::endl;
std::cout << d << std::endl; // Ergebnis?
```

Datentypen - Wahrheitswerte

- Schlüsselwort: `bool`
- Werte
 - `true`
 - `false`

Logische Operatoren

Operator	Bezeichnung
!	Negation
&&	Logisches Und
	Logisches Oder

```
bool b_x(true), b_y, b_z;  
bool b_u = 1;  
  
b_y = 20 < 17;  
b_z = (b_x && b_y) || b_u;  
std::cout << b_z << std::endl
```


Operatoren (2)



Bitoperator

Operator	Bezeichnung
~	Negation
&	Bitweises Und
	Bitweises Oder
^	Exklusiv-Oder

Schiebeoperator

Operator	Bezeichnung
<<	Links-Schift
>>	Rechts-Schift
	Bitweises Oder

	179
&	155
=	147

	1 0 1 1 0 0 1 1
&	1 0 0 1 1 0 1 1
=	1 0 0 1 0 0 1 1

Pointer – Zeiger

```
int *p;           // Zeiger auf int
p = new int;      // int Wert erzeugen
*p = 15;          // Wert zuweisen

cout << p << endl; // Pointer Wert
cout << *p << endl; // Wert 15

delete p;
```

```
int a = 5;
int *p = {&a};    // Zeiger auf int

cout << *p << endl; // Wert 15

delete p;         // Fehler!
```

Arrays – statisch

- Index startet mit 0
- Eindimensionale Arrays

```
constexpr int size = 5;  
double a[size];  
  
a[2] = 5;  
*(a+5) = 5; // Äquivalent zu der Zeile vorher  
  
double initializer[] = {1.0, -4.3, 7.8};
```

- Mehrdimensionale Arrays

```
constexpr int dim = 3;  
double carray[size][dim]  
carray[2][1] = 3
```

Arrays – dynamisch

- Eindimensionale Arrays

```
int * pa = new int[5];  
pa[4] = 35;  
  
delete pa[]
```

- Mehrdimensionale Arrays

```
int rows = 4;  
int cols = 6;  
int **pa2d = new int*[rows];  
for (int i = 0; i < rows; ++i){  
    pa2d[i] = new int[cols];  
}
```

Datentypen - Zeichenketten

- Einzelnes Zeichen
 - Schlüsselwort: char

```
char a = 'c';  
char a_dec = 99;  
std::cout << a << std::endl;
```

- Zeichenkette
 - Eindimensionales Array mit Typ char
 - Schlüsselwort char[constexpr], char *
 - Endet mit \0

```
const char[] source = "Lorem ipsum dolor sit amet."  
char dest[80]; // Platz muss reichen!
```

Datentypen – Eigene

- Aufzählungen

```
enum Wochentag {  
    montag, dienstag, mittwoch,  
    donnerstag, freitag, samstag, sonntag  
};
```

```
Wochentag heute = Wochentag::dienstag;  
  
if (heute == Wochentag::montag) {  
    std::cout << "Heute ist Montag" << std::endl;  
}
```

- Typendefinition

```
typedef std::vector<std::vector<double>> MatrixTyp;  
  
Matrix m = {{1.0, 3.0, 4.0},  
            {-1.0, 3.0, -5.5}};
```


Kontrollstrukturen (1)

```
if (Bedingung) {  
    ...  
} else if (Bedingung_2) {  
    ...  
} else {  
    ...  
}
```

```
if (Bedingung)  
    Anweisung_1  
else if (Bedingung_2)  
    Anweisung_2  
else  
    Anweisung_3
```

```
if (x > 5) {  
    std::cout << 'zu groß';  
} else if (x < 1) {  
    std::cout << 'zu klein';  
}
```

Probleme

```
if (x == 5)  
    if (y <= 4)  
        std::cout << 'fall A';  
    else  
        std::cout << 'fall B';
```

```
auto a = Bedingung ? then_fall : else_fall;
```

Kontrollstrukturen (2)

```
switch (Ausdruck) {  
    case fall_1:  
        ...  
        break;  
    case fall_2:  
        ...  
        break;  
    default:  
        ...  
}
```

- Schleifen

```
while (Bedingung) {  
    ...  
}
```

```
char buchstabe = 'b'  
switch (buchstabe) {  
    case 'a':  
        std::cout << ':' << std::endl;  
        break;  
    case 'b':  
        std::cout << 'b' << std::endl;  
        break;  
    default:  
        std::cout << "ungültig";  
}
```

```
do {  
    ...  
} while (Bedingung)
```

Kontrollstrukturen (3)

```
for(Initialisierung; Bedingung; Veränderung){  
    ...  
}
```

```
Initialisierung  
while(Bedingung){  
    ...  
    Veränderung  
}
```

```
for(size_t i = 20; i < 30; i += 2){  
    if (i == 24){  
        i = 21  
    }  
    std::cout << i << ", ";  
}  
std::cout << std::endl;
```

```
auto container = std::vector<typ>()  
for(auto & v : container){  
    std::cout << v << std::endl;  
}
```

Funktionen (1)

```
double quadrat(double); // Prototyp (Deklaration)

std::cout << quadrat(5.0) << std::endl;

double quadrat(double a) {
    return a*a;
}
```

```
void increment(int& x, int val) {
    x += val;
}

int x = 21;
increment(x, 7);
std::cout << x << std::endl;
```

Funktionen (2)

- Überladung

```
double min(double x, double y) {  
    return x > y ? x : y;  
}  
  
int min(int x, int y) {  
    return x > y ? x : y;  
}
```

- Templates

```
template<type T>  
T min(T x, T y) {  
    return x > y ? x : y;  
}
```

Funktionen – Fehler

```
int& fun(int a, int b) {  
    if (a > b) {  
        return b;  
    } else {  
        int c = a + 20;  
        return c;  
    }  
}
```


Eingabe und Ausgabe

- Standardausgabe (cout)
- Standardfehlerausgabe (cerr)
- Standardeingabe

```
std::string a;  
cin >> a // Eingabe: Hallo Welt!  
std::cout << a; // Ausgabe?  
int b, double c;  
while(cin >> b >> c){ // Eingabe: 10 2.0 2.7 3.5  
    std::cout << b << c; // Ausgabe?  
}
```

```
char d;  
cin.get(d);  
std::cout << d
```

- Behandlung genau wie Eingabe und Ausgabe

```
std::string dateiname = "datei.txt";
std::ifstream quelle(dateiname);
if(!quelle){
    std::cerr << "Fehler" << std::endl;
} else{
    while (quelle){
        std::string line;
        std::getline(quelle, line);
        char a; quelle.get(a)
        int b;
        quelle >> b;
    }
}
```

Literatur

- Der C++ Programmierer – Breymann – 3. Auflage
- C++ – von A bis Z – Galileo Computing – 1. Auflage
- Pixelspiele – Scholz – 1. Auflage
- Visuelle Kryptographie – Klein – 1. Auflage



Prof. Dr. Ivan Kisel

Office 02/10

Giersch Science Center

Max-von-Laue-Straße 12

60438 Frankfurt am Main

I.Kisel [At] compeng.uni-
frankfurt.de

Martin Parnet

parnet@stud.uni-frankfurt.de