Ćwiczenie 6 z Podstaw programowania.
Język C++, programy pisane w nieobiektowym stylu programowania

Zofia Kruczkiewicz

Zakres

- Strukturalna dekompozycja dużych programów oraz złożonych reprezentacji danych.
- Omówienie i ćwiczenia z reprezentacją problemu "prostej bazy danych" typu:

```
struktura BAZA
        {
         - tablica struktur ELEMENT
                {
                  - pola reprezentowane typami prostymi
                  - pola tekstowe (tablice znaków)
                  - pod-tablice z licznikiem
                  - pola kodowane za pomocą słownika
                }
          - licznik wykorzystywanych elementów tablicy
       }
```

- Kodowanie danych "nienumerycznych" - typ wyliczeniowy.
- Kodowanie danych za pomocą słownika.

Zakres

- Funkcje przetwarzające teksty
  - omówienie standardowych funkcji (biblioteka <string.h>)

  wg -
  http://marek.piasecki.staff.iiar.pwr.wroc.pl/dydaktyka/pp/W07_teksty-tablice_znakowe.pdf
  - tworzenie własnych funkcji
- Dynamiczna alokacja pamięci – tablice jednowymiarowe o zmiennym rozmiarze - realokacja.

  http://marek.piasecki.staff.iiar.pwr.wroc.pl/dydaktyka/pp/W08_dynamiczna_alokacja_pamieci.pdf

Przykład funkcji main dla Zad1

```
//prototypy funkcji,  typy danych, zmienne globalne oraz dostęp do bibliotek
enum {zero, dodaj, zmien, sortuj, usun, pokaz, koniec }; //domyślne wartości
void main()
{
      int opcja;                  // zmienna wyboru typu int
      Osoba Dane[N];
      int ile = 0;
 do
  {
      system("cls");
      cout<<("Jesli nacisniesz klawisz:\n");
      cout<<("1 - Dodawanie kolejnego elementu do tablicy\n");
      cout<<("2 - Wyszukiwanie i zmiana zadanego elementu w tablicy\n");
      cout<<("3 - Sortowanie \n");
      cout<<("5 - Wyswietlenie wszystkich elementow wstawionych do tablicy\n");
      cout<<("6 - Koniec programu\n");
      cin>>opcja;
      switch (opcja)
      {
        case  dodaj: cout << ("Nacisnales klawisz 1 - operacja 1.1\n")<<endl;      Dodaj_osobe(Dane, ile);         break;
        case  zmien: cout << ("Nacisnales klawisz 2 - operacja 1.2\n") << endl;    Zmien_ocene(Dane, ile);         break;
        case  sortuj: cout << ("Nacisnales klawisz 3 - operacja 1.3\n") << endl;   Sortowanie(Dane, ile);           break;
        case  pokaz: cout << ("Nacisnales klawisz 5 - operacja 1.4\n") << endl;    Wyswietl_osoby(Dane, ile);    break;
        case  koniec: cout << ("Nacisnales klawisz 6 - koniec programu\n") << endl;                                break;
        default:        cout << ("Nacisnales niewlasciwy klawisz\n") << endl;
      }
    _getch();    //wstrzymanie programu przed clrscr()przez dodatkowe naciskanie klawisza
  } while (opcja != koniec);
}
//definicje funkcji
```

Zad1

Napisz program, który w opcjach (wg wskazówek z Cw4) wywołuje funkcje, które wykonują następujące czynności na tablicy zdefiniowanej jako **Osoba** Dane[N], gdzie Osoba jest typem strukturalnym:

```
struct Osoba
   {  int Wiek;
      char Nazwisko[DL];
      float Ocena;
      };
```

1) **Dodawanie danych do tablicy**

   Napisz funkcję **void Dodaj_osobe(Osoba tab[], int& ile)**, która wprowadza dane **Wiek** za pomocą funkcji **Podaj_liczbe_calkowita(char *)**, **Nazwisko** za pomocą **Podaj_tablice_znakow(char*, char*)** i **Ocena** za pomocą funkcji **Podaj_liczbe_rzeczywista(char*)**

