**Ćwiczenia ASD nr 9 (21/12/2020)**

**Nowe pojęcia**

**-stos i jego budowa**

-tablicowa implementacja stosu-tablica S[1..n]

-atrybut S.top jest indeksem ostatnio dodanego elementu

-początkowa wartość S.top jest równa 0

-stos przechowuje elementy S[1], S[2], ..., S[S.top]

-stos jest pusty, jeżeli S.top == 0

-podstawowe operacje na stosie (STACK\_EMPTY, PUSH, POP)

**-kolejka i jej budowa**

-tablicowa implementacja kolejki

-tablica Q[1..n]

-atrybuty Q.head, Q.tail, Q.length

-Q.head jest indeksem elementu na początku kolejki

-Q.tail jest indeksem pozycji, na którą można wstawić kolejny element

-Q.length jest liczbą elementów kolejki

-kolejka składa się z elementów Q[Q.head],Q[Q.head+1],..., Q[Q.head+length-1]

-kolejka jest pusta, jeżeli Q.head == Q.tail

-kolejka jest pełna, jeżeli Q.head == Q.tail + 1

-początkowo Q.head == Q.tail == 1

-podstawowe operacje na kolejce (QUEUE\_EMPTY, ENQUEUE, DEQUEUE)

**Definicje podstawowych operacji**

**Operacje na stosie**

function STACK\_EMPTY(S)

if S.top == 0

then return true

else return false

procedure PUSH(S,x)

S.top = S.top + 1

S[S.top] = x

Return

function POP(S)

S.top = S.top -1

return S[S.top + 1]

**Operacje na kolejce**

function QUEUE\_EMPTY(Q)

if Q.head == Q.tail

then return true

else return false

procedure ENQUEUE(Q,x)

Q[Q.tail] = x

if Q.tail == Q.length

then Q.tail = 1

else Q.tail = Q.tail + 1

return

procedure DEQUEUE(Q)

x = Q[Q.head]

if Q.head == Q.length

then Q.tail = 1

else Q.head = Q.head + 1

return x

**A1. Zadania dotyczące stosu**

1. Dany jest pusty stos S zaimplementowany w tablicy S[1..6]. Zilustruj na rysunkach efekt wykonania następującego ciągu operacji: PUSH(S,4), PUSH(S,1), PUSH(S,3), POP(S), PUSH(S,8) oraz POP(S)
2. Uzupełnij funkcje POP(S) oraz procedurę PUSH(S,x) o wykrywanie błędów przepełnienia oraz niedomiaru stosu.
3. Napisz procedurę tworzącą z danych dwu stosów, S1 i S2, jeden nowy stos, S.
4. Napisz procedurę odwracającą porządek elementów przechowywanych na stosie S.

**A2. Zadania dotyczące kolejki**

1. Dana jest kolejka Q zaimplementowana w tablicy Q[1..6]. Zilustruj na rysunkach efekt wykonania następującego ciągu operacji: ENQUEUE(Q,4), ENQUEUE(Q,1), ENQUEUE(Q,3), DEQUEUE(Q), ENQUEUE(Q,8) oraz DEQUEUE(Q).
2. Uzupełnij procedury ENQUEUE i DEQUEUE o wykrywanie błędów przepełnienia oraz niedomiaru kolejki.
3. Napisz procedurę tworzenia kolejki Q z elementów przechowywanych na stosie S.
4. Napisz procedurę tworzenia z elementów danej kolejki Q1 nowej kolejki Q2, wktórej elementy są w odwrotnym porządku niż w Q1.

**B. Zadania domowe ZD.**

1. Napisz funkcje zwracającą największy element przechowywany na stosie S.ZD.

2. Zaimplementuj operacje na kolejce Q wykorzystując dwa stosy, S1 i S2.

**Ćwiczenia ASD nr 10 (04/01/2021)**

**Nowe pojęcia**

-lista i jej budowa, czoło i ogon listy, wskaźnik NIL

-lista jednokierunkowa L (pojedynczo wiązana):

-wskaźnik L.head, klucz x.key, wskaźnik x.next

-czoło: L.head == NIL, ogon: x.next == NIL

-lista dwukierunkowa (podwójnie wiązana)

