

Name: _____

Matrikelnr.: _____

Hinweise zur Bearbeitung der Klausur
zum Kurs 01613 „Einführung in die imperative Programmierung“

1. Prüfen Sie die Vollständigkeit Ihrer Unterlagen. Die Klausur umfasst:
 - 2 Deckblätter
 - 1 Formblatt für eine Bescheinigung für das Finanzamt
 - diese Hinweise zur Bearbeitung
 - 4 Aufgaben (Seite 2-14)
 - die Muss-Regeln des Programmierstils
2. Füllen Sie, **bevor** Sie mit der Bearbeitung der Aufgaben beginnen, folgende Seiten des Klausurexemplares aus:
 - Beide Deckblätter mit Namen, Anschrift sowie Matrikelnummer.
 - Falls Sie eine Teilnahmebescheinigung für das Finanzamt wünschen, füllen Sie bitte das entsprechende Formblatt aus und belassen Sie es in der Klausur. Sie erhalten es dann zusammen mit der Korrektur abgestempelt zurück.

Nur wenn Sie die Deckblätter vollständig ausgefüllt haben, werden wir Ihre Klausur korrigieren!
3. Schreiben Sie Ihre Lösungen jeweils auf den freien Teil der Seite unterhalb der Aufgabe bzw. auf die leeren Folgeseiten. Sollte dies nicht möglich sein, so vermerken Sie, auf welcher Seite die Lösung zu finden ist.
4. **Streichen Sie ungültige Lösungen deutlich durch!** Sollten Sie mehr als eine Lösung zu einer Aufgabe abgeben, so wird nur eine davon korrigiert – und nicht notwendig die bessere.
5. Schreiben Sie auf jedes von Ihnen beschriebene Blatt oben links Ihren Namen und oben rechts Ihre Matrikelnummer. Wenn Sie weitere eigene Blätter benutzt haben, heften Sie auch diese, mit Namen und Matrikelnummer versehen, an Ihr Klausurexemplar. Nur dann werden auch Lösungen außerhalb Ihres Klausurexemplares gewertet!
6. Neben unbeschriebenem Konzeptpapier und Schreibzeug (Füller oder Kugelschreiber, benutzen Sie **keinen Bleistift** und **keinen Rotstift!**) sind **keine weiteren Hilfsmittel** zugelassen.
7. Es sind maximal 24 Punkte erreichbar. Sie haben die Klausur bestanden, wenn Sie mindestens 12 Punkte erreicht haben.

Aufgabe 1 (2+4 Punkte)

Gegeben sei das folgende Programm:

```
program WasPassiert(input,output);
{ 
 }

var
  a,b,c:integer;

begin
  b:=0;
  c:=1;
  readln(a);
  while a>0 do
    begin
      b:=b+c*(a mod 2);
      a:=a div 2;
      c:=c*10;
    end;
  writeln(b)
end.
```

a) Was gibt das Programm für die Eingabe a=10 aus?

Was gibt das Programm für die Eingabe a=7 aus?

b) Ergänzen Sie im Programm einen erklärenden Kommentar an der grau eingefärbten Stelle und schreiben Sie eine passende Problemspezifikation:

Eingabe:

Ausgabe:

Nachbedingung:

Kurs 01613 „Einführung in die imperative Programmierung“

Name: _____

Matrikelnr.: _____



Aufgabe 2 (6 Punkte)

Schreiben Sie eine **Prozedur**, die ein Array A vom Typ tFeld mit 20 natürlichen, positiven Zahlen übergeben bekommt. Ihre Prozedur überschreibt so lange positive und doppelt vorkommende Werte aus A mit dem Wert 0, bis jeder ursprünglich in A enthaltene Wert genau einmal in A vorkommt. Der Rest der Werte in A ist nach Aufruf Ihrer Prozedur 0.

Schreiben sie eine Prozedur ohne Hilfsprozeduren oder Funktionen, arbeiten Sie auf dem Array A und erstellen Sie kein weiteres Array.

Für die Eingabe $A = [3, 2, 1, 5, 5, 5, 4, 8, 9, 123, 2, 1, 5, 4, 5, 4, 5, 18, 9, 3]$
liefert Ihre Prozedur $A = [3, 2, 1, 5, 0, 0, 4, 8, 9, 123, 0, 0, 0, 0, 0, 0, 0, 18, 0, 0]$