   - funkcja **int Podaj_liczbe_calkowita(char *komunikat)** sprawdza, czy wprowadzono dane w formacie liczby całkowitej oraz przez wynik zwraca poprawną daną typu **int**
   - funkcja **float Podaj_liczbe_rzeczywista(char* komunikat)** sprawdza, czy wprowadzono dane w formacie liczby rzeczywistej oraz przez wynik zwraca poprawną daną typu **float**
   - funkcja **void Podaj_tablice_znakow(char* komunikat, char* Nazwisko)** sprawdza format łańcucha **Nazwisko** ze względu na niepotrzebne spacje, które usuwa, powtarzające się litery, które zamienia na jedną oraz brak dużej litery jako pierwszy znak łańcucha, którą zamienia na dużą literę. Tablica znaków **Nazwisko** jest przekazywana przez parametr typu wskaźnik na typ char. - wg - **http://marek.piasecki.staff.iiar.pwr.wroc.pl/dydaktyka/pp/W07_teksty-tablice_znakowe.pdf**

   Każda z tych funkcji wyświetla komunikat przekazany przez listę parametrów z globalnej tablicy **Komunikaty.** Należy zastosować cin i cout.

- **Zmienna globalna Komunikaty** - przykład:

```
char *Komunikaty[] =
{ "Wpisz wiek", "Wpisz ocene srednia", "Wpisz nazwisko", "Tablica pelna", " Taka osoba juz istnieje",
" Wstawiono poprawnie", "Tablica pusta", " Usunieto poprawie", " Brak osoby", " Dokonano zmiany oceny",
" Nieprawidlowy format danych – powtorzenie liter", " Nieprawidlowy format danych liczbowych",
" Nieprawidlowy format danych – za duzo spacji", " Wyswietlono poprawnie" };
```

Zad1 cd

```
void Dodaj_osobe( Osoba Dane[], int & ile)
  {   char Nazwisko[DL];
      if (Pelna(ile))
         {   cout << Komunikaty[3] << endl;
             return;  }

      int Wiek = Podaj_liczbe_calkowita(Komunikaty[0]);
      float Ocena = Podaj_liczbe_rzeczywista(Komunikaty[1]);
      Podaj_tablice_znakow(Komunikaty[2], Nazwisko);

      bool wynik = Dodaj_do_tablicy(Wiek, Ocena, Nazwisko, Dane, ile);
       if (!wynik)
              cout <<Komunikaty[4]<< endl ;
       else
             cout <<Komunikaty[5]<< endl ;
  }
```

Funkcja **bool Dodaj_do_tablicy(int Wiek, float Ocena, char Nazwisko[], Osoba Dane[], int& ile)** dodaje po jednym elemencie typu **Osoba** do tablicy na pozycję o indeksie **ile.** Podczas wstawiania pierwszego elementu **ile** powinno być równe 0; po wstawieniu należy powiększyć wartość **ile** o 1. Po ponownym wywołaniu funkcji kolejny element należy wstawiać jako element tablicy o indeksie **ile**, gdy **ile** spełnia warunek: **ile<N.** Po wykonaniu tej opcji, jeśli element został wstawiony do tablicy, należy zawsze powiększyć **ile** o jeden. Zmienną **ile** należy przekazywać przez referencję. Tablica powinna być przekazana przez domyślny wskaźnik , natomiast Wiek i Ocena przez wartość, tablica znaków Nazwisko przez domyślny wskaźnik. Przez wynik funkcji zwracana jest wartość **false** – gdy nie wstawiono, ponieważ obiekt z takim samym nazwiskiem już istnieje oraz **true**, gdy wstawiono. Kolejna dana typu Osoba nie zostanie wstawiona, gdy powtórzyło się takie samo nazwisko. Należy w tej funkcji wywołać funkcję **Osoba* Wyszukaj_wg_nazwiska(Osoba * Dane, int ile, char*Nazwisko)** , która zwraca znalezioną osobę wg tablicy znaków Nazwisko przez wskaźnik lub **NULL**, jeśli nie znaleziono. Gdy wynik jest równy **NULL**, funkcja wstawia nowy element typu Osoba.

