Programmierparadigmen

Wintersemester 2012/13
Torsten Görg, Sandro Degiorgi

2. Übung

14./15./16. November 2012

Organisatorische Hinweise

Geben Sie Ihre Lösungen bzw. die Ihres Abgabe-Teams online über eClaus ab (`https://eclaus.informatik.uni-stuttgart.de/`).
Die Abgaben müssen spätestens bis Mo., 12.11.2012, um 13 Uhr vorliegen.

Aufgabe 2.1 ..... 10 Punkte

Im Hauptspeicher vorliegende Daten können auf dreierlei Weise verwaltet sein: als programmglobale Daten, auf dem Stapel als lokale Daten von Unterprogrammen oder in explizit allokiertem Speicher auf der Halde. In welchen Situationen in Ihrem Programm bzw. unter welchen Randbedingungen wählen Sie welche dieser Varianten zur Speicherung eines Datenobjektes? Welche Probleme können bei Verwendung der Halde auftreten?

Aufgabe 2.2 ..... 18 Punkte

Untersuchen Sie das folgende C++-Programm:

```
struct Artikel {
   int artikelnummer;
   float preis;
};

Artikel* erzeugeArtikel (const int artikelnummer, const float preis)
{
   Artikel *nArt = new Artikel;
   nArt->artikelnummer = artikelnummer;
   nArt->preis = preis;
   return nArt;
}

void aendereArtikel(Artikel *art, const int artikelnummer, const float preis)
{
  if (artikelnummer > 0)   { art->artikelnummer = artikelnummer; }
  if (preis         > 0.0) { art->preis         = preis;         }
}
```

```
void main()
{
   // 1
   Artikel *art1 = new Artikel;
   Artikel *art2 = erzeugeArtikel(4711,23.5);
   Artikel art3  = *art2;
   Artikel *art4 = art2;
   aendereArtikel(art4,0,42.0);
   aendereArtikel(&art3, 110, 0.0);
   delete art1;
   art1 = &art3;
   int artikelNummer = art4->artikelnummer;
   float preis = art1->preis;
   // 2
   delete art2;
   art1 = erzeugeArtikel(1001,9.0);
   art2 = erzeugeArtikel(1010,10.0);
   aendereArtikel(art1,0,11.0);
   aendereArtikel(art2,333,0.0);
   // 3
   art4 = art2;
   delete art1;
   delete art2;
   // 4
   aendereArtikel(art4,4712,999.91);
   delete art4;
   // 5
}
```

1. An welchen Code-Stellen wird in diesem Programm Speicher für Datenobjekte auf dem Stapel oder auf der Halde allokiert? Geben Sie diese Stellen möglichst genau an. Notieren Sie jeweils dabei, für welches Objekt bzw. für welche Variable der Speicher allokiert wird, und kennzeichnen Sie, ob die Allokation auf dem Stapel oder auf der Halde erfolgt. Zeigen Sie ferner die Code-Stellen auf, an denen Werte lediglich auf bereits zuvor existierende Datenobjekte kopiert werden.

2. Welche Werte haben die Variablen `artikelNummer` und `preis` an den mit 1, 2 und 3 gekennzeichneten Stellen?

3. Welche Werte haben die Attribute `artikelnummer` und `preis` von `art4` an den Stellen 4 und 5?

4. Welches Programmverhalten erwarten Sie am Übergang zwischen den Stellen 4 und 5? Warum?

5. Implementieren Sie ein äquivalentes Programm in Java. Bleiben Sie bei dieser Übertragung möglichst nah an der Originalvorlage.

Hinweis: Eine Erklärung, der hier genutzten `reference` und `dereference` Operatoren, finden Sie unter `http://www.cplusplus.com/doc/tutorial/pointers/`.

Aufgabe 2.3 ..... 17 Punkte

Konstruieren Sie für jede der folgenden Freispeicherlisten-Speicherverwaltungsstrategien jeweils ein Beispiel in Form von Ada-Code, bei dem diese Strategie den anderen beiden benannten Strategien überlegen ist (nehmen Sie an, man könnte Ada-Programme wahlweise mit diesen unterschiedlichen Speicherverwaltungsstrategien ausführen), wobei alle auf der Halde allokierten Speicherblöcke bis zum Programmende wieder freigegeben sein müssen. Zeigen Sie, dass die jeweilige Strategie wirklich überlegen ist, indem Sie zu jedem Beispielprogramm jeweils für alle drei Strategien Skizzen der sich verändernden Freispeicherlisten bzw. -ringlisten zeichnen.

1. BestFit
2. FirstFit
3. NextFit

Überlegen sei eine Speicherverwaltungsstrategie einer anderen dann, wenn nach der Ausführung des gesamten Programms die Freispeicherliste bzw. -ringliste weniger Einträge hat als bei der anderen und die Gesamtgröße des fragmentierten Speichers geringer ist. Beachten Sie, dass ein freigegebener Block bei FirstFit und BestFit vorne in die Liste eingehangen wird, während er bei NextFit vor der aktuellen Position in die Ringliste eingefügt und die aktuelle Position auf diesen neuen Eintrag vorgerückt wird.

