Laboratorium nr 1
Stosy i kolejki

Niniejsze zajęcia poświęcone są implementacji struktur stosowych i kolejkowych zarówno w realizacji tablicowej, jak i listowej.

1.1 Stosy

Stosem nazywamy skończony ciąg elementów tego samego typu zorganizowany według zasady LIFO (ang. *Last In First Out*).
Standardowymi operacjami stosowymi są:
• *InitStack*(*s*) : inicjalizacja pustego stosu
• *Push*(*s; x*) : dodanie elementu *x* do stosu *s*; adres tego elementu jest nowym wierzchołkiem stosu
• *Pop*(*x*) : zdjęcie wierzchołka stosu *s*; jest to funkcja, która zwraca zdejmowany element, a nowym wierzchołkiem stosu jest adres poprzednio wstawionego elementu
• *Top*(*s*) : odczyt wierzchołka stosu; jest to funkcja, która zwraca ten element i nie zmienia struktury stosu
• *Empty*(*s*) : sprawdzenie, czy stos *s* jest pusty; jest to funkcja logiczna, która zwraca True, jeśli stos *s* jest pusty, False jezeli niepusty. Funkcja nie zmienia struktury stosu.

Najpopularniejsze reprezentacje stosów:
• tablicowa : stos reprezentujemy parą (*table; topElem*), gdzie *table* jest *N*–elementową tablicą jednowymiarową, zaś *topElem* jest liczbą całkowitą (wskaźnikiem wierzchołka stosu). Procedura *push* wstawia element na pozycję *topElem*+1, natomiast procedura *pop* usuwa ostatni zapełniony element tablicy (zmniejsza o 1 wartość zmiennej *topElem*).

• listowa : stos reprezentujemy listą *s* (wskaźnik początku listy). Procedura *push* wstawia element na początek listy, zaś *pop* usuwa pierwszy jej element. Adres pierwszego elementu listy jest odpowiednio modyfikowany.

1.2 Kolejki

Kolejka jest skończonym ciągiem elementów tego samego typu zorganizowanym według zasady FIFO (ang. *First In First Out*). Jest to struktura dynamiczna, z której usuwany może być tylko najwcześniej wstawiony element. W kolejce mamy zazwyczaj dostęp do dwóch jej elementów: pierwszego (najwcześniej wstawionego) i ostatniego (najpóźniej wstawionego).
Standardowe operacje kolejkowe:

• *InitQueue*(*Q*) : inicjalizacja pustej kolejki
• *EnQueue*(*Q; x*) : wstawienie elementu *x* do kolejki *Q*
• *DeQueue*(*Q*) : usuwanie pierwszego elementu z kolejki; jest to funkcja,
która zwraca usuwany element
• *Empty*(*Q*) : sprawdzanie, czy kolejka *Q* jest pusta; jest to funkcja logiczna, która zwraca True, jeśli kolejka jest pusta i False w przeciwnym przypadku
• *Front*(*Q*) : odczyt pierwszego elementu kolejki *Q*; jest to funkcja, która
zwraca ten element jednocześnie nie zmieniając struktury kolejki.
Najpopularniejsze reprezentacje kolejki to
• reprezentacja tablicowa: kolejka realizowana jest jednowymiarową tablicą *K*[1*::N*] wraz z dwoma indeksami, *first* i *last*, odpowiadającymi
początkowemu i końcowemu elementowi kolejki.
• listowa: kolejka realizowana jest listą jednokierunkową wraz z dwoma
wskaźnikami, *first* i *last* (są to wskaźniki na pierwszy i ostatni element
listy).

1.3 Zadania

1. Zaimplementować stos liczb całkowitych przy realizacji listowej. Opracować algorytm, który w oparciu o podstawowe operacje stosowe odwróci porządek elementów na stosie.
2. Zaimplementować stos liczb całkowitych przy realizacji tablicowej. Opracować algorytm scalający dwa uporządkowane rosnąco ciągi *A* i *B* liczb
całkowitych umieszczone odpowiednio na stosach *stosA* i *stosB*. Ciąg
wynikowy umieścić na stosie *stosC*.
3. Zaimplementować kolejkę liczb rzeczywistych przy realizacji tablicowej.
4. Zaimpementować kolejkę napisów (ciągi cyfr, dużych i małych liter)
przy realizacji listowej. Opracować algorytm, który nie zmieniając struktury kolejki wyszukuje w tej kolejce zadany element.