Zad1 cd

2. Wyszukiwanie danych osobę w celu zmiany średniej oceny.

Napisz funkcję **void Zmien_ocene(Osoba Dane[], int ile),** która zmienia ocenę u wybranej osoby, wyszukanej wg nazwiska. Po wprowadzeniu danych należy wywołać funkcję **Osoba* Zmien_ocene_osobie(float ocena, char * nazwisko, Osoba* Dane, int ile)**, który wyszuka osobę w tablicy Dane wg składowej Nazwisko za pomocą funkcji **Osoba* Wyszukaj_wg_nazwiska(Osoba * Dane, int ile, char* nazwisko) –** użytej podczas wstawiania danych nowej osoby**,**
i zmieni wartość składowej Ocena i zwróci dane tej osoby przez wynik funkcji typu referencja Osoba*, jeśli ją znaleziono lub **NULL** w przeciwnym wypadku.

```
void  Zmien_ocene(Osoba Dane[], int ile)
   {   char Nazwisko[DL];
       if (Pusta(ile))                          //funkcja zwraca true, gdy ile jest równe 0
         {  cout <<Komunikaty[6]<< endl ;
            return; }
       float Ocena = Podaj_liczbe_rzeczywista(Komunikaty[1]);
       Podaj_tablice_znakow(Komunikaty[2], Nazwisko);
       Osoba* osoba = Zmien_ocene_osobie(Ocena, Nazwisko, Dane, ile);
       if (osoba!=NULL)
          Wyswietl_osobe (*osoba);
                  // np. Nazwisko: Kowalski, Wiek: 20, Ocena srednia: 4.24 i dodaje znak końca linii \n
        else
           cout <<Komunikaty[8]<< endl ;
    }
```

Zad1 cd

3. Napisz funkcję Wyswietl_osoby(Osoba [] Dane, **int** ile), która wyświetla zawartość tablicy Dane za pomocą funkcji Wyswietl_osobe(Osoba osoba) wg komentarza z p.2

```
void  Wyswietl_osoby(Osoba  Dane[], int ile)
    {
      if (Pusta(ile))                          //funkcja zwraca true, gdy ile jest równe 0
        {  cout <<Komunikaty[6]<< endl ;
          return; }
      for (int i=0; i<ile;i++)
        Wyswietl_osobe (Dane [i]);
    }
```

Zad1 cd

4. Napisz funkcję **void** Sortuj(Osoba Dane[], **int** ile), która w przypadku, gdy tablica nie jest pusta, sortuje tablicę wg składowej Nazwisko np. za pomocą algorytmu bąbelkowego:

Algorytm sortowania N elementów:
(1) i ← 1;
(2) dopóki i <= N -1, wykonuj co następuje:
(2.1) j ← 1;
(2.2) dopóki j <= N - i, wykonuj, co następuje:
(2.2.1) jeśli T(j + 1) < T(j), to zamień je;
(2.2.2) j ← j + 1;
(2.3) i ← i +1;

Kod w C++

```
inline void zamien(Osoba &a, Osoba &b)
 { Osoba pom = a;
   a=b;
   b=pom;
 }
inline void porownaj_zamien(Osoba &a, Osoba&b)
    { if (strcmp(b.Nazwisko, a.Nazwisko) == -1)
            zamien(a,b);
    }
void Babelki(Osoba t[], long l, long p)
// sortowanie wyznaczonej części tablicy za pomocą indeksów: od  l do p np. Babelki(t,0,ile-1);
{   for( long i = l;  i<p; i++)
      for (long j = l; j<p-i; j++)
            porownaj_zamien(t[j], t[j+1]);
}
```

