Ćwiczenie 7 z Podstaw programowania.
Język C++, programy pisane w nieobiektowym stylu programowania

Zofia Kruczkiewicz

Zakres

- Funkcje przetwarzające teksty
  (biblioteka <string.h>)
  - tworzenie własnych funkcji
- Tablice jednowymiarowe i dwuwymiarowe
- Tablice wskaźników na łańcuchy
- Tablice wskaźników na struktury

Tablica dwuwymiarowa znaków

Przykład funkcji main dla Zad1

```
enum kl{ dodaj = 'a', zmien = 'b', sortuj = 'c', usun = 'd', pokaz = 'e', koniec = 'k' };
int main()
{       char opcja; // zmienna wyboru typu int
        int ile=0;
        char Dane[N][DL];
        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); break;
                        case zmien: cout << "Nacisnales klawisz " << opcja << "- operacja 1.2\n" << endl;
                                        Zmien_nazwisko(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();
        } while (opcja != koniec);
        return 0;
}
```

Zad1 cd

Napisz program, który w opcjach (wg wskazówek z Cw4) wywołuje funkcje, które wykonują następujące czynności na tablicy **Dane** zdefiniowanej jako **char** Dane[N][DL], opisane dalej. **W komentarzach należy przedstawić odwołania do tablicy za pomocą indeksowania**

1) **Dodawanie danych do tablicy**
   Napisz funkcję **void Dodaj_osobe(char (*Dane)[DL], int & ile)**, która wprowadza dane **Nazwisko** za pomocą **Podaj_tablice_znakow(char*, char*)**
   - 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 już 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(char (*Dane)[DL], int & ile)      // void Dodaj_osobe(char   Dane[][DL], int & ile)
{     char Nazwisko[DL];
      if (Pelna(ile))
        {
              cout << *(Komunikaty+3) << endl;
              return;
        }
      cout << "rozmiar: " << ile << endl;
      Podaj_tablice_znakow(*(Komunikaty+1), Nazwisko);
      bool wynik = Dodaj_do_tablicy(Nazwisko, Dane, ile);
      if (!wynik)
              cout << *(Komunikaty+4) << endl;
      else
              cout << *(Komunikaty+5) << endl;
}
```

Funkcja **bool Dodaj_do_tablicy(char* Nazwisko, char (*Dane)[DL], int& ile)** dodaje po jednym elemencie typu **char[DL]** 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 , 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 char[DL] nie zostanie wstawiona, gdy powtórzyło się takie samo nazwisko. Należy w tej funkcji wywołać funkcję **char* Wyszukaj_wg_nazwiska(char (*Dane)[DL], int ile, char*Nazwisko, int &ktory),** 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 char[DL].

Zad1 cd

```
bool Dodaj_do_tablicy(char* Nazwisko, char (*Dane)[DL], int& ile)          //char Dane[][DL]
{
    int ktory;
    char *pom;

    pom = Wyszukaj_wg_nazwiska(Dane, ile, Nazwisko, ktory);
    if (pom != NULL)
       return false;
    strcpy(*(Dane+ile++), Nazwisko);
    return true;
}

char* Wyszukaj_wg_nazwiska( char (*Dane)[DL], int ile, char*Nazwisko, int &ktory)   //char Dane[][DL]
{
    for (int i = 0; i < ile; i++)
          if (strcmp(*(Dane+i), Nazwisko) == 0)
           {
                    ktory = i;
                    return *(Dane+i);
           }
    return NULL;
}
```

Zad1 cd

2. ** (funkcja dodatkowa) **Wyszukiwanie danych osoby w celu zmiany nazwiska.**

Napisz funkcję **void Zmien_nazwisko(char (*Dane)[DL], int ile),** która zmienia ocenę u wybranej osoby, wyszukanej wg nazwiska. Po wprowadzeniu danych należy wywołać funkcję
char* Zmien_nazwisko_osobie(char* nowe, char * nazwisko, char (*Dane)[DL], int ile),
który wyszuka osobę w tablicy Dane wg łańcucha nazwisko za pomocą funkcji
char* Wyszukaj_wg_nazwiska(char (*Dane)[DL], int ile, char*Nazwisko, int &ktory), – użytej podczas wstawiania nazwiska nowej osoby**,**
i zmieni wartość łańcucha znaków wyszukanym elemencie tablicy i zwróci dane tej osoby przez wynik funkcji typu referencja char*, jeśli ją znaleziono lub **NULL** w przeciwnym wypadku.