-wskaźnik L.head, klucz x.key, wskaźniki x.prev, x.next

-czoło: x.prev == NIL, ogon: x.next == NIL

-podstawowe operacje na liście (LIST\_SEARCH, LIST\_INSERT, LIST\_DELETE

**Definicje podstawowych operacji**

LIST\_SEARCH(L,k)

x = L.head

while (x <> NIL) and (x.key <> k) do

x = x.next

return x

LIST\_INSERT(L,x)

x.next = L.head

if L.head <> NIL

then L.head.prev = x

L.head = x

x.prev = NIL

return

LIST\_DELETE(L,x)

if x.prev <> NIL

then x.prev.next = x.next

else L.head = x.next

if x.next <> NIL

then x.next.prev = x.prev

return

**A.Zadania dotyczące list**

1. Napisz funkcję LENGTH(L) zwracającą liczbę elementów listy L

2. Napisz procedurę LIST\_INSERT(L,x,k), która wstawia do listy L element x jako k-ty.

3. Napisz inną postać procedury LIST\_INSERT(L,x), która wstawia element x na koniec listy L.

4. Napisz procedurę,która odwróci kolejnośćwszystkich elementów w liście dwukierunkowej.

**B.Zadania domowe**

ZD. 1. Zaimplementuj operacje na stosie wykorzystując listę jednokierunkową.

ZD. 2. Zaimplementuj operacje na kolejce wykorzystując listę jednokierunkową.

**Ćwiczenia ASD nr 9 (21/12/2020)**

**Nowe pojęcia**

-operacja elementarna, operacja dominująca

-dane wejściowe algorytmu i ich rozmiar

-złożoność obliczeniowa (koszt) wykonania instrukcji podstawienia

-złożoność obliczeniowa (koszt) wykonania zdania warunkowego

-złożoność obliczeniowa (koszt) wykonania pętli sterowanej licznikiem

-złożoność obliczeniowa (koszt) wykonania pętli sterowanej warunkiem

-pesymistyczna złożoność obliczeniowa (koszt) wykonania algorytmu nierekurencyjnego

-pesymistyczna złożoność obliczeniowa (koszt) algorytmu rekurencyjnego

-złożoność pamięciowa algorytmu

Zasady obliczania złożoności (pesymistycznej)

**ad. złożoność obliczeniowa**

-koszt wykonania 1 operacji elementarnej jest równy 1

-koszt wykonania procedur i funkcji standardowych jest równy 1

-koszt wykonania zdania warunkowego jest równy maksimum z kosztów części po 'then' oraz 'else'

-koszt wykonania pętli sterowanej licznikiem jest równy iloczynowi kosztu wykonania wnętrza pętli razy liczba wykonań tej pętli

-koszt wykonania pętli sterowanej warunkiem jest równy iloczynowi kosztu wykonania wnętrza pętli razy maksymalna liczba wykonań tej pętli

-koszt wykonania algorytmu jest równy sumie kosztów wykonania wszystkich jednostek składowych tego algorytmu

-rząd kosztu wykonania algorytmu jest równy maksimum z kosztów wykonania wszystkich jednostek składowych tego algorytmu

-koszt wykonania algorytmu rekurencyjnego jest równy kosztowi będącemu rozwiązaniem równania rekurencyjnego opisującego ten algorytm

**ad. złożoność pamięciowa**

-złożoność pamięciowa algorytmu jest równa liczbie jednostek pamięci (komórek) potrzebnych do zapamiętania wszystkich struktur danych algorytmu-zapamiętanie zmiennej dowolnego typu predefiniowanego wymaga 1 jednostki pamięci

-zapamiętanie tablicy elementów dowolnego typu predefiniowanego wymaga 1 jednostki pamięci razy iloczyn wszystkich wymiarów tej tablicy

**A1. Zadania na badanie złożoności algorytmów (nierekurencyjnych)**

Zad. 1. Napisać algorytm obliczający f(x) dla rzeczywistego x, jeżeli f(x)=-x dla x<0 oraz f(x)=x^2\*exp(x) dla x>=0. Obliczyć rząd złożoności czasowej i pamięciowej tego algorytmu.

Zad. 2. Obliczyć rząd złożoności czasowej i pamięciowej algorytmu znajdowania

a) pola trójkąta za pomocą wzoru Herona

b) pierwiastków rzeczywistych równania kwadratowego

Zad. 3. Obliczyć dokładną złożoność obliczeniową i pamięciową algorytmu obliczającego x^n a) uwzględniając operacje składowe pętli b) pomijając operacje składowe pętli. Porównać rzędy złożoności obliczeniowej przypadków a) i b) oraz sformułować odpowiedni wniosek.

Zad. 4. Napisać algorytm obliczający iloczyn skalarny dwu n-wymiarowych wektorów x i y. Obliczyć rząd złożoności czasowej i pamięciowej tego algorytmu.

Zad. 5. Dana jest kwadratowa tablica A[1..n,1..n] liczb całkowitych. Napisać algorytm obliczający kwadrat sumy wszystkich elementów nieparzystych tej tablicy. Obliczyć rząd złożoności obliczeniowej i pamięciowej tego algorytmu.

Zad. 6. Obliczyć rząd złożoności czasowej i pamięciowej algorytmów sortowania przez a) wybieranie b) wstawianie A2. Zadania na badanie złożoności algorytmów (rekurencyjnych)

Zad. 7. Obliczyć rząd złożoności czasowej i pamięciowej rekurencyjnej wersji algorytmu obliczania n-tej liczby Fibonacci'ego.

