Technische Universität Dortmund

Klausur „Einführung in die Programmierung“

	erreichbare Punkte	erhaltene Punkte			
Aufgabe 1	20	**a**	**b**	**c**	
Aufgabe 2	20	**a**	**b**	**c**	
Aufgabe 3	20	**a**	**b**		
Aufgabe 4	20	**a**	**b**		
Aufgabe 5	20	**a**	**b**	**c**	
Aufgabe 6	20	**a**	**b**		
Summe	120				
Spickzettel vorhanden?					

(Matrikel-Nr.)

(Name) (Vorname)

Durch meine Unterschrift bestätige ich

- den Empfang der vollständigen Klausur (24 Seiten inklusive Deckblatt),
- die Kenntnisnahme der Hinweise auf Seite 2.

Dortmund, 18.03.2025

(Unterschrift)

Hinweise

Bitte lesen Sie die folgenden Informationen **aufmerksam** und unterschreiben Sie die Erklärung auf der ersten Seite.

- Es können **120 Punkte** erreicht werden; 50% der Gesamtpunktzahl sind zum Bestehen der Klausur erforderlich. Zur Bearbeitung stehen insgesamt **180 Minuten** zur Verfügung.
- Als Hilfsmittel ist lediglich ein **von Ihnen eigenhändig geschriebenes** (Handschrift, kein Ausdruck oder Kopie) **A4-Blatt** (beidseitig), welches eingesammelt wird. Ansonsten dürfen **keine** weiteren Hilfsmittel verwendet werden.
- Die Antworten sind in **deutscher** oder **englischer** Sprache zu verfassen.
- Notieren Sie Ihre Lösungen direkt auf den ausgeteilten Aufgabenblättern unter Verwendung eines dokumentenechten, schwarzen oder blauen Stifts. Entfernen Sie **nicht** die Heftung der Blätter. Falls der vorgesehene Platz nicht ausreichen sollte, können Sie auch die Rückseiten der Blätter und die Reserveseiten am Ende verwenden. Notieren Sie in diesem Fall aber an der für die Lösung vorgesehenen Stelle einen Verweis auf die Seite.
- Da es unterschiedliche Sprachgebräuche sowie Algorithmen- und Konzept-Ausprägungen gibt, werden hier ausdrücklich die aus Vorlesung und Übungen bekannten Ausdrucksweisen und Ausprägungen zu Grunde gelegt.

Aufgabe 1: Programmverständnis (20 Punkte)

a) Kontrollstrukturen

6 Punkte

Geben Sie die Ausgabe des folgenden C++-Programms an. Das Symbol ␣ steht für ein Leerzeichen.

```
1  #include <iostream>
2  using namespace std;
3
4
5  void f(int i) {
6    i = i - 2;
7  }
8
9  int main() {
10   int i = 4, k = 11;
11
12   do {
13     k++;
14   } while (k < 11);
15
16   while (k > 0) {
17     if (i == 3) {
18       i--;
19       continue;
20     } else if (i > 0) {
21       cout << i * 2;
22     } else {
23       cout << i + 1;
24     }
25     k -= 2;
26
27     f(k);
28     --i;
29     cout << ",";
30   }
31   cout << endl;
32
33   return 0;
34 }
```

Ausgabe:

---

b) Parameterübergabe

9 Punkte

Geben Sie die Ausgabe des folgenden C++-Programms an. Das Symbol ␣ steht für ein Leerzeichen.

```
1  #include <iostream>
2  using namespace std;
3  int l = 8;
4
5  void f(int k) {
6      k = k * 1000 / k + k * 3;
7  }
8
9  int g(int *i, int a) {
10   *i -= 1;
11   int h = *i;
12   if (a < 0) {
13     return *i;
14   } else if (a == 0) {
15     return g(i, a - 1);
16   } else {
17     return h - g(i, a - 1);
18   }
19 }
20
21 void h(int &k) {
22   for (int i = 0; i < 5; i++) {
23     k = k + 3;
24   }
25 }
26
27 int main() {
28   int *k = &l;
29
30   f(*k);
31   cout << *k << endl;
32
33   cout << g(k, 1) << endl;
34
35   h(l);
36   cout << l << endl;
37
38   return 0;
39 }
```

Ausgabe:

---

c) Fehlersuche

5 Punkte

In dem folgenden Programm haben sich mehrere Fehler versteckt. Sie können zu einem Fehler beim Übersetzen *oder* zu einem potentiellen Fehler während der Ausführung führen. Pro Zeile versteckt sich maximal *ein* Fehler. Das Symbol ␣ steht für ein Leerzeichen.