```
void  Zmien_nazwisko(char (*Dane)[DL], int ile)
  {
    char Nazwisko[DL], Nowe[DL];
    if (Pusta(ile))                    //funkcja zwraca true, gdy ile jest równe 0
    {
            cout << *(Komunikaty+6) << endl;
            return;
    }
    Podaj_tablice_znakow(*(Komunikaty+1), Nazwisko);
    Podaj_tablice_znakow(*(Komunikaty+2), Nowe);
    char* osoba = Zmien_nazwisko_osobie(Nowe, Nazwisko, Dane, ile);
    if (osoba != NULL)
            Wyswietl_osobe(osoba);

    else
            cout << *(Komunikaty+8) << endl;
}
```

Zad1 cd

3. Napisz funkcję **void Usun_osobe(char (*Dane)[DL], int& ile),** która pobiera poprawną daną typu tablica znaków za pomocą funkcji Podaj_tablice_znakow(char*, char*) i wywołuje funkcję **int Usun_z_tablicy(char (*Dane)[DL], int& ile, char * Nazwisko)**, która za pomocą funkcji **char* Wyszukaj_wg_nazwiska(char (*Dane)[DL], 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 char[DL] w tablicy **Dane** o indeksie **który**, a następnie 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 char[DL] został usunięty z tablicy.

```
void Usun_osobe(char (*Dane)[DL], 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;
}
```

Zad1 cd

4. Napisz funkcję **void Wyswietl_osoby_(char (*Dane)[DL], int ile)**, która wyświetla zawartość tablicy **Dane** za pomocą kolejnych znaków w tablicy dwuwymiarowej, odwołując się do nich za pomocą wskaźników. Należy to zmienić na odwołania indeksowane.

```
void  Wyswietl_osoby_(char  (*Dane)[DL], 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++)
     { for (int j=0; *(*(Dane+i)+j)!='\0';j++) //
            cout<<*(*(Dane+i)+j);  //Dane[i][j]
      cout<<endl;                  //  *(Dane+i) – równoważne char*, czyli Dane[i]
    }                              //  (Dane+i) – wskaźnik na tablicę znaków  char (*Dane)[DL] umieszczoną w wierszu i.
 }                                 //Dodanie i oznacza niejawne zwiększenie wartości wskaźnika Dane: *Dane+i*sizeof(char)*DL
                                   //  dodanie do *(Dane+i) +j oznacza niejawne wyznaczenie adresu znaku
                                   // o indeksie j w tablicy znaków Dane[i], czyli : *Dane+(i*DL + j)*sizeof(char)
```

// równoważny, jawny zapis: *(*Dane+ i*DL +j) – gdzie *Dane jest wskaźnikiem na znak (char*), wskazujący na pierwszy znak o //indeksie 0 w wierszu o indeksie 0

Zad1 cd

5. Napisz funkcję, **void Sortowanie(char (*t)[DL], int ile)**, która sortuje tablicę np. za pomocą algorytmu bąbelkowego:

```
void Sortowanie(char (*t)[DL], int ile)
{
      if (Pusta(ile))                        //funkcja zwraca true, gdy ile jest równe 0
      {
            cout << *(Komunikaty+6) << endl;
            return;
      }
      Babelki(t, 0, ile - 1);
}
```

Kod w C++ (ogólny)

```
inline void zamien(element &a, element &b)
 { element pom = a;
   a=b;
   b=pom;
 }
inline void porownaj_zamien(element &a, element &b)
    { if (b<a) zamien(a,b); }

void babelki(element 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]);
}
```

Tablica dynamicznych łańcuchów

Przykład funkcji main dla Zad2