Zad. 8. Obliczyć rząd złożoności czasowej i pamięciowej rekurencyjnego algorytmu

a) obliczania sumy składowych tablicy A[1..n]

b) znajdowania największego elementu w tablicy A[l..p]

**B. Zadanie domowe**

ZD. Przejrzeć pseudokody wszystkich poznanych dotąd algorytmów i dla każdego z nich obliczyć rząd złożoności czasowej i pamięciowej

**Rozwiązania zadań ze złożoności algorytmów (ćw ASD nr 9)**

**Ad. Zad. 1**

0: F(x)

1: if x<0

2: then return -x

3: else return x\*x + exp(x)

operacje: 1 negacja, 1 mnozenie, 1 exp(x)

pamiec: 1 argument, 0 zmiennych pomocniczych

Zlozoność: T(n)=O(1), jeżeli exp(x) jest op. Elementarną

M(n)=(1)

**Ćwiczenia ASD nr 11(11/01/2021)**

**Nowe pojęcia**

-Kopiec binarny i jego budowa

-Kopiec binarny typu max (typu min)

-Operacje na kopcu binarnym typu max (min):

-przywracanie własności kopca typu max (min)

-złożoność czasowa rzędu O(log n)

-wstawienie elementu do kopca:

-wstawiamy element na pozycje n+1 i przywracamy własność kopca typu max (min)

-złożoność czasowa rzędu O(log n)

-odczytanie elementu największego (najmniejszego):

-odczytujemy element w korzeniu drzewa

-złożoność czasowa rzędu O(1)

-usuniecie elementu największego (najmniejszego)

-zapamiętujemy wartość w korzeniu

-wstawiamy do korzenia element z pozycji n

-przywracamy własność kopca typu max (min)

-złożoność czasowa rzędu O(log n)

-utworzenie nowego kopca

**Definicje podstawowych operacji na kopcu typu max**

**-przywracanie własności kopca typu max**

MAX\_HEAPIFY (A, i)

left = 2\*i

right = 2\*i + 1

if (left <= A.heap-size)and (A[left] > A[i])

then largest = left

else largest = i

if (right <= A.heap-size)and (A[right] > A[largest])

then largest = right

if largest <> i

then A[i] <-> A[largest]

MAX\_HEAPIFY(A, largest)

Return

-**zwiekszenie wartosci klucza**

HEAP-INCREASE-KEY(A,i,key)

if key < A[i]

then write"Blad"A[i] = key

while (i > 1)and (A[floor(i/2)] < A[i]) do

A[i] <-> A[floor(i/2)] i = floor(i/2)

Return

-**wstawienie elementu do kopca**

MAX\_HEAP\_INSERT (A, key)

A.heap-size = A.heap-size + 1

A[A.heap-size] = -nieskonczonosc

HEAP-INCREASE-KEY(A,A.heap-size,key)

Return

-**odczytanie elementu największego**

HEAP\_MAXIMUM(A)

return A[1]

HEAP\_EXTRACT\_MAX (A)

if A.heap-size < 1

then write"Kopiec pusty"

max = A[1]

A[1] = A[A.heap-size]

A.heap-size = A.heap-size -1

MAX\_HEAPIFY (A, 1)

return max

**-budowanie kopca**

BUILD\_MAX\_HEAP (A)

A.heap-size = A.length

for i = floor(A.length/2) downto 1 do

MAX\_HEAPIFY(A, i)

Return

**A. Zadania na ćwiczenia**

1. Czy tablica [25,20,23,14,7,18,19,8,3,2,0] jest kopcem typu max ?

2. Jaka jest największa i najmniejsza możliwa liczba elementów w kopcu o wysokości h?

3. Zilustruj działanie procedury MAX-HEAPIFY(A,3) dla tablicy A=[27,17,3,16,13,10,1,5,7,12,4,8,9].

**B. Zadania domowe**

ZD. 1. Czy tablica [3,4,5,20,30,28,21,23,31,33,41,9] jest kopcem typu min?

ZD. 2. Napisz pseudokody odpowiedników operacji MAX\_HEAPIFY, MAX\_HEAP\_INSERT, HEAP\_MAXIMUM, HEAP\_EXTRACT\_MAX oraz BUILD\_MAX\_HEAP dla kopca typu min.

ZD. 3. Zilustruj działanie procedury BUILD-MIN-HEAP(A) dla tablicy A=[5,3,17,10,74,19,6,23,9].

