Laboratorium Informatyki II
Ćwiczenie 3

1 Dynamiczna alokacja danych

W języku C++ dynamiczną alokację pamięci można realizować za pomocą narzędzi stosowanych w języku C lub dedykowanych operatorów `new` i `delete`. Funkcje `calloc()`, `malloc()`, `realloc()` i `free()` wymagają podpięcia do programu biblioteki `<cstdlib>`. Ich implementacja w kodzie C++ jest niemal identyczna jak w C, w przypadku funkcji deklaracyjnych konieczne jest umieszczenie przed nazwa funkcji rzutowania na właściwy wskaźnik, jak pokazano poniżej.

```
int *tab1, *tab2;
tab1 = (int*)malloc(5*sizeof(int));
tab2 = (int*)calloc(5,sizeof(int));
tab1 = (int*)realloc(tab1,10*sizeof(int));
free(tab1); free(tab2);
```

W kodzie pisanym C++ do dynamicznej alokacji pamięci dedykowane są operatory `new` (odpowiednik `calloc()`) i `delete` (odpowiednik `free()`). Poniższy kod pokazuje proces dynamicznej alokacji zmiennej, tablicy i tablicy dwuwymiarowej.

```
#include <iostream>
using namespace std;
int main()
{
    int *x, *tab, **tab2d;
    x = new int;
    *x = 20;
    cout << "x = " << *x << endl;
    delete x;
//-------------------------------
    tab = new int[3];
    cout << "tab: { ";
    for(int i=0;i<3;i++){
        tab[i] = i+3;
        cout << tab[i] << " ";
    }
    cout << "}" << endl;
    delete [] tab;
//-------------------------------
    tab2d = new int *[3];
    for(int i=0;i<3;i++){
```

```
        tab2d[i] = new int[4];
        for(int j=0;j<4;j++)
            tab2d[i][j] = 3*i-2*j;
    }
    cout << "tab2d:" << endl;
    for(int i=0;i<3;i++){
        for(int j=0;j<4;j++){
            cout.width(3);
            cout << tab2d[i][j] << " ";
        }
        cout << endl;
    }
    for(int i=0;i<3;i++)
        delete[] tab2d[i];
    delete[] tab2d;
}
```

W języku C++ nie ma dostępnego narzędzia zmiany rozmiaru tablicy jakim jest funkcja `realloc()` znana z języka C. Konieczne jest opracowanie własnej funkcji. Przykład funkcji re lokującej (`resize()`) pokazano poniżej.

```
#include <iostream>
int* resize(int *,int*, int);
using namespace std;
int main()
{
    int n=3, *tab;
    tab = new int[n];
    cout << "tab: { ";
    for(int i=0;i<n;i++){
        tab[i] = i+3;
        cout << tab[i] << " ";
    }
    cout << "}" << endl;
    tab = resize(tab,&n,5);
    cout << "nowa tab: { ";
    for(int i=0;i<n;i++)
        cout << tab[i] << " ";
    cout << "}" << endl;
    delete [] tab;
}
int* resize(int *T,int *old, int nowa)
{
    int n;
    int *tab = new int[nowa];
    if(nowa<*old)
        n=nowa;
    else
        n=*old;
    for(int i=0;i<n;i++)
        tab[i] = T[i];
```

```
    delete[] T;
    *old = nowa;
    return tab;
}
```

Kod funkcji można także oprzeć na funkcji `memcpy()` lub `memmove()`, realizując za ich pomocą przeniesienie danych pomiędzy tablicami, jak pokazano poniżej.

```
int* resize(int *T,int *old, int nowa)
{
    int n;
    int *tab = new int[nowa];
    if(nowa<*old)
        n=nowa;
    else
        n=*old;
    memmove(tab,T,n*sizeof(int)); //lub memcpy(tab,T,n*sizeof(int));
    delete[] T;
    *old = nowa;
    return tab;
}
```

2 Konstruktory i destruktor

Projektując klasę często zachodzi konieczność aby wybrane cechy plasy miały ustawione wartości inicjujące. Zasady języka C++ nie pozwalają na inicjowanie cech klasy. Konieczne więc staje się ustawienie ich w inny sposób. W języku C++ realizuje się to za pomocą specjalnej metody zwanej konstruktorem. Jest ona wywoływana automatycznie w momencie tworzenia obiektu. Metoda ta ma nazwę taką samą jak nazwa klasy i nie posiada określonego typu, nie zwraca żadnej wartości, nie jest też typu `void`. Konstruktor może posiadać atrybuty wywołania. Klasa może mieć zdefiniowanych kilka konstruktorów różniących się liczba i typem parametrów wejściowych. W momencie tworzenia obiektu za na podstawie przypisanych atrybutów wykonywana jest odpowiednia metoda konstruktora.

```
#include <iostream>
using namespace std;
class dane{
private:
   int a;
   float b;
public:
    dane(){
        cout << "a="; cin >> a;
        cout << "b="; cin >> b;
    }
    dane(int a, float b){
        this->a = a;
        this->b = b;
    }
    void draw(){
        cout << "a = " << a << " : b = " << b << endl;
```

```
    }
};
int main()
{
    dane a, b(5,34.123);
    a.draw();
    b.draw();
}
```

Zazwyczaj konieczne staje się też wykonanie pewnych czynności "sprzątających" gdy obiekt w programie jest kasowany. Ma to miejsce gdy program lub funkcja kończy swoje działanie i zmienne są kasowane, bądź taka czynność wymuszana jest w kodzie. W języku C++ można zaprogramować metodę która jest automatycznie wykonywana w przypadku kasowania obiektu. Jest ona określana jako destruktor. W projektowanej klasie można zdefiniować tylko jedną taką metodę. Nie może ona mieć także żadnych parametrów wywołania. Jest ona deklarowana analogicznie jak konstruktor, ale poprzedza się ją znakiem ~. Przykład destruktora pokazano w kodzie umieszczonym poniżej.

```
#include <iostream>
using namespace std;
class dane{
private:
   int *tab,n;
public:
    dane(){
        n=1;
        tab = new int[n];
        tab[0] = n;
    }
    dane(int n){
        this->n = n;
        tab = new int[n];
        for(int i=0;i<n;i++)
            tab[i] = n;
    }
    ~dane(){
        delete[] tab;
        cout << "Tablica [" << tab << "] znostała skasowana" << endl;
    }
    void draw(){
        cout << "tab|"<< tab << "|-->";
        for(int i=0;i<n;i++)
            cout << tab[i] << " ";
        cout << endl;
    }
};
int main()
{
    dane a, b(8);
    a.draw();
    b.draw();
}
```

Zadania

W oparciu o informacje wykładowe i treść niniejszej instrukcji wykonać poniższe zadania. Kod programów oprzeć o zasady programowania obiektowego. Kod klasy zapisać w dedykowanych plikach źródłowym i nagłówkowym.

a) Zmodyfikować program z zadania zamieszczonego w instrukcji do ćwiczenia 2 aby korzystał z dynamicznej alokacji pamięci.

b) Zmodyfikować program z zadania zamieszczonego w instrukcji do ćwiczenia 2 aby korzystał z konstruktorów (zaproponować przynajmniej dwa sposoby realizacji) i destruktora. item[c)] W oparciu o treści wykładowe oprzeć tworzenie listy punktów wycieczki o technologię **linked list** lub **double linked list**.