```
enum kl{ dodaj = 'a', zmien = 'b', sortuj = 'c', usun = 'd', pokaz = 'e', koniec = 'k' };
int main()
{       char opcja; // zmienna wyboru typu int
        int ile=0;
        char* Dane[N];
        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); break;
                        case zmien: cout << "Nacisnales klawisz " << opcja << "- operacja 1.2\n" << endl;
                                        Zmien_nazwisko(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();
        } while (opcja != koniec);
        return 0;
}
```

Zad2 cd

Napisz program, który w opcjach (wg wskazówek z Cw4) wywołuje funkcje, które wykonują następujące czynności na tablicy **Dane** zdefiniowanej jako **char*** Dane[N] (tablica dynamicznych tablic znaków), opisane dalej. W komentarzach należy przedstawić odwołania do tablicy za pomocą indeksowania

1) **Dodawanie danych do tablicy**

   Napisz funkcję **void Dodaj_osobe(char * Dane[], int & ile)**, która wprowadza dane **Nazwisko** za pomocą **Podaj_tablice_znakow(char*, char*)** :

   - 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 już 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(char * Dane[], int & ile)
{
      if (Pelna(ile))
        {       cout << *(Komunikaty+3) << endl;
                return;
         }
      char* Nazwisko  = new char [DL];   //przydział pamięci dla kolejnego elementu tablicy
      cout << "rozmiar: " << ile << endl;
      Podaj_tablice_znakow(*(Komunikaty+1), Nazwisko);

      bool wynik = Dodaj_do_tablicy(Nazwisko, Dane, ile);
      if (!wynik)
                cout << *(Komunikaty+4) << endl;
      else
                cout << *(Komunikaty+5) << endl;
}
```

Funkcja **bool Dodaj_do_tablicy(char* Nazwisko, char* Dane[], int& ile)** dodaje po jednym elemencie typu **char[DL]** do tablicy na pozycję o indeksie**.** 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 , 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 char[DL] nie zostanie wstawiona, gdy powtórzyło się takie samo nazwisko. Należy w tej funkcji wywołać funkcję
char* Wyszukaj_wg_nazwiska(char *Dane[], int ile, char*Nazwisko, int &ktory), 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 char[DL].

Zad2 cd

```
char* Wyszukaj_wg_nazwiska( char *Dane[], int ile, char*Nazwisko, int &ktory)
{
    for (int i = 0; i < ile; i++)
            if (strcmp(*(Dane+i), Nazwisko) == 0)
            {
                    ktory = i;
                    return *(Dane+i);
            }
    return NULL;
}

bool Dodaj_do_tablicy(char* Nazwisko, char* Dane[], int& ile)
{
    int ktory;
    char *pom;
    pom = Wyszukaj_wg_nazwiska(Dane, ile, Nazwisko, ktory);
    if (pom != NULL)
       {   delete  Nazwisko;  //usunięcie z pamięci łańcucha, jeśli jest niepotrzebny
           return false;
        }
    *(Dane+ile++)= Nazwisko;
    return true;
}
```

Zad2 cd

2. ** (funkcja dodatkowa) **Wyszukiwanie danych osoby w celu zmiany nazwiska.**

Napisz funkcję **void Zmien_nazwisko(char** Dane, int ile),** która zmienia ocenę u wybranej osoby, wyszukanej wg nazwiska. Po wprowadzeniu danych należy wywołać funkcję
char* Zmien_nazwisko_osobie(char* nowe, char * nazwisko, charDane, int ile);,**
który wyszuka osobę w tablicy Dane wg łańcucha Nazwisko za pomocą funkcji
char* Wyszukaj_wg_nazwiska(char *Dane[], int ile, char*Nazwisko, int &ktory), – użytej podczas wstawiania nazwiska nowej osoby**,**
i zmieni wartość łańcucha znaków w yszukanym elemencie tablicy i zwróci dane tej osoby przez wynik funkcji typu referencja char*, jeśli ją znaleziono lub **NULL** w przeciwnym wypadku.

```
void  Zmien_nazwisko(char** Dane, int ile)
{
    char Nazwisko[DL], Nowe[DL];
    if (Pusta(ile))                    //funkcja zwraca true, gdy ile jest równe 0
    {
        cout << *(Komunikaty+6) << endl;
        return;
    }
    Podaj_tablice_znakow(*(Komunikaty+1), Nazwisko);
    Podaj_tablice_znakow(*(Komunikaty+2), Nowe);
    char* osoba = Zmien_nazwisko_osobie(Nowe, Nazwisko, Dane, ile);
    if (osoba != NULL)
        Wyswietl_osobe(osoba);
    else
        cout << *(Komunikaty+8) << endl;
}
```

