
	Uniwersytet Technologiczno-Przyrodniczy Im. J. J. Śniadeckich w Bydgoszczy Wydział Telekomunikacji, Informatyki i Elektrotechniki Zakład Techniki Cyfrowej		
Przedmiot	Algorytmy i Struktury Danych		
Prowadzący			
Temat	Kolejka		
Student			
Nr ćw.	10	Data wykonania	
Ocena		Data oddania spr.	

1. Cel ćwiczenia

Celem ćwiczenia jest poznanie struktury danych, jaką jest kolejka, jej implementacji oraz możliwego zastosowania.

2. Informacje podstawowe

2.1. Kolejka

Jest to struktura danych, która przypomina kolejkę w sklepie. Osoba, która pierwsza przyszła, pierwsza zostanie obsłużona – FIFO (ang. *first in, first out*). Poniżej zamieszczono kod implementacji kolejki stringów za pomocą tablicy (do funkcji *queue* podaje się wielkość tablicy). Taka implementacja ma pewną wadę, gdyż jest ona ograniczona deklarowaną wielkością tablicy. W kolejce możemy przechowywać różne rodzaje danych, np. liczby, teksty, struktury itp.

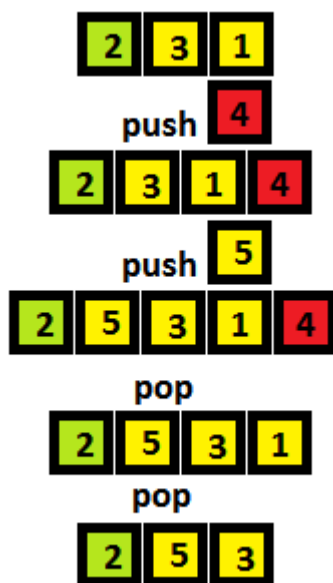
```
void queue(int s) {
    string *array = new string[s];
    cout << "Podaj jedna z instrukcji:\n"
         << "d - aby dodac napis do kolejki\n"
         << "u - aby usunac napis z kolejki\n"
         << "w - aby wyswietlic zawartosc kolejki\n"
         << "x - aby zakonczyc\n";
    char instrukcja; //info od uzytkownika
    bool loop = true; //czy zapetlac
    int elements = 0; //ilosc elementow
    while (loop) {
        cin >> instrukcja;
        switch (instrukcja) {
            case 'd': //dodawanie
                if (elements == s) {
                    cout << "kolejka jest pelna!\n";
                }
                else {
                    string str;
                    cout << "podaj napis\n";
                    cin >> str;
                    for (int i = elements-1; i >= 0; i--) {
                        array[i + 1] = array[i];
                    }
                    array[0] = str;
                    elements++;
                    cout << "dodano!\n";
                }
            }
        }
        break;
    }
```

```

        case 'u': //usuwanie
            if (elements == 0) {
                cout << "kolejka jest pusta!\n";
            }
            else {
                elements--;
                cout << "usunieto: " << array[elements] << "!\n";
                array[elements] = "";
            }
            break;
        case 'w': //wyswietlanie
            if (elements == 0) {
                cout << "kolejka jest pusta!\n";
            }
            else {
                cout << "aktualna kolejka:\n";
                for (int i = 0; i < elements; i++) {
                    cout << array[i] << " ";
                }
                cout << "\n";
            }
            break;
        case 'x': //wyjscie
            loop = false;
            break;
        default: //nieznana instrukcja
            cout << "Podaj jedna z instrukcji:\n"
                << "d - aby dodac napis do kolejki\n"
                << "u - aby usunac napis z kolejki\n"
                << "w - aby wyswietlic zawartosc kolejki\n"
                << "x - aby zakonczyc\n";
            break;
    }
}

```

2.2. Kolejka priorytetowa



Jest ona specjalnym przypadkiem kolejki, gdyż każdy element ma ustawiony priorytet. Elementy nie są wstawiane na koniec kolejki, ale zgodnie ze swoim priorytetem. Dodawanie (polecenie *push*) i usuwanie elementów (polecenie *pop*) zostało przedstawione na rysunku obok, gdzie priorytety oznaczono kolorami (czerwony to najwyższy priorytet, żółty – średni, zielony – najniższy).

Element „4” trafił na początek kolejki, bo nie było w niej żadnego elementu czerwonego.

Element „5” trafił na swoje miejsce, gdyż przybył najpóźniej ze wszystkich elementów żółtych, a ma wyższy priorytet niż element „2”, który jest zielony.

3. Przebieg ćwiczenia

3.1. Zadanie 1.

Przeanalizować wyżej przedstawiony kod. Na jego podstawie napisać własny program implementujący kolejkę, nie należy jednak wykorzystywać tablicy.

Skopiować treść rozwiązania, aby umieścić je w sprawozdaniu.

3.2. Zadanie 2.

Zmienić napisany kod z zadania 1, tak aby implementował kolejkę priorytetową przedstawioną na rysunku.

Skopiować treść rozwiązania, aby umieścić je w sprawozdaniu.

4. Sprawozdanie

Sprawozdanie z laboratorium powinno zawierać:

- wypełnioną tabelę z początku instrukcji
- kody programów będących rozwiązaniami wszystkich zadań wraz z komentarzami,
- wnioski.