**Laboratorium ASD(11/01/2021)**

**Biblioteka STL**

**Uwagi dot. wykorzystywania kontenera vector:**

-rozmiar wektora nie jest ograniczony

-funkcja push\_back dodaje element na końcu wektora

-funkcja clear usuwa wszystkie elementy wektora

-funkcja empty sprawdza czy wektor jest pusty

-funkcja size zwraca rozmiar (=liczbę elementów) wektora

**Uwagi dot. wykorzystywania kontenera list:**

-wielkość listy nie jest ograniczona

-funkcja push\_back dodaje element na końcu listy

-funkcja push\_front dodaje element na początku listy

-funkcja front zwraca wartość elementu z czoła listy

-funkcja pop\_front zdejmuje element z czoła listy

-funkcja clear usuwa wszystkie elementy listy

-funkcja empty sprawdza czy lista jest pusta

-funkcja size zwraca rozmiar (=liczbę elementów) listy

-funkcja sort sortuje elementy listy

**Wskaźniki**

-wskaźnik to zmienna przechowująca adres w pamięci

-wskaźnik może wskazywać na zmienną bądź stałą określonego typu

-symbol NULL oznacza stałą wskaźnikową złożoną z zer (wskaźnikowe zero)

-operator &służy do pobierania adresu zmiennej

-operator new służy do przydzielenia (alokacji) pamięci nowego wskaźnika

-operator delete służy do zwalniania przydzielonej pamięci

-symbol -> służy do odwoływania się do pól struktury wskazywanej przez wskaźnik

-wykorzystując wskaźniki można tworzyć dynamiczne tablice

-wskaźnik może być wykorzystany jako parametr funkcji

**A. Zadania do wykonania w trakcie laboratorium**

1. Napisz program w C++ tworzący zmienną x typu double oraz zmienną wsk typu wskaźnikowego na wartości typu double, nadający jej wartość oraz zmieniający tę wartość z wykorzystaniem tego wskaźnika.

2. Napisz program w C++ tworzący strukturę o nazwie punkt o polach x, y typu double oraz zmienną wskaźnikową wsk wskazującą na tego typu strukturę. Nadaj wartości polom x,y, a następnie wykorzystując operator new utwórz nowy wskaźnik p i za jego pomocą wypisz na ekranie wartości pól x, y.

3. Napisz program w C++ który wykorzystując wskaźniki zdefiniuje nową dynamiczna tablicę elementów typu int.

4. Napisz program w C++ w którym wykorzystana zostanie jakaś funkcja z jednym argumentem typu wskaźnikowego.

**Ćwiczenia ASD nr 12 (18/01/2021)**

**Nowe pojęcia**

Drzewo

-drzewo binarne

-budowa węzła drzewa: key, left, right, p

-odwiedzanie wszystkich wierzchołków drzewa

-inorder

-preorder

-postorder

-drzewo BST, własność drzewa BST

-dla każdego węzła w drzewie BST zachodzi własność: wartości kluczy węzłów leżących w lewym poddrzewie węzła są niewiększe (mniejsze lub równe) wartości klucza danego węzła, natomiast wartości kluczy węzłów leżących w prawym poddrzewie węzła są niemniejsze (większe lub równe) wartości klucza danego węzła

**Definicje podstawowych operacji**

procedure PREORDER (T)

p=T.root

if p <> nil

then write p.key

PREORDER(p.left)

PREORDER(p.right)

Return

**A. Zadania na ćwiczenia**

1. Napisz procedury odwiedzania wszystkich wierzchołków drzewa metodaa) INORDER b) POSTORDER

2. Wypisz numery wszystkich wierzchołków drzewa (1,2), (1,3), (2,4), (2,5), (3,6) odwiedzanych metoda PREORDER.

3. Dane jest drzewo (1,2), (1,3), (2,4), (2,5), (5,6), (5,7), (7,8),(7,9), (8,10). Wypisz numery wszystkich wierzchołków tego drzewa odwiedzanych metoda

a) PREORDER

b) INORDER

c) POSTORDER

4. Zbuduj drzewo BST z tablicy A=[4,9,11,5,7,12,32,2,1,10]

5. Napisz funkcje zwracającą sumę najmniejszego i największego klucza w drzewie BST.

**B. Zadania domowe**

ZD. 1. Dane jest drzewo (F,B), (F,G), (B,A), (B,D), (D,C), (D,E), (G,I), (I,H). Wypisz numery wszystkich wierzchołków tego drzewa odwiedzanych metoda

a) PREORDER

b) INORDER

c) POSTORDER

ZD. 2. Zbuduj drzewo BST z tablicy A=[50,76,17,23,9,54,72,19,14,12,67]

ZD. 3\*. Czy można skonstruować odpowiedniki metod PREORDER, INORDER oraz POSTORDER dla dowolnych drzew? Odpowiedź uzasadnij.

ZD. 4. Napisz funkcję zwracającą kwadrat klucza poprzednika największego klucza w drzewie BST.