Zad2 cd

3. Napisz funkcję **void Usun_osobe(char** 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(char **Dane, int& ile, char * Nazwisko), która za pomocą funkcji

char* Wyszukaj_wg_nazwiska(char *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 char[DL] w tablicy **Dane** o indeksie **który**, usuwając z pamięci element **delete *(Dane+ktory)**, a następnie zsuwając elementy w tablicy **Dane** rozpoczynając od indeksu **ktory**. Funkcja **Usun_z_tablicy** zwraca 0, gdy nie znaleziono elementu, a różną od 0, gdy element typu char[DL] został usunięty z tablicy.

```
void Usun_osobe(char** 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;
}
```

Zad2 cd

4. Napisz funkcję **void Wyswietl_osoby_(char** Dane, int ile)**, która wyświetla zawartość tablicy **Dane** za pomocą kolejnych znaków w tablicy dwuwymiarowej, odwołując się do nich za pomocą wskaźników. Należy to zmienić na odwołania indeksowane.

```
void  Wyswietl_osoby_(char** 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++)
     { for (int j=0; *(*(Dane+i)+j)!='\0';j++)
            cout<<*(*(Dane+i)+j);     // Dane[i][j]   lub *(*Dane + i*DL + j)
      cout<<endl;
   }
}
```

Zad2 cd

5. Napisz funkcję **void Sortowanie(char **t, int ile)**, która sortuje tablicę np. za pomocą algorytmu bąbelkowego:

```
void Sortowanie(char **t, int ile)
{
      if (Pusta(ile))                        //funkcja zwraca true, gdy ile jest równe 0
      {
            cout << *(Komunikaty+6) << endl;
            return;
      }
      Babelki(t, 0, ile - 1);
}
```

Kod w C++ (ogólny)

```
inline void zamien(element &a, element &b)
 { element pom = a;
   a=b;
   b=pom;
 }
inline void porownaj_zamien(element &a, element &b)
    { if (b<a) zamien(a,b); }

void babelki(element 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]);
}
```

Tablica dynamicznych struktur

Przykład funkcji main dla Zad3

```
enum{ dodaj='a', zmien='b', sortuj='c', usun='d', pokaz='e', koniec='k' };
enum { wiek, ocena, nazwisko, pelna, jest, dod_poprawnie, pusta, us_poprawnie, brak, zmiana, litery, liczby, spacje, ok};
int main()
{       char opcja; // zmienna wyboru typu int
        Osoba* Dane[N];
        int ile = 0;
        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 a - operacja 1.1\n")<<endl;
                                        Dodaj_osobe(Dane, ile); break; //wstawic wywolanie funkcji wstawiajacej kolejny element do tablicy wg 1.1.
                        case zmien:cout << ("Nacisnales klawisz b - operacja 1.2\n") << endl;
                                        Zmien_ocene(Dane, ile); break;// wstawic wywolanie funkcji wyszukującej element w tablicy wh p.1.2
                        case sortuj:cout << ("Nacisnales klawisz c - operacja 1.3\n") << endl;
                                       Sortowanie(Dane, ile); break;// wstawic wywolanie funkcji wg 1.3 wyswietlającej element tablicy o zadanym indeksie
                        case usun:  cout << "Nacisnales klawisz d" << opcja << " - operacja 1.4\n" << endl;
                                        Usun_osobe(Dane, ile); break;
                        case pokaz:cout << ("Nacisnales klawisz e - operacja 1.4\n") << endl;
                                        Wyswietl_osoby(Dane, ile); break;// wstawic wywolanie funkcji wyswietlajacej zawartosc tablicy wg p.1.4
                        case koniec:cout << ("Nacisnales klawisz k - koniec programu\n") << endl; break;
                        default:cout << ("Nacisnales niewlasciwy klawisz\n") << endl;
                        }
                        _getch();    //wstrzymanie programu przed system("cls") przez dodatkowe naciskanie klawisza
        } while (opcja != koniec);
        fflush(stdin);
        return 0;
}
```

Zad3 cd

Napisz program, który w opcjach (wg wskazówek z Cw4) wywołuje funkcje, które wykonują następujące czynności na tablicy **Dane** 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.

