

[p\_00.cpp]

```
// Przykładowy program ilustrujący operacje na zwykłej tablicy liczb "float".  
// Operacja zapisania wczytanej "wartosci" do tablicy  
// oraz operacja wyswietlenia wskazanego elementu.  
// Bez kontrolowania poprawnosci indeksu !
```

```
#include <iostream.h>  
#include <conio.h>  
  
void main()  
{  
    float TAB[50];  
    int   pozycja;  
    float x;  
  
    cout << "Podaj zapisywana pozycje: ";  
    cin  >> pozycja;  
    cout << "Podaj wartosc: ";  
    cin  >> x;  
  
    TAB[pozycja] = x;  
  
    cout << "Podaj odczytywana pozycje: ";  
    cin  >> pozycja;  
  
    x = TAB[pozycja];  
  
    cout << "Odczytana wartosc: " << x;  
    getch();  
}
```

## [p\_01.cpp]

```
// Program realizujący te same operacje co przykład [ p_00.cpp]
// ale z zastosowaniem abstrakcyjnego interfejsu do tablicy
// za pomocą funkcji

// NIEFORMALNY OPIS FUNKCJI OPERUJĄCYCH NA TABLICY
//
// WstawElement - funkcja umieszczająca zadana <wartosc>
//                 w liście na zadanej <pozycji>
//                 Podczas wykonywania tej operacji
//                 powinna być kontrolowana poprawność (zakres)
//                 wskazanej pozycji i sygnalizowana poprzez kod_bledu.
//
// PodajElement - funkcja zwracająca <wartosc> odczytana
//                 z zadanej <pozycji> w liście.
//                 Podczas wykonywania tej operacji
//                 powinna być kontrolowana poprawność (zakres)
//                 wskazanej pozycji

// FORMALNY INTERFEJS FUNKCJI
void WstawElement(float tablica[], int rozmiar,
                  int pozycja, float wartosc,
                  int& kod_bledu);

float PodajElement(float tablica[], int rozmiar,
                  int pozycja,
                  int& kod_bledu);

#include <iostream.h>
#include <conio.h>

void main()
{
    float TAB[50];
    int kod_bledu;
    int pozycja;
    float x;

    cout << "Podaj zapisywaną pozycję: ";
    cin >> pozycja;
    cout << "Podaj wartość: ";
    cin >> x;

    WstawElement( TAB, 50, pozycja, x, kod_bledu);

    if( kod_bledu!=0 )
        cout<<"\nWystąpił błąd zakresu podczas zapisu do tablicy\n";

    cout << "Podaj odczytywaną pozycję: ";
    cin >> pozycja;

    x = PodajElement( TAB, 50, pozycja, kod_bledu);

    if( kod_bledu!=0 )
        cout<<"\nWystąpił błąd zakresu podczas odczytu z tablicy\n";

    cout << "Odczytana wartość: " << x;
    getch();
}
```

*//PRZYKLADOWA IMPLEMENTACJA FUNKCJI WstawElement i PodajElement*

```
void WstawElement(float tablica[], int rozmiar, int pozycja,
                  float wartosc, int& kod_bledu)
{
    if( pozycja<0 || pozycja>=rozmiar )
        kod_bledu = 1;
    else
    {
        tablica[pozycja] = wartosc;
        kod_bledu = 0;
    }
}
```

```
float PodajElement(float tablica[], int rozmiar, int pozycja,
                  int& kod_bledu)
{
    if( pozycja<0 || pozycja>=rozmiar )
    {
        kod_bledu = 1;
        return 0;
    }
    else
    {
        kod_bledu = 0;
        return tablica[pozycja];
    }
}
```

## [p\_02.cpp]

*// Operacja normalizacji zadanego indeksu*

*// poprzez wskazanie najbliższego elementu o poprawnym indeksie*

*//----- OPIS FUNKCJI -----*

*//                   Podczas wykonywania operacji na tablicy*  
*//                   zadana pozycja elementu powinna być normalizowana*  
*//                   poprzez wskazanie najbliższego "poprawnego" indeksu*