Przykład funkcji main dla Zad2 – wersja poszerzona

```
//prototypy funkcji, zmienne globalne , typy danych oraz dostęp do bibliotek
enum { dodaj=1, zmien=2, sortuj=3, usun=4, pokaz=5, koniec=0 };
void main()
{
  int opcja;                        // zmienna wyboru typu int
  Baza baza;
  baza.ile = 0;
  do
   {
       system("cls");
       cout << ("Jesli nacisniesz klawisz:\n");
       cout << ("1 - Dodawanie kolejnego elementu do tablicy\n");
       cout << ("2 - Wyszukiwanie i zmiana zadanego elementu w tablicy\n");
       cout << ("3 - Sortowanie\n");
       cout << ("4 - Usuwanie elementu wstawionych do tablicy\n");
       cout << ("5 - Wyswietlenie wszystkich elementow wstawionych do tablicy\n");
       cout << ("0- Koniec programu\n");
       cin >> opcja;
    switch (opcja)
     {
       case dodaj:   cout << ("Nacisnales klawisz 1 - operacja 1.1\n") << endl;          Dodaj_osobe(baza);          break;
       case zmien:   cout << ("Nacisnales klawisz 2 - operacja 1.2\n") << endl;          Zmien_ocene(baza);          break;
       case sortuj:  cout << ("Nacisnales klawisz 3 - operacja 1.3\n") << endl;          Sortowanie(baza);           break;
       case usun:    cout << ("Nacisnales klawisz 4 - operacja 1.4\n") << endl;          Usun_osobe(baza);           break;
       case pokaz:   cout << ("Nacisnales klawisz 5 - operacja 1.5\n") << endl;          Wyswietl_osoby(baza);       break;
       case koniec: cout << ("Nacisnales klawisz 0 - koniec programu\n") << endl;                                    break;
       default:      cout << ("Nacisnales niewlasciwy klawisz\n") << endl;
     }
  _getch();    //wstrzymanie programu przed clrscr()przez dodatkowe naciskanie klawisza
  } while (opcja != koniec);
}
//definicje funkcji
```

Zad2

Napisz program, który w opcjach (wg wskazówek z Cw4) wywołuje funkcje, które wykonują następujące czynności na strukturze Baza zdefiniowanej jako:

```
struct Baza
   {
     int ile;
      Osoba Dane[N];
  };
```

gdzie:

```
struct Osoba
   {  int Wiek;
     char Nazwisko[DL];
     float Ocena;
     };
```

1) **Dodawanie danych do tablicy** (takie same wskazówki jak w zad1)
   Napisz funkcję **void Dodaj_osobe(Baza & baza)**, która wprowadza dane **Wiek** za pomocą funkcji **Podaj_liczbe_calkowita(char *)**, **Nazwisko** za pomocą **Podaj_tablice_znakow(char*, char*)** i **Ocena** za pomocą funkcji **Podaj_liczbe_rzeczywista(char*)**:
   - funkcja **int Podaj_liczbe_calkowita(char *komunikat)** sprawdza, czy wprowadzono dane w formacie liczby całkowitej oraz przez wynik zwraca poprawną daną typu **int**
   - funkcja **float Podaj_liczbe_rzeczywista(char* komunikat)** sprawdza, czy wprowadzono dane w formacie liczby rzeczywistej oraz przez wynik zwraca poprawną daną typu **float**
   - funkcja **void Podaj_tablice_znakow(char* komunikat, char* Nazwisko)** sprawdza format łańcucha **Nazwisko** ze względu na niepotrzebne spacje, które usuwa, powtarzające się litery, które zamienia na jedną oraz brak dużej litery jako pierwszy znak łańcucha, którą zamienia na dużą literę. Tablica znaków **Nazwisko** jest przekazywana przez parametr typu wskaźnik na typ char.

   Każda z tych funkcji wyświetla komunikat przekazany przez listę parametrów z globalnej tablicy **Komunikaty.** Należy zastosować cin i cout.