   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 już 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"};
```

Zad3 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+wiek));
   float Ocena = Podaj_liczbe_rzeczywista(*(Komunikaty+ocena));
   Podaj_tablice_znakow(*(Komunikaty+nazwisko), Nazwisko);

   bool wynik = Dodaj_do_tablicy(Wiek, Ocena, Nazwisko, Dane, ile);
   if (!wynik)
            cout << *(Komunikaty +jest) << endl;
   else
            cout << *(Komunikaty+dod_poprawnie) << endl;
}
```

Funkcja **int 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.** Należy przydzielić pamięć na element tablicy typu **Osoba.** 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, int& ktory),** 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.

Zad3 cd

```
bool Dodaj_do_tablicy(int Wiek, float Ocena, char Nazwisko[], Osoba** Dane,  int& ile)
{
    Osoba* osoba, *pom;
    int ktory;
     pom = Wyszukaj_wg_nazwiska(Dane, ile, Nazwisko, ktory);
    if (pom != NULL)
            return false;
    osoba= new Osoba();                    //wywołanie domyślnego konstruktora struktury Osoba()
    strcpy(osoba->Nazwisko, Nazwisko);     //(*osoba).Nazwisko
    osoba->Ocena = Ocena;                  //(*osoba).Ocena
    osoba->Wiek = Wiek;                    //(*osoba).Wiek
    *(Dane + ile++) = osoba;
    return true;
}

Osoba* Wyszukaj_wg_nazwiska(Osoba ** Dane, int ile, char*Nazwisko, int& ktory)
{
    for (int i = 0; i < ile; i++)
            if (strcmp((*(Dane+i))->Nazwisko, Nazwisko) == 0) // if (strcmp((*(*(Dane+i))).Nazwisko, Nazwisko) == 0)
            {
                    ktory=i;
                   return *(Dane+i);
               }
    return NULL;
}
```

Zad3 cd

2. ** (funkcja dodatkowa) **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, int& ktory) –** 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+pusta) << endl;
            return;
    }
    Podaj_tablice_znakow(*(Komunikaty+nazwisko), Nazwisko);
    float Ocena = Podaj_liczbe_rzeczywista(*(Komunikaty+ocena));
    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+brak) << endl;
}
```

Zad3 cd

3. Napisz funkcję **void 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**, usuwając element za pomocą operatora delete: **delete *(Dane+który),** a następnie 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+pusta) << endl;
            return;
    }
    Podaj_tablice_znakow(*(Komunikaty+2), Nazwisko);

    int wynik = Usun_z_tablicy(Dane, ile, Nazwisko);
    if (wynik == 0)
            cout << *(Komunikaty+brak) << endl;
    else
            cout << *(Komunikaty+us_poprawnie) << endl;
 }
```

Zad3 cd

4. 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+pusta) << endl;
            return;
    }
    for (int i = 0; i<ile; i++)
            Wyswietl_osobe(*(Dane+i));

}
```

Zad3 cd

5. Napisz funkcję **void Sortuj(Osoba ** Dane, int ile),** która sortuje tablicę wg składowej Nazwisko np. za pomocą algorytmu bąbelkowego:

```
void Sortowanie(Osoba** t, int ile)
{
      if (Pusta(ile))                        //funkcja zwraca true, gdy ile jest równe 0
      {
            cout << Komunikaty[pusta] << endl;
            return;
      }
      Babelki(t, 0, ile - 1);
}
```

Kod w C++ (ogólny)

```
inline void zamien(element &a, element &b)
 { element pom = a;
   a=b;
   b=pom;
 }
inline void porownaj_zamien(element &a, element &b)
    { if (b<a) zamien(a,b); }

void babelki(element 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]);
}
```