*// WstawElement - funkcja umieszczająca zadana <wartosc>*

*//                   na zadanej <pozycji> w liście*

*// PodajElement - funkcja zwracająca przez listę parametrów <wartosc>*

*//                   odczytana z zadanej <pozycji> w liście*

*//----- INTERFEJS FUNKCJI -----*

**void WstawElement(float tablica[], int rozmiar, int pozycja, float wartosc);**

**float PodajElement(float tablica[], int rozmiar, int pozycja);**

*//----- PRZYKŁADOWY PROGRAM -----*

**#include <iostream.h>**

**#include <conio.h>**

**void main()**

**{**  
  **float TAB[50];**  
  **int pozycja;**  
  **float x;**  
  **cout << "Podaj zapisywaną pozycję: ";**  
  **cin >> pozycja;**  
  **cout << "Podaj wartość: ";**  
  **cin >> x;**

**WstawElement( TAB, 50, pozycja, x);**

**cout << "Podaj odczytywaną pozycję: ";**  
  **cin >> pozycja;**

**x = PodajElement( TAB, 50, pozycja);**

**cout << "Odczytana wartość: " << x;**  
  **getch();**

**}**

*//----- PIERWSZA WERSJA IMPLEMENTACJI OPERACJI WSTAWIANIA / POBIERANIA -----*

**void WstawElement(float tablica[], int rozmiar, int pozycja, float wartosc)**

**{**  
  **if( pozycja<0 )**  
    **pozycja = 0;**  
  **if( pozycja>=rozmiar )**  
    **pozycja = rozmiar-1;**  
  **tablica[pozycja] = wartosc;**  
**}**

**float PodajElement(float tablica[], int rozmiar, int pozycja)**

**{**  
  **if( pozycja<0 )**  
    **pozycja = 0;**  
  **if( pozycja>=rozmiar )**  
    **pozycja = rozmiar-1;**  
  **return tablica[pozycja];**  
**}**

## [p\_03.cpp]

```
// Operacja cyklicznej normalizacji zadanego indeksu
// poprzez wyznaczenie wartosci "modulo"

//----- OPIS FUNKCJI -----
//      Podczas wykonywania operacji na tablicy
//      zadana pozycja elementu powinna byc normalizowana
//      poprzez wyznaczenie wartosci "modulo"
//
// WstawElement - funkcja umieszczajaca zadana <wartosc>
//                 na zadanej <pozycji> w liscie
// PodajElement - funkcja zwracajaca przez liste parametrów <wartosc>
//                 odczytana z zadanej <pozycji> w liscie

//----- INTERFEJS FUNKCJI -----
void WstawElement(float tablica[], int rozmiar, int pozycja, float wartosc);
float PodajElement(float tablica[], int rozmiar, int pozycja);

//----- PRZYKŁADOWY PROGRAM -----
#include <iostream.h>
#include <conio.h>

void main()
{
    float TAB[50];
    int   pozycja;
    float x;

    cout << "Podaj zapisywana pozycje: ";
    cin  >> pozycja;
    cout << "Podaj wartosc: ";
    cin  >> x;

    WstawElement( TAB, 50, pozycja, x);

    cout << "Podaj odczytywana pozycje: ";
    cin  >> pozycja;

    x = PodajElement( TAB, 50, pozycja);

    cout << "Odczytana wartosc: " << x;
    getch();
}

//----- DRUGA WERSJA IMPLEMENTACJI OPERACJI WSTAWIANIA / POBIERANIA -----
void WstawElement(float tablica[], int rozmiar, int pozycja, float wartosc)
{
    pozycja %= rozmiar;
    tablica[pozycja] = wartosc;
}

float PodajElement(float tablica[], int rozmiar, int pozycja)
{
    pozycja %= rozmiar;
    return tablica[pozycja];
}
```