- **Zmienna globalna Komunikaty** - przykład:

  char *Komunikaty[]=

  {"Wpisz wiek", "Wpisz ocene srednia", "Wpisz nazwisko","Tablica pelna", " Taka osoba juz istnieje", " Wstawiono poprawnie", "Tablica pusta", " Usunieto poprawie", " Brak osoby"," Dokonano zmiany oceny"," Nieprawidlowy format danych – powtorzenie liter", " Nieprawidlowy format danych liczbowych", " Nieprawidlowy format danych – za duzo spacji"," Wyswietlono poprawnie"};

Zad2 cd

```
void Dodaj_osobe( Baza& baza)
{   char Nazwisko[DL];
    if (Pelna(baza.ile))                //zwraca true, gdy baza.ile==N
      {   cout << Komunikaty[3] << endl;
          return;
       }
    int Wiek = Podaj_liczbe_calkowita(Komunikaty[0]);
    float Ocena = Podaj_liczbe_rzeczywista(Komunikaty[1]);
    Podaj_tablice_znakow(Komunikaty[2], Nazwisko);

    bool wynik = Dodaj_do_tablicy(Wiek, Ocena, Nazwisko,  baza);
     if (!wynik )
            cout <<Komunikaty[4]<< endl ;
       else
          cout <<Komunikaty[5]<< endl ;
}
```

Funkcja **int Dodaj_do_tablicy(int Wiek, float Ocena, char Nazwisko[], Baza& baza)** dodaje po jednym elemencie typu **Osoba** do tablicy **baza.Dane** na pozycję o indeksie **baza.ile.** Podczas wstawiania pierwszego elementu **baza.ile** powinno być równe 0; po wstawieniu należy powiększyć wartość **baza.ile** o 1. Po ponownym wywołaniu funkcji kolejny element należy wstawiać jako element tablicy o indeksie **baza.ile**, gdy **baza.ile** spełnia warunek: **baza.ile<N.** Po wykonaniu tej opcji, jeśli element został wstawiony do tablicy, należy zawsze powiększyć **baza.ile** o jeden. Zmienną **baza.ile** należy przekazywać przez referencję. Struktura baza typu Baza powinna być przekazana przez referencję , natomiast Wiek i Ocena przez wartość, tablica znaków Nazwisko przez domyślny wskaźnik. Przez wynik funkcji zwracana jest wartość **false**– gdy nie wstawiono, ponieważ obiekt z takim samym nazwiskiem już istnieje oraz **true**, gdy wstawiono. Kolejna dana typu Osoba nie zostanie wstawiona, gdy powtórzyło się takie samo nazwisko. W tym celu należy w tej funkcji wywołać funkcję

Osoba* Wyszukaj_wg_nazwiska(Baza baza, char*Nazwisko, int& ktory) , która zwraca znalezioną osobę wg tablicy znaków Nazwisko przez wskaźnik lub **NULL**, jeśli takiej danej nie znaleziono. Parametr **który** zawiera indeks znalezionego elementu w tablicy **baza.Dane**. Gdy wynik jest równy **NULL**, należy wstawić nową osobę.

Zad2 cd

2. Napisz funkcję **Wyswietl_osoby(Baza baza),** która wyświetla zawartość tablicy Dane za pomocą funkcji **Wyswietl_osobe(Osoba osoba)** wg komentarza z p.2. Należy zastosować cout

```
void  Wyswietl_osoby(Baza baza)
    {   if (Pusta(baza.ile))                          //funkcja zwraca true, gdy baza.ile jest równe 0
          {   cout <<Komunikaty[6]<< endl ;
              return; }
        for (int i=0; i<ile;i++)
          Wyswietl_osobe (baza.Dane[i]);
    }
```