Kurs 01613 „Einführung in die imperative Programmierung“

Name: _____

Matrikelnr.: _____

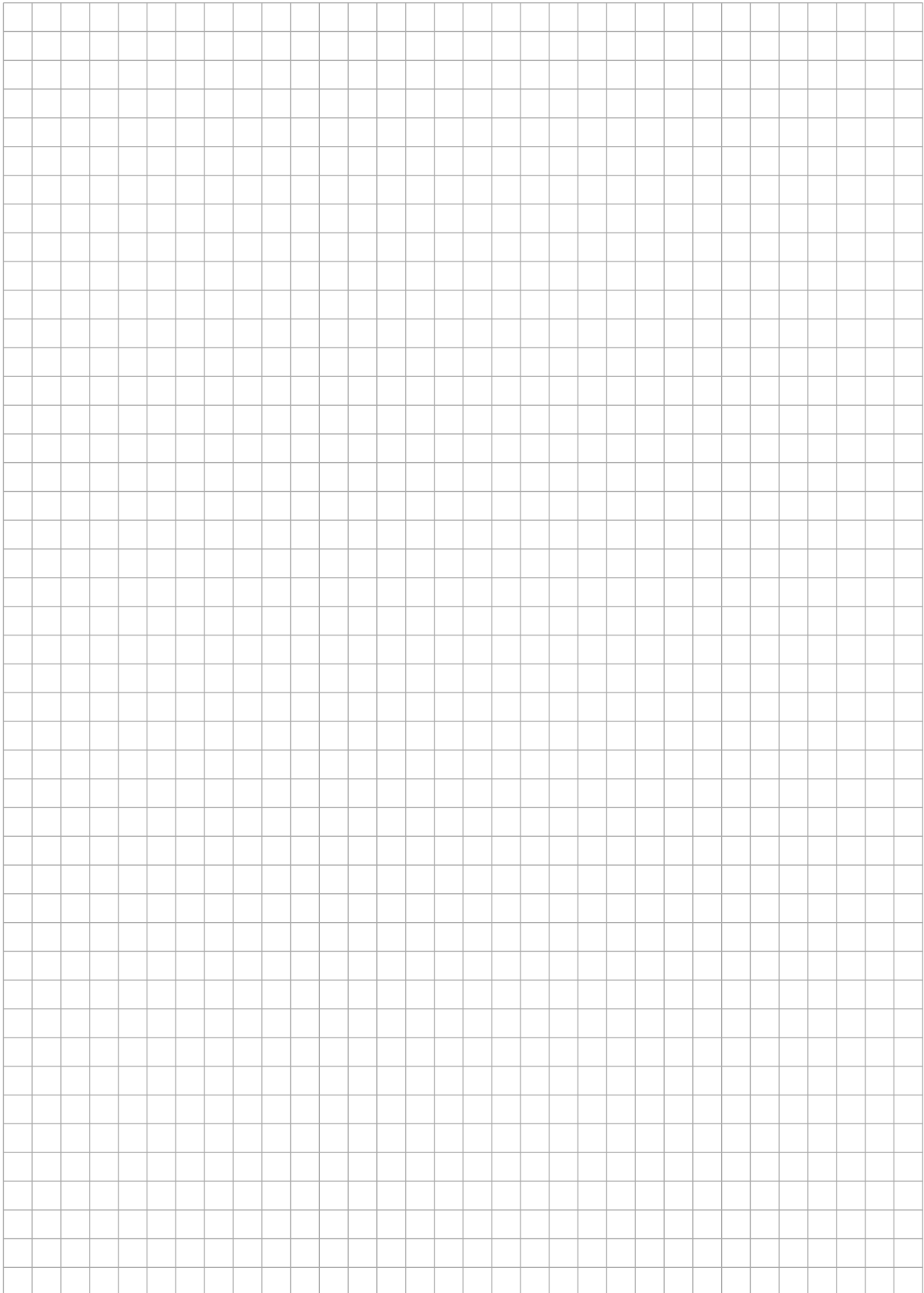


Kurs 01613 „Einführung in die imperative Programmierung“

Name: _____

Matrikelnr.: _____





Kurs 01613 „Einführung in die imperative Programmierung“

Name: _____

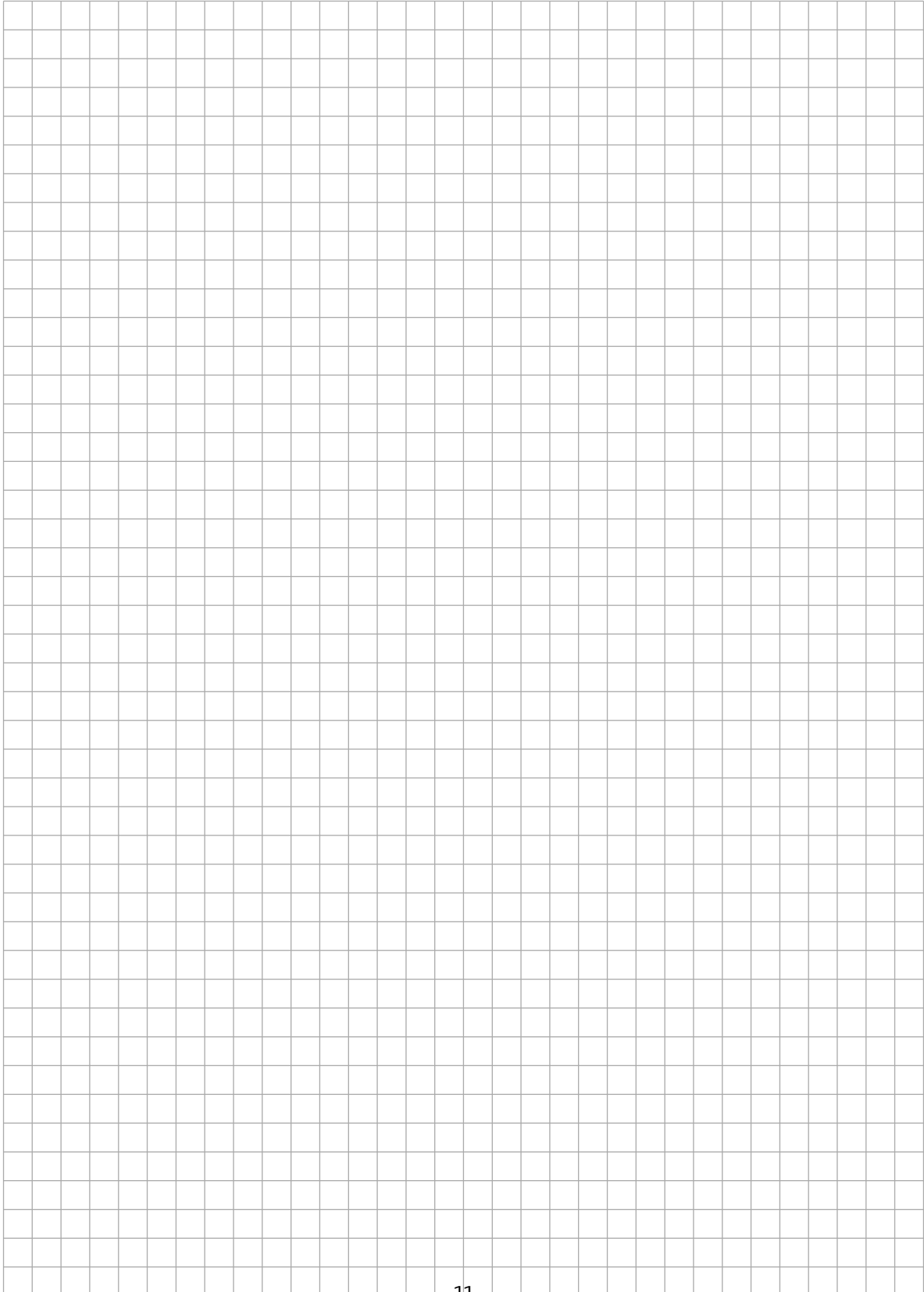
Matrikelnr.: _____



Kurs 01613 „Einführung in die imperative Programmierung“

Name: _____

Matrikelnr.: _____





Kurs 01613 „Einführung in die imperative Programmierung“

Name: _____

Matrikelnr.: _____





Zusammenfassung der Muss-Regeln

1. Selbstdefinierte Konstantenbezeichner bestehen nur aus Großbuchstaben. Bezeichner von Standardkonstanten wie z.B. `maxint` sind also ausgenommen.
2. Typenbezeichnern wird ein `t` vorangestellt. Bezeichnern von Zeigertypen wird ein `tRef` vorangestellt. Bezeichner formaler Parameter beginnen mit `in`, `io` oder `out`.
3. Jede Anweisung beginnt in einer neuen Zeile. `begin` und `end` stehen jeweils in einer eigenen Zeile.
4. Anweisungsfolgen werden zwischen `begin` und `end` um eine konstante Anzahl von 2-4 Stellen eingerückt. `begin` und `end` stehen linksbündig unter der zugehörigen Kontrollanweisung, sie werden nicht weiter eingerückt.
5. Anweisungsteile von Kontrollanweisungen werden genauso eingerückt.
6. Im Programmkopf wird die Aufgabe beschrieben, die das Programm löst.