## [p\_04.cpp]

```
// Reprezentacja abstrakcyjnej tablicy  
// z cyklicznym przesunięciem elementów w tablicy rzeczywistej  
// (i z normalizacją poprzez "modulo")  
  
//----- OPIS FUNKCJI -----  
//      Podczas wykonywania operacji na tablicy  
//      zadana pozycja elementu powinna być normalizowana  
//      poprzez wyznaczenie wartości "modulo"  
//  
// WstawElement - funkcja umieszczająca zadana <wartosc>  
//      na zadanej <pozycji> w liście  
// PodajElement - funkcja zwracająca przez listę parametrów <wartosc>  
//      odczytana z zadanej <pozycji> w liście  
  
//----- INTERFEJS FUNKCJI -----  
void WstawElement(float tablica[], int rozmiar, int pozycja, float wartosc);  
float PodajElement(float tablica[], int rozmiar, int pozycja);  
  
//----- PRZYKŁADOWY PROGRAM -----  
#include <iostream.h>  
#include <conio.h>  
void main()  
{  
    float TAB[50];  
    int   pozycja;  
    float x;  
  
    cout << "Podaj zapisywaną pozycję: ";  
    cin  >> pozycja;  
    cout << "Podaj wartość: ";  
    cin  >> x;  
  
    WstawElement( TAB, 50, pozycja, x);  
  
    cout << "Podaj odczytywaną pozycję: ";  
    cin  >> pozycja;  
  
    x = PodajElement( TAB, 50, pozycja);  
  
    cout << "Odczytana wartość: " << x;  
    getch();  
}  
  
//----- TRZECIA WERSJA IMPLEMENTACJI OPERACJI WSTAWIANIA / POBIERANIA -----  
#define PRZESUNIECIE 7  
  
void WstawElement(float tablica[], int rozmiar, int pozycja, float wartosc)  
{  
    pozycja = (pozycja+PRZESUNIECIE)%rozmiar;  
    tablica[pozycja] = wartosc;  
}  
  
float PodajElement(float tablica[], int rozmiar, int pozycja)  
{  
    pozycja = (pozycja+PRZESUNIECIE)%rozmiar;  
    return tablica[pozycja];  
}
```

## [p\_05.cpp]

// "obiekutowa" implementacja przykładów p\_02 / p\_03 / p\_04

// z wykorzystaniem przeciązonego operatora indeksu

//----- DEFINICJA TYPU DANYCH -----

```
struct T_tablica
{
    #define ROZMIAR 50
    float tab[ROZMIAR];
    float& operator[](int pozycja);
};
```

//----- PRZYKŁADOWY PROGRAM (5) -----

```
#include <iostream.h>
#include <conio.h>

void main()
{
    T_tablica TAB;
    int      pozycja;
    float    x;
    cout << "Podaj zapisywana pozycje: ";
    cin  >> pozycja;
    cout << "Podaj wartosc: ";
    cin  >> x;
    TAB[pozycja] = x;    // TAB.operator[](pozycja) = x;
    cout << "Podaj odczytywana pozycje: ";
    cin  >> pozycja;
    x = TAB[pozycja];    // x = TAB.operator[](pozycja);
    cout << "Odczytana wartosc: " << x;
    getch();
}
```

//-----RÓZNE IMPLEMENTACJE OPERACJI (operatora) INDEKSU -----

*//operacja indeksu analogiczna do przykladu (p\_02)*

```
float& T_tablica::operator[](int pozycja)
{
    if( pozycja<0 )
        pozycja = 0;
    if( pozycja>= ROZMIAR )
        pozycja = ROZMIAR-1;
    return tab[ pozycja ];
}
```

*//operacja indeksu analogiczna do przykladu (p\_03)*

```
float& T_tablica::operator[](int pozycja)
{
    pozycja %= ROZMIAR;
    return tab[ pozycja ];
}
```

*//operacja indeksu analogiczna do przykladu (p\_04)*

```
float& T_tablica::operator[](int pozycja)
{
    int rzeczywista_pozycja = (POCZATEK+pozycja)%ROZMIAR;
    return tab[rzeczywista_pozycja];
}
```