3. Napisz funkcję **int Usun_osobe(Baza & baza),** która pobiera poprawną daną typu tablica znaków za pomocą funkcji Podaj_tablice_znakow(char*, char*) i wywołuje funkcję **int Usun_z_tablicy(Baza &baza, char * Nazwisko)**, która za pomocą funkcji **Osoba* Wyszukaj_wg_nazwiska(Baza baza, char*Nazwisko, int& ktory)** znajduje element tablicy Dane jako składowej struktury baza. Jeśli zwraca **NULL**, oznacza to brak danych osoby o podanym nazwisku lub dane osoby w przeciwnym wypadku. W przypadku znalezienia elementu usuwa element typu Osoba w tablicy **baza.Dane** o indeksie **ktory**, zsuwając elementy w tablicy **baza.Dane** rozpoczynając od indeksu **ktory**. Funkcja **Usun_z_tablicy** zwraca 0, gdy nie znaleziono elementu, a różną od 0, gdy element typu Osoba został usunięty z tablicy.

```
void Usun_osobe( Baza& baza)
    {  char Nazwisko[DL];
       if (Pusta(baza.ile))                           //funkcja zwraca true, gdy baza.ile jest równe 0
         {  cout <<Komunikaty[6]<< endl ;
            return; }
       Podaj_tablice_znakow(Komunikaty[2], Nazwisko);
       int wynik = Usun_z_tablicy( baza, Nazwisko);
         if (wynik ==0)
               cout <<Komunikaty[8]<< endl ;
          else
              cout <<Komunikaty[7]<< endl ;
}
```

Pytania – wg materiałów
http://marek.piasecki.staff.iiar.pwr.wroc.pl/dydaktyka/pp/laboratorium.html, Ćwiczenie 5

- „Wyjaśnij różnice pomiędzy realizacją dynamicznej alokacji pamięci za pomocą funkcji **calloc**( ), **malloc**( ), **free**() a wykorzystaniem do tego celu operatorów **new** i **delete**
- Wyjaśnij jak działa funkcja **realloc**( ). Zastanów się dlaczego w języku **C++** nie wprowadzono operatora, który byłby odpowiednikiem funkcji **realloc**( )”

Przykład funkcji main dla Zad3 – wersja większa niż wymagana w zadaniu

```
//prototypy funkcji, typy danych, zmienne globalne oraz dostęp do bibliotek
enum kl { dodaj='a', zmien='b', sortuj='c', usun='d', pokaz='e', koniec='k' };
void main()
{
	char opcja;                                                    // zmienna wyboru typu char
	int ile , rozmiar;
	Osoba* Dane= Wykonaj_tablice(ile, rozmiar);
do
{
	system("cls");
	cout << ("Jesli nacisniesz klawisz:\n");
	cout << ("a - Dodawanie kolejnego elementu do tablicy\n");
	cout << ("b - Wyszukiwanie i zmiana zadanego elementu w tablicy\n");
	cout << ("c - Sortowanie \n");
	cout << ("d - Usuwanie \n");
	cout << ("e - Wyswietlenie wszystkich elementow wstawionych do tablicy\n");
	cout << ("k - Koniec programu\n");
	cin >> opcja;
	switch (opcja)
	{
	case dodaj:    cout << "Nacisnales klawisz "<<opcja<< " - operacja 1.1\n" << endl;    Dodaj_osobe(Dane, ile,rozmiar);        break;
	case zmien:    cout << "Nacisnales klawisz "<<opcja<< "- operacja 1.2\n" << endl;     Zmien_ocene(Dane, ile);                break;
	case sortuj:   cout << "Nacisnales klawisz "<<opcja<<" - operacja 1.3\n"<< endl;      Sortowanie(Dane, ile);                 break;
	case usun:     cout << "Nacisnales klawisz "<<opcja<<" - operacja 1.4\n" << endl;     Usun_osobe(Dane, ile);                 break;
	case pokaz:    cout << "Nacisnales klawisz "<<opcja<<" - operacja 1.5\n" << endl;     Wyswietl_osoby(Dane, ile);             break;
	case koniec : cout << "Nacisnales klawisz "<<opcja<<" - koniec programu\n" << endl;                                         break;
	default:       cout << ("Nacisnales niewlasciwy klawisz\n") << endl;
	}
	_getch();                                  //wstrzymanie programu przed przez dodatkowe naciskanie klawisza
  } while (opcja != koniec);
}
//definicje funkcji
```

Zad3

Napisz program, który w opcjach (wg wskazówek z Cw4) wywołuje funkcje, które wykonują następujące czynności na dynamicznej tablicy struktur typu Osoba : **Osoba * Dane** o zadanym rozmiarze **rozmiar**. Zmienna **rozmiar** oznacza rozmiar tablicy, a **ile** oznacza liczbę elementów wstawionych do tablicy (**W08_dynamiczna_alokacja_pamieci)**

```
struct Osoba
    {   int Wiek;
        char Nazwisko[DL];
        float Ocena;
        };
```