Aufgabe 2.4 ..... 20 Punkte

In der Vorlesung wurde die Bitvektor-Speicherverwaltungs-Strategie vorgestellt. Diese soll im Rahmen dieser Aufgabe in einer einfachen Eigenimplementierung in Ada nachempfunden werden. Die Schnittstellendefinition eines entsprechenden Moduls finden Sie auf der Veranstaltungs-Homepage in der Datei `bitvector_memory_management.ads`. Für jede Integer-Speicherzelle wird ein Bit mit dem aktuellen Belegungszustand verwaltet. Zur Vereinfachung ist das Modul auf Funktionalitäten zur Allokation und Deallokation beschränkt, ohne Möglichkeiten zum Zugriff auf den allokierten Speicher bereitzustellen.

1. Implementieren Sie entsprechend der gegebenen Schnittstellendefinition die Prozedur `Clear_All`, die die Speicherverwaltung in einen Initialzustand versetzt, in dem alle Speicherzellen als nicht belegt markiert sind.
2. Implementieren Sie entsprechend der gegebenen Schnittstellendefinition die Funktion `Allocate_Block`, mit der ein Speicherblock einer angegebenen Größe allokiert werden kann. Bei jeder Speicheranforderung ist ein geeigneter Speicherblock mit der Bitvektor-Strategie zu ermittelen, wobei beginnend beim Anfang des verwalteten Gesamtspeichers linear nach dem ersten freien Speicherbereich gesucht wird, dessen Größe für die Speicheranforderung ausreichend ist. Als Handle auf den allokierten Speicherblock ist die Indexposition des Blockanfangs im Array `Memory` zurückzugeben oder `-1`, wenn die Speicheranforderung nicht erfüllt werden kann.

3. Implementieren Sie entsprechend der gegebenen Schnittstellendefinition die Prozedur `Deallocate_Block`, die unter Angabe der Indexposition des Blockanfangs und der Blocklänge einen allokierten Speicherblock wieder freigibt, indem die Speicherzellen als unbelegt markiert werden. Beachten Sie auch alle möglichen Fälle inkorrekter Eingaben und beschneiden Sie in diesen Fällen die Angaben jeweils so, dass nur die Zellen freigegeben werden, die wirklich im Gesamtspeicher liegen.

4. Um das Allokierungsverhalten zu testen, schreiben Sie nun eine Test-Prozedur, die solange Blöcke der Größe 77 allokiert, bis eine weitere Allokation nicht mehr möglich ist, und dann die Anzahl der allokierten Speicherblöcke und die Gesamtanzahl der bis dahin allokierten Speicherzellen ausgibt.

5. Schreiben Sie eine weitere Test-Prozedur, um auch die Deallokierung zu testen. Diese soll 100 Speicherblöcke allokieren (beginnend mit dem kleinsten), wobei der `i`-te Speicherblock die Größe `i` hat, und dabei jeweils die Blockgröße und die zugehörige Indexposition des Blocks ausgeben. Danach sind die Blöcke der Größen 10, 20, 30 und 40 zu deallokieren, worauf eine erneute Allokation eines Blocks der Größe 27 erfolgt und die Indexposition dieses Blocks ausgegeben wird.
   Erklären Sie, warum sich genau diese Indexposition für den 27er-Block ergibt.

Aufgabe 2.5 ..... 15 Punkte

Auf der Veranstaltungs-Webseite finden Sie die Datei `Memory_Manager_Strategies.jar`, welche Ihnen die Speicherverwaltungs-Strategien FirstFit, NextFit und BestFit zur Verfügung stellt. Erstellen Sie unter Nutzung des JAR Archivs ein Testprogramm, welches nach Eingabe einer gewünschten Anzahl von Durchläufen nach folgendem Schema wiederholt Speicherblöcke allokiert und freigibt:

```
MemoryManager t = new FirstFit(); // bzw. NextFit() oder BestFit()
for( int cycle=0; cycle < cyclesCount; cycle++) {
   int block1 = t.allocate( 52 );
   int block2 = t.allocate( 70 );
   t.deallocate( block1 );
   t.deallocate( block2 );
}
```

1. Führen Sie das Programm mit 10000, 20000, ..., 80000 (in 5000er-Schritten) Allokierungswiederholungen aus. Tragen Sie die gemessenen Ausführungszeiten, die Gesamtgröße des fragmentierten Speichers (Methode `fragSize()`) und die Anzahl der Blöcke in der Freispeicherliste (Methode `freeListBlocksCount()`) in Diagrammen als Kurven über die Anzahl der Allokierungswiederholungen auf. Das Experiment ist sowohl mit dem FirstFit- als auch mit dem NextFit- und BestFit-Speicherverwaltungsmodul durchzuführen, so dass sich insgesamt 9 Diagramme ergeben. Für die Zeitmessung können Sie `System.currentTimeMillis()` nutzen.

2. Was fällt Ihnen an den zuvor erstellten Diagrammen auf? Welche Unterschiede sind zwischen den Diagrammen zu den verschiedenen Allokierungsstrategien zu erkennen? Erklären Sie Ihre Beobachtungen.