# Języki programowania (laboratorium) — Lista zadań nr 2

*dr Miłosz Michalski, 21.II.2024*

**1.** Napisać program wykorzystujący statycznie deklarowaną tablicę `int S[n]` jako stos. Można wygodnie przyjąć, że `S[0]` przechowuje wskaźnik na aktualny szczyt stosu. W szczególności, gdy `S[0]=0`, stos jest pusty. Zaimplementować operacje `push(S,x)`, `pop(S)` i `peek(S)`. Pierwsza z nich dodaje element `x` do stosu, druga zdejmuje z niego wierzchni element, zwracając go jako swoją wartość, `x=pop(S)`, a trzecia zwraca wartość wierzchniego elementu, pozostawiając go na stosie. Potrzebna jest również funkcja logiczna `empty(S)` zwracająca wartość `1`, gdy stos `S` jest pusty lub `0` w przeciwnym wypadku. Próba odwołania się do pustego stosu przez `x=pop(S)` lub `x=peek(S)` ma skutek nieokreślony. Innymi słowy, testowanie czy stos nie jest pusty leży w gestii programisty.

**2.** Podobnie jak poprzednio, zaimplementować w tablicy `Q[n]` kolejkę jednokierunkową z operacjami `pushq(Q,x)`, `popq(Q)` i `peekq(Q)`. Pierwsza z nich dodaje element `x` na koniec kolejki, a `popq(Q)` zdejmuje z niej pierwszy element, `peekq(Q)` zaś zwraca wartość pierwszego elementu bez usuwania go z `Q`. Potrzebna jest także funkcja logiczna `emptyq(Q)` o działaniu opisanym wyżej. Przemyśleć i zaproponować sposób przechowania wskaźników na początek i koniec kolejki w samej tablicy `Q`.

**3.** Wadą użycia statycznej tablicy w roli stosu lub kolejki jest jej ustalony od początku rozmiar i co za tym idzie — możliwość jej przepełnienia. W przypadku próby dodania kolejnego elementu do zapełnionego stosu lub kolejki, funkcja `push` powinna zwracać kod błędu informujący o tym, że operacja nie powiodła się. Dla kolejki należałoby przewidzieć także operację `shift(Q)` przemieszczania danych w tablicy w niższe adresy, gdy zajdzie taka konieczność. Operacja `shift` powinna być wewnętrznym mechanizm implementacyjnym kolejki, niewidocznym dla użytkownika.

**4.** Wzorem poprzednich zadań zaimplementować w tablicy kompletną obsługę kolejki dwustronnej, w której można dodawać i usuwać elementy w każdym z dwóch końców. Podstawowe operacje to: `pushq_front`, `pushq_end`, `popq_front`, `popq_end`, `peekq_front`, `peekq_end` oraz funkcja logiczna `emptyq`. Dobre rozwiązanie powinno zawierać także mechanizm `shift`, jak opisano wyżej.

**5.** Zaimplementować cykliczny bufor (kolejkę cykliczną) w tablicy `int C[n]`. Nowe elementy dodawane są zawsze na końcu tej kolejki, przy czym po zapisaniu ostatniego elementu w tablicy, kolejny zapis odbywa się automatycznie (i niewidocznie dla użytkownika) od jej początku. W przypadku przepełnienia tablicy nowy element nadpisuje początek kolejki, a wskaźnik na początek przesuwa się o jedną pozycję. W ten sposób przy wyczerpaniu miejsca najstarsze elementy są automatycznie nadpisywane przez nowe (nidgy nie dochodzi więc do przepełnienia tablicy). Operacje na kolejce cyklicznej to: `pushc(C)`, `popc(C)`, `peekc(C)`, `emptyc(C)`. Zaproponować najwygodniejszy sposób przechowania wskaźników na początek i koniec cyklicznego bufora w samej tablicy `C`.

**6.** Kolejka priorytetowa to struktura danych, w której każdy element posiada dodatkowo priorytet typu `int`. Dane do takiej kolejki dodawane są w zwykły sposób, natomiast operacje `popq(P)` i `peekq(P)` zwracają zawsze element kolejki o najwyższym priorytecie lub "najstarszy" z nich, gdy jest ich kilka. Oprócz operacji `pushq`, `popq`, `peekq` i `emptyq`, możliwa jest także zmiana priorytetu wskazanego elementu `adjust(P,x,r)` polegająca na odnalezieniu w `P` elementu `x` i zmiany jego priorytetu na wartość `r`. Zaimplementować kolejkę priorytetową z ww. operacjami w statycznej tablicy. Przewidzieć możliwość przepełnienia tablicy i odpowiednie mechanizmy na to reagujące.

**7–11.** Zaimplementować wszystkie ww. konstrukcje za pomocą dynamicznie alokowanych struktur ze wskaźnikami.

**12–16.** Powtórzyć zad. 7–11 wykorzystując wskaźniki na wskaźniki.

**17.** Uzupełnić procedury obsługi kolejek z zad. 8–11 o funkcję lokalizującą element o zadanej wartości `x`, `find(Q,x)`. Zwraca ona wskaźnik na pierwszy licząc od początku element `Q` o wartości `x` lub `NULL`, gdy `x` nie występuje w `Q`.

**18.** Korzystając z funkcji `find` z poprzedniego zadania, napisać funkcje `before(Q,x,y)` oraz `behind(Q,x,y)`, które wstawiają do `Q` nowy element o wartości `y` bezpośrednio przed lub odp. za znalezionym elementem `x`. Jeśli `x` nie występuje w `Q`, `y` trafia standardowo na koniec kolejki.

**19,20.** Powtórzyć zad. 17 i 18 z wykorzystaniem wskaźników na wskaźniki.

**21–23.** Zrealizować stos, kolejkę jedno- i dwustronną z zadań 1–4 w dynamicznie alokowanej tablicy. Oprócz procedury `shift`, rozwiązanie powinno zawierać mechanizm automatycznej, niewidocznej dla użytkownika realokacji tablicy w przypadku jej przepełnienia. Dobre rozwiązanie powinno przewidzieć również możliwość zmniejszenia tablicy, gdy wcześniej alokowana jej część przestaje być potrzebna.