1) **Należy wykonać dynamiczną tablice Dane** o rozmiarze **rozmiar** nadanym umownie w funkcji oraz wartości ile równiej 0:
   Osoba * Wykonaj_tablice(int& ile, int& rozmiar)
1) **Dodawanie danych do tablicy** (takie same wskazówki jak w zad1)
   Napisz funkcję **void Dodaj_osobe(Osoba *& Dane, int & ile, int& rozmiar)**, która wprowadza dane **Wiek** za pomocą funkcji **Podaj_liczbe_calkowita(char *)**, **Nazwisko** za pomocą **Podaj_tablice_znakow(char*, char*)** i **Ocena** za pomocą funkcji **Podaj_liczbe_rzeczywista(char*)**:
   - funkcja **int Podaj_liczbe_calkowita(char *komunikat)** sprawdza, czy wprowadzono dane w formacie liczby całkowitej oraz przez wynik zwraca poprawną daną typu **int**
   - funkcja **float Podaj_liczbe_rzeczywista(char* komunikat)** sprawdza, czy wprowadzono dane w formacie liczby rzeczywistej oraz przez wynik zwraca poprawną daną typu **float**
   - funkcja **void Podaj_tablice_znakow(char* komunikat, char* Nazwisko)** sprawdza format łańcucha **Nazwisko** ze względu na niepotrzebne spacje, które usuwa, powtarzające się litery, które zamienia na jedną oraz brak dużej litery jako pierwszy znak łańcucha, którą zamienia na dużą literę. Tablica znaków **Nazwisko** jest przekazywana przez parametr typu wskaźnik na typ char.

Każda z tych funkcji wyświetla komunikat przekazany przez listę parametrów z globalnej tablicy **Komunikaty.** Należy zastosować cin i cout.

- **Zmienna globalna Komunikaty** - przykład:

```
char *Komunikaty[]=
{"Wpisz wiek", "Wpisz ocene srednia", "Wpisz nazwisko","Tablica pelna", " Taka osoba juz istnieje", " Wstawiono poprawnie", "Tablica pusta", " Usunieto poprawie", " Brak osoby", "Nie powiekszono tablicy",  " Dokonano zmiany oceny"," Nieprawidlowy format danych – powtorzenie liter", " Nieprawidlowy format danych liczbowych", " Nieprawidlowy format danych – za duzo spacji"," Wyswietlono poprawnie"};
```

Zad3 cd

```
void Dodaj_osobe(Osoba *& Dane, int & ile, int& rozmiar)
{  char Nazwisko[DL];
   if  (Pelna(ile, rozmiar))                    //jeśli ile==rozmiar funkcja zwraca true, w przeciwnym przypadku false
    if( Powieksz_tablice(Dane, ile, rozmiar)==0)   //funkcja powiększa tablicę dynamiczną
      {
          cout <<Komunikaty[9]<< endl ;
          return;
       }
    int Wiek = Podaj_liczbe_calkowita(Komunikaty[0]);
    float Ocena = Podaj_liczbe_rzeczywista(Komunikaty[1]);
    Podaj_tablice_znakow(Komunikaty[2], Nazwisko);

    bool wynik = Dodaj_do_tablicy(Wiek, Ocena, Nazwisko, Dane, ile);
     if (!wynik)
            cout <<Komunikaty[4]<< endl ;
        else
          cout <<Komunikaty[5]<< endl ;
}
```

Funkcja **int Powieksz_tablice(Osoba*&Dane, int ile, int& rozmiar)** tworzy nową, większą tablicę o nowym rozmiarze **rozmiar** (za pomocą operatora new), kopiuje do niej **ile** elementów z tablicy **Dane**, usuwa tablicę **Dane** z pamięci (za pomocą operatora delete) przypisuje do niej nową tablicę. Jeśli uda się utworzyć nową tablicę, funkcja zwraca wartość różną od 0, w przeciwnym wypadku 0. W tym przypadku funkcja Dodaj_osobe kończy swoje działanie.