Nennen Sie pro Fehler die Zeile, in der er vorkommt, und was genau das Problem ist.

```
1  #include <iostream>
2  using cout;
3
4
5  int fakultaet(int i) {
6
7    return fakultaet(i - 1 ) * i;
8  }
9
10
11 int main() {
12   int k = {1, 6, 4, 3, 5, 2};
13
14   for ( i = 0; i < 6; i++) {
15     cout << "fakultaet(" << k[i] << ")␣=␣" << fakultaet(k[i]) << endl;
16   }
17
18   return 0;
19 }
```

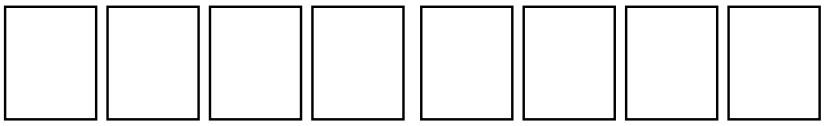
Fehler:

Aufgabe 2: Darstellung von Daten und Speicherverwaltung (20 Punkte)

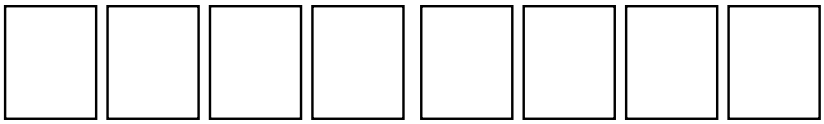
a) Binärsystem

6 Punkte

Geben Sie die Zahl 93 als 8-Bit breite vorzeichenlose Binärzahl an. Tragen Sie die Bits in die nachfolgenden Kästchen ein, wobei das linkeste das höchstwertigste Bit (MSB) und das rechteste das niederwertigste Bit (LSB) der Summe enthalten soll:



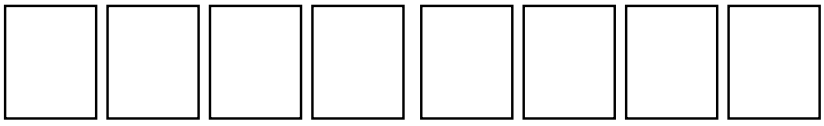
Geben Sie außerdem die Zahl -93 als 8-Bit breite Binärzahl in der Zweierkomplementendarstellung an. Tragen Sie die Bits in die nachfolgenden Kästchen ein, wobei das linkeste das Vorzeichenbit enthalten soll:



b) Multiplikation

4 Punkte

Gegeben seien die beiden 8-Bit breite vorzeichenlosen Binärzahlen `0b00100101` und `0b00000101`. Berechnen Sie das Produkt beider Zahlen und geben Sie diese als 8-Bit breite vorzeichenlosen Binärzahl an. Tragen Sie die Bits in die nachfolgenden Kästchen ein, wobei das linkeste das höchstwertigste Bit (MSB) und das rechteste das niederwertigste Bit (LSB) der Summe enthalten soll:



c) Speichersegmente

10 Punkte

Schauen Sie sich alle Bezeichner im folgenden C++-Code an und tragen Sie diese in die Liste der Speichersegmente ein, welche Sie auf der nächsten Seite finden. Tragen Sie ausschließlich nur den Bezeichner in die Liste ein! Ordnen Sie ebenfalls den Inhalt von ∗p einem Segment zu.

```
1  #include <iostream>
2  #include <new>
3  #define BUFSIZE 128
4
5  constexpr int size = 8;
6  int arr[size] = {7, 3, 9, 1, 5, 2, 8, 9};
7  double avg = 0.0;
8  char buffer[BUFSIZE];
9
10 void sort(int* a, int n) {
11     for (int i = 0; i < n - 1; ++i) {
12         for (int j = 0; j < n - i - 1; ++j) {
13             if (a[j] > a[j + 1]) {
14                 int temp = a[j];
15                 a[j] = a[j + 1];
16                 a[j + 1] = temp;
17             }
18         }
19     }
20 }
21
22 double mean(int* b, int m) {
23     double sum = 0;
24     for (int k = 0; k < m; ++k) { sum += b[k]; }
25     return sum / m;
26 }
27
28 int select(int* c, int l) {
29     return (l % 2) ? c[l / 2] : (c[l / 2 - 1] + c[l / 2]) / 2;
30 }
31
32 int main() {
33     static int arrsize = size;
34     int* p = new int[arrsize];
35     for (int i = 0; i < arrsize; ++i) { p[i] = arr[i]; }
36     sort(p, arrsize);
37     avg = mean(p, arrsize);
38     std::cout << "Median:␣" << select(p, arrsize) << "\n";
39     std::cout << "Mean:␣" << avg << "\n";
40     delete[] p;
41     return 0;
42 }
```

Tabelle für Teilaufgabe b):

Stack:
Heap:
BSS-Segment:
Datensegment:
Textsegment:

Aufgabe 3: Zeiger (20 Punkte)

a) Zeigerarithmetik

8 Punkte

Gegeben sei folgendes Programm, welches ausführlich Gebrauch von Zeigern macht.

```
1  #include <iostream>
2
3  const int ONE = 1;
4
5  int main() {
6      int a = 8;
7      int* pa = &a;
8      int b = a + 7;
9      int c = *pa - 4;
10
11     std::cout << "A) " << a << ", " << b << ", " << c << std::endl;
12
13     int d = *(&ONE) + ONE;
14     int* pd = &d;
15     pd = &pd[2];
16     int e = *(pd - 2) - 26;
17
18     std::cout << "B) " << d << ", " << e << std::endl;
19
20     short f[] = {12, 23, 35, 58, 93, 0};
21     short* g = &f[4];
22     f[*(g+1)] = g[-2] - 20;
23     *(f+4) += g[-4];
24     g[1] = f[1];
25
26     std::cout << "C) " << g[0] << ", " << g[1] << std::endl;
27
28     char h[] = "REGALXLAGER";
29     char* i = &h[5];
30     i[0] = h[0] - h[10];
31
32     std::cout << "D) " << h << std::endl;
33
34     return 0;
35 }
```

Welche Ausgabe erzeugt das gegebene Programm?

Tipp: Rechnen Sie schriftlich, um Rechenfehler zu vermeiden.

Platz für Notizen:

b) Verkettung

12 Punkte

Gegeben Sei folgende Header-Datei zur Klasse `RingBinder`, die ein Ringbuch mit Seiten umsetzt. Dabei besitzt jede `Page` (Seite) einen Inhalt, eine Referenz auf die Seite vor ihr und eine Referenz auf die Seite nach ihr. Das Ringbuch selbst hält mit Referenzen nach, was seine erste bzw. letzte Seite ist und zählt, wie viele Seiten es insgesamt enthält.

Implementieren Sie die Funktion `filePage`, die simuliert eine neue Seite zu erzeugen und in das Ringbuch einzuheften. Die Funktion nimmt als Parameter die Seitenzahl (`pageNumber`), die die neue Seite im Ringbuch haben soll, sowie den Inhalt der neuen Seite (`pageContent`).

