Auffrischungen zu Allgemeine Informatik
3. Juni 2005

Andreas F. Borchert
borchert@mathematik.uni-ulm.de

2. Juni 2005

Aufgabe 7

Bei dem Spiel Fif sind n natürliche Zahlen gegeben. Jeder der beiden Spieler wählt abwechselnd eine der Zahlen aus und fügt sie seiner Sammlung hinzu. Sobald ein Spieler in seiner Sammlung über m Zahlen verfügt, die eine vorgegebene Summe N ergeben, hat er das Spiel gewonnen. Wenn keine mehr auszuwählende Zahl mehr zur Verfügung steht und keiner der Spieler gewonnen hat, geht das Spiel unentschieden aus.

Bei der Standardkonfiguration sind die Zahlen 1 bis 9 gegeben (und somit $n = 9$), es werden jeweils Summen aus drei Zahlen gebildet (also $m = 3$) und die zu erreichende Summe N beträgt 15. Daher auch der Name des Spiels: Fif ist abgeleitet von Fifteen.

So könnte ein Spielverlauf aussehen:

Zur Verfügung stehende Zahlen: $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
Sammlung Spieler #1: $\{\}$
Sammlung Spieler #2: $\{\}$

Das Spiel beginnt damit, daß der erste Spieler die 5 auswählt, während sich der zweite Spieler für die 1 entscheidet. Danach stehen $\{2, 3, 4, 6, 7, 8, 9\}$ noch zur Verfügung. In der zweiten Runde wählt der erste Spieler die 7. Damit sich der erste Spieler nicht im nächsten Zug die 3 krallt, um die 15 zu erreichen, schnappt sich diese der zweite Spieler. Das ergibt folgende Situation:

Zur Verfügung stehende Zahlen: $\{2, 4, 6, 8, 9\}$.
Sammlung Spieler #1: $\{5, 7\}$
Sammlung Spieler #2: $\{1, 3\}$

Der erste Spieler wählt als nächstes die 2. Darauf wählt der zweite Spieler die 8, worauf der erste Spieler die 6 auswählt:

Zur Verfügung stehende Zahlen: $\{4, 9\}$.
Sammlung Spieler #1: $\{5, 7, 2, 6\}$
Sammlung Spieler #2: $\{1, 3, 8\}$

In dieser Situation hat der erste Spieler gewonnen, weil $7 + 2 + 6 = 15$.

Schritt 1

In dem Spiel sind drei lineare Listen für natürliche Zahlen zu verwalten. Eine repräsentiert die noch zur Verfügung stehenden Zahlen und die beiden anderen Listen nehmen die jeweiligen aufgesammelten Zahlen der beiden Spieler auf. Entsprechend sollte zu Beginn eine Datenstruktur für eine lineare Liste von Zahlen definiert werden. Eine einfache Verkettung genügt dafür vollkommen und da die Reihenfolge keine Rolle spielt, benötigen wir nur einen Zeiger auf das eine der beiden Enden der Liste. Es ist allerdings sinnvoll, zwei Record-Typen zu definieren: Einen für ein einzelnes Listen-Element und einen Record, der die gesamte Liste repäsentiert.

Schritt 2

Schreiben Sie eine Prozedur **CreateList**, die eine Liste als **VAR**-Parameter erhält und diese zu einer leeren Liste initialisiert.

Schritt 3

Fügen Sie als nächstes eine Prozedur **AddToList** hinzu, die als Parameter eine Liste und eine Zahl erhält und diese Zahl der Liste hinzufügt.

Schritt 4

Schreiben Sie eine Prozedur **WriteList**, die eine Liste ausgibt, wobei die Zahlen durch Kommata zu trennen sind. Testen Sie anschließend sofort Ihre bislang erstellten Prozeduren, indem Sie **CreateList** aufrufen und die neue

Liste mit **AddToList** etwas füllen. Nach jedem Aufruf von **AddToList** sollten Sie dabei den neuen Stand mit **WriteList** ausgeben.