## [p\_06.cpp]

// przykładowa "obiektoowa" implementacja tablicy z kontrolowaniem zakresu indeksów  
// i wewnętrzna zmienna stanu informująca o występujących błędach zakresu

```
//----- DEFINICJA TYPU DANYCH -----  
struct T_tablica  
{  
    #define ROZMIAR 50  
    float tab[50];  
    int blad_zakresu;  
    T_tablica()  
        { blad_zakresu=0; }  
    float& operator[](int pozycja);  
    void ZerujStan()  
        { blad_zakresu=0; }  
};  
  
//----- IMPLEMENTACJA OPERACJI (operatora) INDEKSU -----  
float& T_tablica::operator[](int pozycja)  
{  
    if( blad_zakresu || pozycja<0 || pozycja>=ROZMIAR )  
        {  
            blad_zakresu=1;  
            return tab[0];  
        }  
    else  
        return tab[pozycja];  
}  
  
//----- PRZYKŁADOWY PROGRAM (5) -----  
#include <iostream.h>  
#include <conio.h>  
  
void main()  
{  
    T_tablica TAB;  
    int pozycja;  
    float x;  
  
    cout << "Podaj zapisywaną pozycję: ";  
    cin >> pozycja;  
    cout << "Podaj wartość: ";  
    cin >> x;  
  
    TAB[pozycja] = x;  
  
    if( TAB.blad_zakresu )  
        cout<<"\nWystąpił błąd zakresu podczas zapisu do tablicy\n";  
    TAB.ZerujStan();  
  
    cout << "Podaj odczytywaną pozycję: ";  
    cin >> pozycja;  
  
    x = TAB[pozycja];  
  
    if( TAB.blad_zakresu )  
        cout<<"\nWystąpił błąd zakresu podczas zapisu do tablicy\n";  
  
    cout << "Odczytana wartość: " << x;  
    getch();  
}
```



## [p\_07.cpp] - tworzenie abstrakcyjnych reprezentacji danych za pomoca wzorców

```
// przykładowe funkcje "Maksimum" dla typu "int" oraz "float"
int Maksimum( int a, int b )
{
    if( a>b )
        return a;
    else
        return b;
}

float Maksimum( float a, float b )
{
    if( a>b )
        return a;
    else
        return b;
}

// definicja abstrakcyjnego wzorca-szablonu funkcji "Maksimum"

template <class TTT>

TTT Maksimum( TTT a, TTT b )
{
    if( a>b )
        return a;
    else
        return b;
}

//----- PRZYKŁADOWY PROGRAM (7) -----
#include <iostream.h>
#include <conio.h>

void main()
{
    int x_i=5,w_i;
    w_i = Maksimum( x_i, 10 );    //wywołanie funkcji: int Maksimum( int , int );

    float x_f=5, w_f;
    w_f = Maksimum( x_f, 10.0 ); //wywołanie funkcji: float Maksimum(float ,float );

    double x_d=5,w_d;
    w_d = Maksimum( x_d, 10 ); //funkcja: double Maksimum(double,double);
                               //wygenerowana na podstawie wzorca

    unsigned long int x_uli=5, w_uli;
    w_uli = Maksimum( x_uli, 10 );

                               // funkcja: unsigned long Maksimum(unsigned long, unsigned long);
                               // również wygenerowana na podstawie tego samego wzorca
}
```

## [p\_08.cpp]

*// abstrakcyjna implementacja tablicy z przykladu [ p\_05 ] za pomoca szablonu klasy*

*//----- DEFINICJA NOWEGO TYPU ZA POMOCA WZORCA-SZABLONU KLASY -----*

```
template <class ELEMENT>
struct T_tablica
{
    #define ROZMIAR 50
    ELEMENT tab[ROZMIAR];
    ELEMENT& operator[](int pozycja);
};
```

*//-----SZABLON IMPLEMENTACJI OPERACJI (operatora) INDEKSU -----*

```
template <class ELEMENT>
ELEMENT& T_tablica<ELEMENT>::operator[](int pozycja)
{
    if( pozycja<0 )
        pozycja = 0;
    if( pozycja>= ROZMIAR )
        pozycja = ROZMIAR-1;
    return tab[ pozycja ];
}
```

*//----- PRZYKLADOWY PROGRAM (5) -----*

```
#include <iostream.h>
#include <conio.h>

void main()
{
    T_tablica<double> TAB_1;    //utworzenie tablicy elementów "double"
    T_tablica<long> TAB_2;    //utworzenie tablicy elementów "long"

    // wpisywanie danych do tablic
    TAB_1[ 0 ] = 10.0;
    TAB_2[ -7 ] = 10;

    //wyswietlanie zawartosci elementów tablic
    cout << "\nWartosc double z pierwszej tablicy: " << TAB_1[0];
    cout << "\nWartosc long z drugiej tablicy: " << TAB_2[-7];

    getch();
}
```