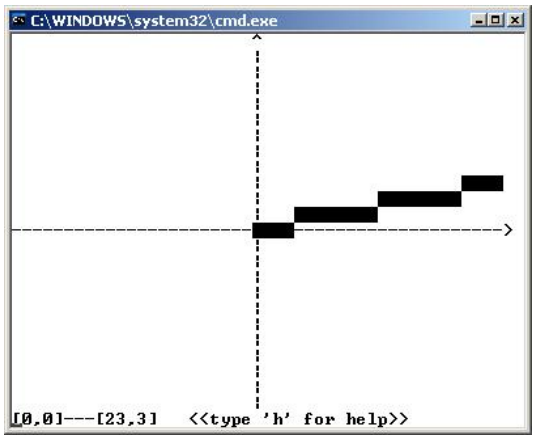
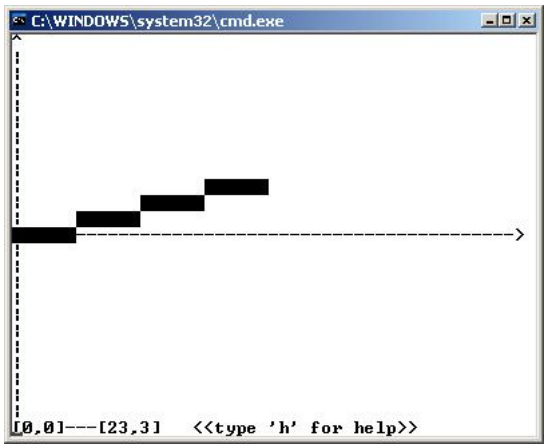
Übung Nr. 1:

In Anlehnung an den Linien-Algorithmus von J.E. Bresenham ist ein Interpolationsverfahren für ganzzahlige Definitions- und Wertebereiche zu implementieren. Anders als beim ursprünglichen Algorithmus soll dabei eine „Plateau"-Bildung vermieden werden (s. Abb.).



Bresenham-Algorithmus



Gesuchter Algorithmus `set_iLerp`

Nach Eingabe der Anzahl (`Dx`) von Stützstellen, die berechnet werden sollen, und der Grenzen (`y0` und `yn`) des Wertebereichs (die ebenfalls zum Wertebereich gehören: abgeschlossenes Zahlenintervall), sollen ganzzahlige Werte errechnet und hinter einer übergebenen Adresse (`yStore`) abgelegt werden, so, daß sie möglichst genau eine Interpolation zwischen den Enden des Zahlenintervalls wiedergeben. Dazu ist der in der Vorlesung besprochene Algorithmus für den ersten Oktanten implementiert worden. Zur Visualisierung des Ergebnisses soll die aufrufende (nicht die berechnende) Funktion in einem Konsole-Fenster nach der obigen Abbildung die Stützstellen ausgeben.

Die Aufgabe besteht darin, das Verfahren auf die anderen benötigten Oktanten zu übertragen; hierfür ist unter http://homepages.thm.de/christ/ das MS-VC-Projekt „`BresenLerp`" vorbereitet worden. Es enthält, neben dem Codefragment zur o.a. Implementierung (im Unterverzeichnis `Src`), den Code zur hier verwendeten Visualisierung mit ASCII-Zeichen und compilierte Versionen der Code-Dateien der Lösung (im Unterverzeichnis `_Libs`). Letztere können bei Bedarf in das Projekt eingebunden werden, um eine Überprüfung eigener Zwischenergebnisse zu ermöglichen. Das lauffähige Ergebnis-Programm (`BresenLerp.exe`) ist im Unterverzeichnis `Demo` enthalten.

Das o.a. Projekt besteht aus drei Quellen- und drei dazugehörigen Header-Dateien:

`conDraw.c` `conDraw.h`	sind vollständig. Sie stellen jene Funktionen bereit, die das „Pixel-Setzen" (ASCII-Zeichen an beliebigen Stellen) und damit das Zeichnen im Konsole-Fenster ermöglichen.
`InterpoLine.c` `InterpoLine.h`	sind „gebrauchsfertig". Dort ist das `main()` und die eigentliche „Infrastruktur" für diese Anwendung: Gestaltung des Fenster-Inhalts, Bedienungslogik, Hilfe-Menü etc..

`interOps.c`
`interOps.h`
	haben den C-Code zur Interpolation: Die Funktion
	`int set_iLerp (int Dx, int y0, int yn, int *yStore)`
	ist nur für den 1. Oktanten implementiert und soll noch vervollständigt werden (ca. 6 Zeilen), damit sie für alle benötigten Oktanten gilt.
	Die Funktion
	`float lerp (float a, float b, float t)`
	enthält den C-Code zur (nicht weiter gebrauchten) linearen Interpolation unter Verwendung von `float`–Variablen.

Diese Übung kann (und sollte) dazu genutzt werden, sich mit der hier verwendeten Visualisierung vertraut zu machen, vor allem mit den wenigen bereitgestellten Funktionen, die alle unverzichtbaren Funktionalitäten erledigen. Ebenso können Hedonist/inn/en (d.h. lustgetriebene Teilnehmer/innen) während der Fertigstellung der geforderten Lösung das eigene Verständnis von der gesamten Aufgabenstellung überprüfen, sich mit dem Unterschied zum Linien-Algorithmus (aus dem Bachelor-Teil dieses Faches) befassen und Fragen wie den folgenden beantworten:

- Warum geht diese Implementierung von einem Definitionsbereich $\in \mathbf{N_0}$ und einem Wertebereich $\in \mathbf{Z}$ aus? Welche Oktanten werden hier gebraucht und warum?
- Welche Oktanten-Grenzen werden einem Oktanten zugerechnet?
- Wie viele unterschiedliche Werte (bzw. wie viele Stufen) hat eine Interpolation, von welcher Höhe und welcher Länge werden sie sein?
- Sind andere Ansätze denkbar, die eine genauere, schnellere Interpolation gewährleisten?