Schritt 5

Danach wird eine Prozedur **RemoveFromList** benötigt, die als Parameter eine Liste und eine Zahl erhält. Falls die Zahl in der Liste enthalten ist, wird das entsprechende Listen-Element entfernt und der Return-Wert **TRUE** zurückgegeben. Falls jedoch die angegebene Zahl nicht in der Liste vorkommt, ist **FALSE** zurückzuliefern.

Auch diese Prozedur sollten Sie unmittelbar danach testen. Schreiben Sie eine Testprozedur, mit deren Hilfe Sie beliebig viele Zahlen eingeben können, die alle in die Liste eingetragen werden. Sobald eine 0 eingegeben wird, geht es zum zweiten Teil des Programms, bei dem zu entfernende Zahlen eingegeben werden können. Nach jedem Aufruf von **RemoveFromList** sollten Sie den aktuellen Stand mit **WriteList** ausgeben.

Schritt 6

Definieren Sie im nächsten Schritt einen Record-Typ, der ein Spiel repräsentiert. Sie benötigen dazu die drei Listen und die Werte von N und m.

Schritt 7

Um zu einer Startaufstellung zu kommen, gibt es folgende Alternativen:

- Sie lesen aus einer Datei die Zahlen N, m und dann den Anfangsbestand ein. Beispiel für die oben verwendete Standardkonstellation:

```
15
3
1 2 3 4 5 6 7 8 9
```

- Oder Sie lassen sich vom Benutzer nur m vorgeben, setzen

$$N = \frac{1}{2}m(m^2 + 1)$$

  und verwenden die Zahlen 1 bis m^2.

Wählen Sie eine der beiden Alternativen aus und schreiben Sie eine Prozedur **InitGame**, die einen als **VAR**-Parameter übergebenen Spiele-Record initialisiert. Je nach Variante sollte diese Prozedur entweder einen Dateinamen oder m als Parameter erhalten.

Auch hier empfiehlt sich unmittelbar danach ein Test, bei dem eine Startaufstellung erzeugt und unmittelbar danach ausgegeben wird.

Schritt 8

Schreiben Sie eine Prozedur **RunGame**, die eine Partie für zwei Spieler durchführt. Abwechselnd ist ein Zug für den ersten und zweiten Spieler einzulesen und auszuführen. Die Ausführung besteht darin, die gewünschte Zahl aus der Liste der noch zur Verfügung stehenden Zahlen zu entfernen (**RemoveFromList**) und diese in der Liste des aktiven Spielers einzufügen (**AddToList**). Vor jedem Zug sollte der jeweils aktuelle Stand ausgegeben werden, damit dem Spieler klar ist, welche Zahlen noch zur Verfügung stehen und welche bereits in den Besitz der beiden Spieler übergegangen sind.

In diesem Stadium sollten Sie noch auf die Überprüfung, ob jemand bereits gewonnen hat, verzichten und stattdessen die Partie enden, sobald keine Zahlen mehr zur Verfügung stehen.

Schritt 9

Wenn Sie mutig sind, können Sie sich an die Prozedur **Succeeded** wagen, die **TRUE** zurückliefert, wenn in einer gegebenen Liste eine vorgegebene Zahl von Zahlen existiert, die aufsummiert die gewünschte Zahl ergeben. Mit dieser Funktion läßt sich überprüfen, ob jemand gewonnen hat.

Diese Prozedur sollte mit einer lokalen rekursiven Prozedur arbeiten, die statt einer Liste ein Listen-Element erhält und ansonsten die gleichen Parameter hat. Das Ende der Rekursion ist erreicht, wenn nur noch 0 Zahlen fehlen. Falls dann die gewünschte Zahl ebenfalls 0 beträgt, war es erfolgreich, ansonsten nicht. Im rekursiven Falle (also mit noch mindestens einer fehlenden Zahl) sollten beginnend mit dem aktuellen Listen-Element alle durchprobiert werden, um damit die gleiche Prozedur aufzurufen, dann aber mit einer um eins reduzierten Zahl auszuwählender Zahlen und einer Zielzahl, die um den Wert des aktuellen Listen-Elements vermindert worden ist.

Bauen Sie dann diese Prozedur in **RunGame** ein, so daß sofort festgestellt wird, wann einer der Spieler gewonnen hat.