Aufgabe 2: malloc

In dieser Aufgabe geht es um die modulare Programmierung in ANSI-C und die dynamische Speicherverwaltung. Im Verzeichnis */proj/i4sos/pub/aufgabe2/* finden Sie Vorlagen und Testprogramme für die Aufgabe. Kopieren Sie diese Dateien in Ihr Arbeitsverzeichnis. Die Vorlagen enthalten Kommentare, welche ebenfalls zu beachten sind.

a) Bitmap als abstrakter Datentyp

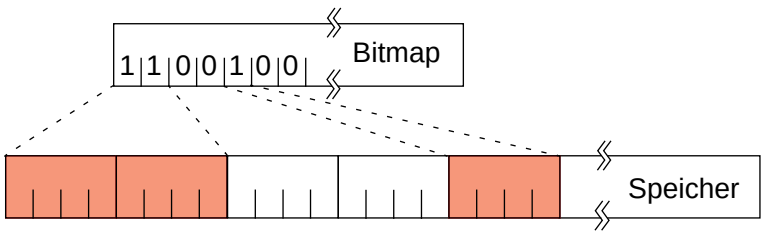
Programmieren Sie eine Bitmap als abstrakten Datentyp in Form eines Moduls. Eine Bitmap soll mit dem Makro **BITMAP_STATIC_INIT(_name_,_size_)** statisch erzeugt werden und stellt eine mit **_size_** festgelegte Anzahl von Bits zur Verfügung. Über die Funktionen **bitmap_mark** und **bitmap_unmark**, sollen die Bits gesetzt bzw. gelöscht werden können. Es soll möglich sein den Zustand eines Bits über die Funktion **bitmap_get** abzufragen und mit der Funktion **bitmap_search_unmarked_range** soll es möglich sein in der gesamten Bitmap nach einem zusammenhängenden Bereich nicht gesetzter Bits zu suchen.

Das Makro und die Funktions-Prototypen finden Sie in der Datei *bitmap.h*. Die Implementierung soll in der Datei *bitmap.c* vorgenommen werden. Erstellen Sie ein dazu passendes *Makefile*, welches ein kleines Testprogramm (*bitmap_test*) erzeugt. Die Quellen für das Testprogramm finden Sie in der Datei *bitmap_test.c*.

b) Dynamische Speicherverwaltung

Erstellen Sie nun eine ANSI-C-konforme dynamische Speicherverwaltung. Als Vorlage soll die Datei *malloc.c* dienen. Der zu verwaltende Speicher soll eine feste Größe (**STATIC_HEAP_SIZE**) besitzen. Die Größe soll zum Übersetzungszeitpunkt festgelegt werden. Erzeugen Sie hierfür einfach ein entsprechend grosses statisches Array.

Der freie Speicher soll von der Speicherverwaltung in gleich grossen Slots (**SLOT_SIZE**) verwaltet werden. Für jeden Slot gibt es dazu in einer statischen Bitmap genau ein Bit. Dieses Bit ist markiert, wenn der Slot belegt ist bzw. unmarkiert wenn der Slot frei ist.



Wird von der Anwendung Speicher angefordert (**malloc()**), so sollen so viele Slots belegt werden wie benötigt werden, um den Speicherbedarf zu decken. Das hierbei oft mehr Speicher belegt wird als wirklich benötigt wird spielt keine Rolle. Beim Freigeben von Speicher (**free()**) sind die Slots in der Bitmap wieder als frei zu markieren.

Beachten Sie, dass die Funktion **free()** keine Größenangabe als Parameter übergeben bekommt. Sie müssen daher die Anzahl der Slots im belegten Speicher mit abspeichern.

Bei Fragen zur Aufgabenstellung bitte Mail an *i4sos@informatik.uni-erlangen.de*