Seitenzahlen in Büchern beginnen bei 1. Gehen Sie davon aus, dass das Ringbuch immer eine Deckseite (erste Seite) und eine leere Schlussseite (letzte Seite) hat. Es darf also *niemals* eine neue Seite als neue erste oder neue letzte Seite ins Ringbuch eingeheftet.

```
1  #include <iostream>
2  using namespace std;
3
4  class RingBinder{
5  private:
6      struct Page{
7          string pageContent; // Seiteninhalt
8          Page* previousPage; // Zeiger auf die vorherige Seite
9          Page* nextPage;     // Zeiger auf die nachfolgende Seite
10     };
11     Page* firstPage;
12     Page* lastPage;
13     int numberOfPages;
14
15 public:
16     RingBinder() {
17         firstPage = lastPage = nullptr;
18         numberOfPages = 0;
19     }
20     ~RingBinder() { destroy(); }
21     void destroy();
22     int getNumberOfPages() { return this->numberOfPages; }
23     void filePage(int pageNumber, string pageContent); // Zu implementieren
24 };
```

```
1 #include "a3-b.hpp"

void RingBinder::filePage(int pageNumber, string pageContent) {
}
```

Aufgabe 4: Rekursion (20 Punkte)

a) Sparprogramm

8 Punkte

Eine Bank bietet das Sparprogramm *Kapital1000* an. Dabei zahlt man ein Mal 1000 Euro auf einem Sparkonto ein. Danach werden dem Konto jährlich Zinsen zu einem Zinssatz von $p \in \mathbb{R}$ gutgeschrieben. Jedoch werden jährlich Gebühren in Höhe von ein Prozent des Guthabens fällig, wenn das Guthaben 2500 Euro übersteigt. Für das Guthaben nach n Jahren ergibt sich damit

$$K_n = \begin{cases} 1000 & \text{, wenn } n = 0, \\ (1+p) \cdot K_{n-1} - \frac{K_{n-1}}{100} & \text{, wenn } n > 0 \text{ und } K_{n-1} > 2500 \\ (1+p) \cdot K_{n-1} & \text{, sonst.} \end{cases}$$

Implementieren Sie eine Funktion `double guthaben(unsigned int n, double p)`, die für einen gegebenen Zinssatz $p > 0.0$ rekursiv das Guthaben nach n Jahren nach obiger Vorschrift berechnet und zurückgibt. Erfüllt der Zinssatz p die Bedingung nicht, soll nichts berechnet werden.

```
double guthaben(unsigned int n, double p) {
}
```

b) Binärer Suchbaum

12 Punkte

Gegeben sei die Implementierung eines binären Suchbaums, deren relevante Bestandteile unten dargestellt sind.
Implementieren Sie die private Methode `depthFactorSum`, die rekursiv eine Summe über alle Knoten berechnet, wobei für jeden Knoten das Produkt des Knotenwertes (`data`) mit der Tiefe des Knotens berechnet wird.
Die Tiefe eines Knotens gibt die Anzahl der Kanten bis zur Wurzel an. Die Wurzel hat Tiefe 0 und steigt mit jeder weiteren Kante nach unten um 1 an – siehe Zeile 15. Ein leerer Baum hat keine Knoten.

```
1  class BinTree {
2  private:
3    struct Node {
4      int data;
5      Node *left, *right;
6    } *root;
7    Node *insert(Node *node, int value);
8    int depthFactorSum(Node *node, int depth);
9  public:
10   BinTree() : root(nullptr) { }
11   void insert(int value) {
12     root = insert(root, value);
13   }
14   int depthFactorSum()  {
15     return depthFactorSum(root, 0);
16   }
17   ~BinTree();
18 };
```

```
int BinTree::depthFactorSum(Node *node, int depth) {
}
```

Aufgabe 5: Klassen (20 Punkte)

In dieser Aufgaben sollen Sie eine Klassenhierarchie mit virtuellen Methoden realisieren.
Sie sollen die *abstrakte* Klasse `GeomForm`, die eine beliebige geometrische Form repräsentiert, implementieren. Sofern es *nicht* anders angegeben wird, soll alles öffentlich sichtbar sein. Von der Oberklasse sollen Sie die konkrete Unterklasse `Rectangle` ableiten. Die Funktionalität der Klassen sieht wie folgt aus:

a) Klasse `GeomForm`

- Jede geometrische Form soll lediglich über zwei Koordinaten `x` und `y` verfügen. Diese sollen Dezimalzahlen mit doppelter Genauigkeit speichern können. Außerdem sollen sie nur für sich selbst sowie für abgeleitete Klassen sichtbar sein.
- Dem Konstruktor werden zwei Parameter, die den Attributen entsprechen, übergeben und diese entsprechend initilisiert.
- Für beide Attribute soll es eine Methode geben, die den Wert zurückliefert.
- Jede Form hat eine *rein virtuelle* Methode `getArea()`, die die Fläche der geometrischen Form als Dezimalzahl mit doppelter Genauigkeit zurückgibt.
- Abschließend soll es eine Methode `getDesc()` (nicht virtuell) geben, die den Text „Ich bin eine Form“ zurückliefert.