Funkcja **int Dodaj_do_tablicy(int Wiek, float Ocena, char Nazwisko[], Osoba* Dane, int& ile)** dodaje po jednym elemencie typu **Osoba** do tablicy **Dane** na pozycję o indeksie **ile.** Podczas wstawiania pierwszego elementu **ile** powinno być równe 0; po wstawieniu należy powiększyć wartość **ile** o 1. Po ponownym wywołaniu funkcji kolejny element należy wstawiać jako element tablicy o indeksie **ile**, gdy **ile** spełnia warunek: **ile<N.** Po wykonaniu tej opcji, jeśli element został wstawiony do tablicy, należy zawsze powiększyć **ile** o jeden. Zmienną **ile** należy przekazywać przez referencję. Tablica Dane powinna być przekazana przez wskaźnik Osoba*, ile przez referencję , natomiast Wiek i Ocena przez wartość, tablica znaków Nazwisko przez domyślny wskaźnik. Przez wynik funkcji zwracana jest wartość false– gdy nie wstawiono, ponieważ obiekt z takim samym nazwiskiem już istnieje oraz true, gdy wstawiono. Kolejna dana typu Osoba nie zostanie wstawiona, gdy powtórzyło się takie samo nazwisko. W tym celu należy w tej funkcji wywołać funkcję **Osoba* Wyszukaj_wg_nazwiska(Osoba* Dane, int ile, char* Nazwisko, int& ktory)** , która zwraca znalezioną osobę wg tablicy znaków Nazwisko przez wskaźnik lub **NULL**, jeśli takiej danej nie znaleziono. Parametr **który** zawiera indeks znalezionego elementu w tablicy **Dane**. Gdy wynik jest równy **NULL**, należy wstawić nową osobę.

Zad3 cd

2. Napisz funkcję **Wyswietl_osoby(Osoba* Dane, int ile),** która wyświetla zawartość tablicy Dane za pomocą funkcji **Wyswietl_osobe(Osoba osoba)** wg komentarza z p.2. Należy zastosować cout

```
void  Wyswietl_osoby(Osoba* Dane, int ile)
    {    if (Pusta(ile))                                  //funkcja zwraca true, gdy ile jest równe 0
           {   cout <<Komunikaty[6]<< endl ;
               return; }
         for (int i=0; i<ile;i++)
           Wyswietl_osobe (Dane[i]);
    }
```

3. Napisz funkcję **int Usun_osobe(Osoba* Dane, int& ile),** która pobiera poprawną daną typu tablica znaków za pomocą funkcji Podaj_tablice_znakow(char*, char*) i wywołuje funkcję

   int Usun_z_tablicy(Osoba* Dane, int& ile, char * Nazwisko), która za pomocą funkcji

   Osoba* Wyszukaj_wg_nazwiska(Osoba* Dane, int ile, char*Nazwisko, int& ktory) znajduje element tablicy Dane. Jeśli zwraca **NULL**, oznacza to brak danych osoby o podanym nazwisku lub dane osoby w przeciwnym wypadku. W przypadku znalezienia elementu usuwa element typu Osoba w tablicy **Dane** o indeksie **który**, zsuwając elementy w tablicy Dane rozpoczynając od indeksu który. Funkcja **Usun_z_tablicy** zwraca 0, gdy nie znaleziono elementu, a różną od 0, gdy element typu Osoba został usunięty z tablicy.

```
void Usun_osobe( Osoba* Dane, int& ile)
    {   char Nazwisko[DL];
        if (Pusta(ile))                                   //funkcja zwraca true, gdy ile jest równe 0
          {  cout <<Komunikaty[6]<< endl ;
             return; }
        Podaj_tablice_znakow(Komunikaty[2], Nazwisko);
        int wynik = Usun_z_tablicy(Dane,  ile, Nazwisko);
          if (wynik ==0)
                cout <<Komunikaty[8]<< endl ;
           else
                cout <<Komunikaty[7]<< endl ; }
```