Język C++ zajęcia nr 7

Dynamiczna alokacja/zwalnianie pamięci przez konstruktor/destruktor

Jeżeli konstruktor obiektu przydziela pamięć operatorem **new**, to destruktor tego obiektu powinien zwalniać przydzielony obszar przy pomocy operatora **delete**.

Wprowadź poniższy program, ZINTERPRETUJ POSZCZEGÓLNE ELEMENTY KODU ŹRÓDŁOWEGO!!! Skompiluj i uruchom program.

Zwalnianie pamięci przez destruktor:

```
#include <iostream>
using namespace std;

class tablica
{
   int rozmiar;
   int* poczatek;
 public:
   tablica(int n);                          // deklaracja konstruktora
   ~tablica();                              // deklaracja destruktora
   void drukuj();
   void wpisz(int n,int x);
   int odczytaj(int n);
};

tablica::tablica(int n)                     // definicja konstruktora
{
   poczatek=new int[rozmiar=n];             // przydzielenie pamięci wolnej
   for(int i=0;i<rozmiar;i++)poczatek[i]=0;
}

tablica::~tablica()                         // definicja destruktora
{
   delete[]poczatek;                        // zwolnienie przydzielonej pamięci
}

void tablica::drukuj()
{
   cout<<endl;
   for(int k=0;k<rozmiar;k++)cout<<poczatek[k]<<" ";
}
```

```
void tablica::wpisz(int n,int x){poczatek[n]=x;}
int tablica::odczytaj(int n){return poczatek[n];}

int main()
{
   int i;
   tablica tab(20);                    // definicja obiektu lokalnego w funkcji
   tab.drukuj();
   tab.wpisz(0,1);
   tab.wpisz(1,1);
   for(i=2;i<20;i++)
      tab.wpisz(i,tab.odczytaj(i-1)+tab.odczytaj(i-2));
   tab.drukuj();
   {
      tablica tab(15);          // definicja obiektu w bloku lokalnym
      tab.wpisz(0,1);
      for(i=1;i<15;i++)tab.wpisz(i,2*tab.odczytaj(i-1));
      tab.drukuj();
   }                                   // automatyczne uruchomienie destruktora
   tab.drukuj();
}                                      // automatyczne uruchomienie destruktora
```

Oczekiwane wyniki:

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
6765

1 2 4 8 16 32 64 128 256 512 1024 2048 4096 8192 16384

1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
6765
```

Program CPP07.

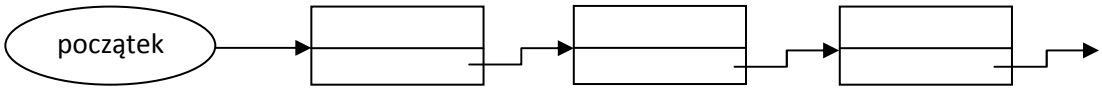
Na początku w komentarzu wpisz drukowanymi literami IMIĘ i NAZWISKO.

Program tworzy i modyfikuje na bieżąco jednokierunkową listę reprezentującą kolejkę pacjentów do lekarza. Długość kolejki jest nieograniczona. Należy przewidzieć następujące operacje:

- dopisanie pacjenta na koniec listy,

- usunięcie pierwszego pacjenta z listy,

UWAGA: lista jest budowana/modyfikowana przez odpowiednie funkcje składowe klasy pacjent.



W celu realizacji programu:

1. Zdefiniuj klasę pacjent zawierającą daną *nazwisko* (20-elementowa tablica znaków). Dla tej klasy należy też zdefiniować odpowiedni wskaźnik (na następny element listy), oraz konstruktor i destruktor.

2. Zdefiniuj i zainicjuj wskaźniki na początkowy i końcowy element listy (są to wskaźniki na obiekty klasy pacjent).

3. Konstruktor powinien utworzyć nowy obiekt klasy pacjent, następnie poprosić o podanie nazwiska nowego pacjenta, wprowadzać tekst nazwiska do odpowiedniej danej składowej nowo utworzonego obiektu, przypisać wskaźnikowi wartość NULL, oraz umieścić ten obiekt na końcu listy (a więc zmodyfikować wskaźnik dotychczas ostatniego elementu oraz wskaźnik **koniec**).

4. Destruktor powinien spowodować usunięcie pierwszego elementu z listy (pacjent wchodzi do gabinetu).

5. Należy zdefiniować **funkcję drukuj**, która drukuje (jedno pod drugim) kolejne nazwiska pacjentów z listy.

W scenariuszu w funkcji main należy przewidzieć następujące zdarzenia (realizowane w pętli):

- klawisz **n** – przyjęcie nowego pacjenta na listę – na koniec listy (*wprowadzenie nazwiska z klawiatury, do realizacji tej operacji użyj odpowiedniego konstruktora),*

- klawisz **p** – wejście pierwszego na liście pacjenta do gabinetu i usunięcie go z kolejki,

- klawisz **q** – wyjście z pętli i zakończenie programu.

Po każdym zdarzeniu (z wyjątkiem **q**) następuje wydrukowanie listy (nazwiska poszczególnych pacjentów w kolejnych wierszach).

Wskazówka: w celu dodawania/usuwania elementów listy stosuj operatory **new** i **delete.** Ponieważ jedynymi metodami dostępu do elementów listy (obiektów) są wskaźniki, zadbaj aby po każdej operacji nie utracić wskaźnika do żadnego elementu i aby je właściwie przechowywać.

Program należy dostarczyć poprzez platformę Moodle we wskazanym terminie.

Sugestie dot. kodu źródłowego programu:

```
#include <iostream>
#include <conio.h>
using namespace std;

class pacjent
{
public:
    ……..   // pole nazwisko
    ……..   // wskaźnik na następny element
    pacjent();          //  deklaracja konstruktora
    ~pacjent();        // deklaracja destruktora
};

pacjent *poczatek=NULL, *koniec=NULL;   // wskaźniki na pocz i końc. element

// definicja konstruktora
    ……

// definicja destruktora
    ……

void drukuj()           // definicja funkcji drukującej listę
   ……...

int main()
{
    char c;
    cout<<"Wybierz operacje (n,p,q): ";
    while ((c=getch())!='q')
      { switch (c)
          {
          ……..
          }
        cout<<"\nWybierz operacje (n,p,q): ";
      }; // koniec petli while
    cout << "Koniec pracy!" << endl;
}
```