b) Klasse `Rectangle`

- Jedes Rechteck hat *zusätzlich* noch zwei Attribute für die Breite und Höhe, die nur ihr gehören. Der Datentyp soll passend zu den bisheringen Attributen gewählt werden.
- Der Konstruktor für ein Rechteck bekommt alle *vier* Attribute übergeben. Die eigenen Attribute sollen entsprechend gesetzt werden. Außerdem sollen die Koordinaten `x` und `y` mit Hilfe des Konstruktors der Oberklasse gesetzt werden.
- Eine Rechteck implementiert die Methode `getArea()`, indem sie die Fläche berechnet und zurückgibt .
- Die Methode `getDesc()` soll die Zeichenkette „Ich bin ein Rechteck“ zurückgeben.

```
1 #include <iostream>
2 #include <deque>
3 using namespace std;
```

a) Klasse `GeomForm` (9.5 Punkte)

```
class GeomForm {
};
```

a:

b) Klasse `Rectangle` (6.5 Punkte)

```
class Rectangle : public GeomForm {
};
```

b:

c) Polymorphie

4 Punkte

In dem folgenden Programm werden in den Zeilen 19-22 vier Instanzen der beiden zuvor implementierten Klassen `Rectangle` und `Triangle` erzeugt und in eine Liste eingefügt. Im Anschluss wird über jedes Element der List iteriert und eine Ausgabe getätigt.

Notieren Sie unten, wie die Ausgabe für die vier Instanzen lautet.

```
1  class Triangle : public GeomForm {
2      private:
3          double m_height;
4          double m_base;
5      public:
6          Triangle(double x, double y, double height, double base) :
7              GeomForm(x, y), m_height(height), m_base(base) {}
8          string getDesc() {
9              return "Ich bin ein Dreieck";
10         }
11         double getArea() {
12             return m_height * m_base / 2;
13         }
14 };
15
16 int main() {
17     deque<GeomForm*> list;
18
19     list.push_back(new Rectangle(1.0, 2.0, 3.0, 3.0));
20     list.push_back(new Triangle (4.0, 4.0, 4.0, 4.0));
21     list.push_back(new Triangle (1.5, 3.0, 1.0, 3.0));
22     list.push_back(new Rectangle(1.5, 3.0, 1.0, 1.0));
23
24     int i = 1;
25     for (GeomForm *form : list) {
26         cout << i << ": " << form->getDesc() << ", A=";
27         cout << form->getArea() << endl;
28
29         i++;
30     }
31     // [...]
32
33     return 0;
34 }
```

Aufgabe 6: Schablonen & Ausnahmen (20 Punkte)

a) Summentyp als Schablone (9 Punkte)

Die unten angegebene Klasse stellt eine Summentypen bereit. Mit dessen Hilfe können Werte verschiedener Typen als ein gemeinsamer Typ verpackt und später wieder extrahiert werden.

In dieser konkreten Implementierung können Werte entweder vom Typ `char` oder `const char*` sein.

Ändern Sie die Klasse zu einer Klassen-Schablone mit zwei Typ-Parametern (Vorzugsweise als A und B benannt), die Werte von beliebigen Typen halten kann. Ändern Sie außerdem die `main`-Funktion so ab, dass Sie die Schablone benutzt.