7. Jeder Funktions- und Prozedurkopf enthält eine knappe Aufgabenbeschreibung als Kommentar. Ggf. werden zusätzlich die Parameter kommentiert.
8. Die Parameter werden sortiert nach der Übergabeart: Eingangs-, Änderungs- und Ausgabeparameter.
9. Die Übergabeart jedes Parameters wird durch Voranstellen von `in`, `io` oder `out` vor den Parameternamen gekennzeichnet.
10. Das Layout von Funktionen und Prozeduren entspricht dem von Programmen.
11. Jede von einer Funktion oder Prozedur benutzte bzw. manipulierte Variable wird als Parameter übergeben. Es werden keine globalen Variablen manipuliert.
12. Jeder nicht von der Prozedur veränderte Parameter wird als Wertparameter übergeben. Lediglich Felder können auch anstatt als Wertparameter als Referenzparameter übergeben werden, um den Speicherplatz für die Kopie und den Kopiervorgang zu sparen. Der Feldbezeichner beginnt aber stets mit dem Präfix `in`, wenn das Feld nicht verändert wird.
13. Pascal-Funktionen werden wie Funktionen im mathematischen Sinne benutzt, d.h. sie besitzen nur Wertparameter. Wie bei Prozeduren ist eine Ausnahme nur bei Feldern erlaubt, um zusätzlichen Speicherplatz und Kopieraufwand zu vermeiden.
14. Wertparameter werden nicht als lokale Variable missbraucht.
15. Die Laufvariable wird innerhalb einer `for`-Anweisung nicht manipuliert.
16. Die Grundsätze der strukturierten Programmierung sind strikt zu befolgen.