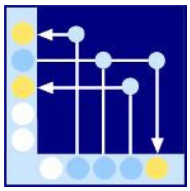
Fakultät Elektronik und Informatik
Studiengang Informatik



Programmieren in C++

Vorlesung im Wintersemester 2017/2018
Prof. Dr. habil. Christian Heinlein

1. Übungsblatt (16. Oktober 2017)

Aufgabe 1: Ganze Zahlen

Teilaufgabe 1.a)

Finden Sie heraus, wie groß die Typen `short`, `int`, `long` und `long long` auf Ihrem System sind und ermitteln Sie ihre minimalen und maximalen Werte!
Was lässt sich damit über die interne Darstellung negativer Werte aussagen?

Finden Sie außerdem heraus, welche Typen sich hinter den Aliasen `int_least16_t` und `int_fast16_t` verbergen!
Tipp: Übergeben Sie Werte dieser Typen jeweils an eine Funktion `test`, die für alle in Frage kommenden Typen überladen ist!

Teilaufgabe 1.b)

Überprüfen Sie die in §2.1.4 der Vorlesungsfolien genannten Regeln, nach denen der Typ eines ganzzahligen Literals ermittelt wird, indem sie gezielt dezimale, oktale, hexadezimale und duale Literale mit bestimmten Werten erstellen und ihren Typ ermitteln!

Teilaufgabe 1.c)

Überprüfen Sie die in §2.1.8 genannten Regeln, nach denen bei einer arithmetischen Operation der gemeinsame Typ der Operanden – der gleichzeitig auch der Resultattyp der Operation ist – ermittelt wird!

Wie lautet demnach das Ergebnis von `1U - 2U` und das von `(unsigned short)1 - (unsigned short)2`?

Teilaufgabe 1.d)

Für welche `int`-Werte `x` und `y` liegt `x/y` nicht im Wertebereich von `int`, sofern `int`-Werte im Zweierkomplement dargestellt werden?

Aufgabe 2: Lange vorzeichenlose ganze Zahlen

Implementieren Sie Addition und Multiplikation langer vorzeichenloser ganzer Zahlen!

Repräsentieren Sie eine solche Zahl quasi zur Basis 256 (allgemeiner `numeric_limits<unsigned char>::max() + 1` oder `(unsigned char)-1`), indem Sie sie als dynamisches Feld von `unsigned-char`-Werten speichern, dessen erstes Element als Längenfeld dient! Realisieren Sie die Operationen dann ähnlich wie beim schriftlichen Addieren und Multiplizieren!
Verwenden Sie für Zwischenergebnisse einen größeren ganzzahligen Typ, beispielsweise `uint_fast32_t`, damit Überträge nicht verloren gehen! Außerdem können Sie ausnutzen, dass arithmetische Operationen mit vorzeichenlosen Werten der Größe n Bit immer korrekt modulo 2^n sind!

Schreiben Sie außerdem Funktionen zum Erstellen solcher Zahlen aus „kleinen“ `unsigned-char`-Werten sowie zur hexadezimalen Ausgabe solcher Zahlen!