Verwenden Sie dafür den vorgegebenen Code auf den folgenden Seiten. Passen Sie diesen direkt an, indem Sie **Code hinzufügen** oder bestehenden **Code durchstreichen und ersetzen**.

```
1 #include <iostream>
2 #include <string>
3 using namespace std;

class Summentyp {

public:

  Summentyp() : value(nullptr), type(0) {}

  Summentyp(char value) : value(new char(value)), type('A') {}

  Summentyp(const char* value) : value(new const char*(value)),
     type('B') {}

  char* getValueA() const {

    if (type == 'A') {

      return static_cast<char *>(value);

    } else { return nullptr;}
  }
```

```
  const char** getValueB() const {
    if (type == 'B') {
      return static_cast<const char**>(value);
    } else { return nullptr;}
  }
  void setValue(char value) {
    deleteValue();
    this->value = new char(value);
    type = 'A';
  }
  void setValue(const char* value) {
    deleteValue();
    this->value = new const char*(value);
    type = 'B';
  }
  char getVariante() { return type; }
```

```
  void deleteValue() {

    switch (type) {

    case 'A':

      delete static_cast<char *>(value);

      break;
    case 'B':

      delete static_cast<const char**>(value);

    default:
      break;
    }
    value = nullptr;
    type = 0;
  }

  ~Summentyp() { deleteValue(); }

private:

  void *value;
  char type;
};
int main() {

  Summentyp summentyp;
  summentyp.setValue("Hallo");
  summentyp.setValue('!');
  cout << "Wert: " << *summentyp.getValueA() << endl;
  return 0;}
```

b) Ausnahmen (11 Punkte)

Das nachfolgend angegebene Programm arbeitet mit Fehlercodes. Verändern Sie die Funktion und das Hauptprogramm, indem Sie die Fehlerbehandlung mit Fehlercodes entfernen (ggf. den Rückgabetyp ändern) und durch eine Ausnahmebehandlung mit C++-Exceptions ersetzen. Nutzen Sie dazu eine sinnvolle Anzahl an Klassen zur differenzierten Fehlerbehandlung, welche von einer gemeinsamen Oberklasse erben. Die Ausgabe der Fehlermeldungen soll in eine virtuelle Methode der Ausnahmeklassen erfolgen. Der Parameter `ungueltigerWert` soll aus dem Kopf von `berechneWerte` entfernt werden, ungültige Werte sollen aber weiterhin in der Fehlermeldung ausgegeben werden. Notieren Sie Ihre Lösung **auf den nächsten Seiten**.

Die Bedeutung der Fehlercodes sowie das daraus resultierende Verhalten in der ursprünglichen Implementierung sieht wie folgt aus:

- **0:** Es trat kein Fehler auf.
- **1:** Es wurde ein leeres Array übergeben. Es erfolgte keine Berechnung.
- **2:** Der im Array gespeicherte Wert liegt außerhalb des Wertebereichs. In diesem Fall wird der ungültige Wert in der Referenz `ungueltigerWert` abgelegt.

```
1  #include <iostream>
2  using namespace std;
3
4  int berechneWerte(int * werte, unsigned int const n,
5          int &ungueltigerWert) {
6      int errCode = 0;
7      if (n == 0) return errCode = 1;
8      for (unsigned int i = 0; i < n; ++i) {
9          if (werte[i] < 0 || werte[i] > 2047) {
10             errCode = 2;
11             ungueltigerWert = werte[i];
12         }
13         werte[i] *= (werte[i] + 1);
14     }
15     return errCode;
16 }
17 int main() {
18     int const n = 4;
19     int werte[n] = { 2, 1, -2, 1 };
20     int ungueltigerWert = 0;
21     int errCode = berechneWerte(werte, n, ungueltigerWert);
22     if (errCode == 1)
23         cerr << "Fehler:␣Arraygroesse␣0" << endl;
24     else if (errCode == 2)
25         cerr << "Fehler:␣Array␣enthaelt␣ungueltigen␣Wert␣"
26              << ungueltigerWert << endl;
27     else {
28         for (unsigned int i = 0; i < n; ++i) {
29             cout << werte[i] << endl;
30         }
31     }
32     return 0;
33 }
```

```
1 #include <iostream>
2 using namespace std;
```

```
int main() {
  int const n = 4;
  int werte[n] = { 2, 1, 2, 0 };
}
```