



Politechnika Białostocka  
Wydział Elektryczny  
Katedra Elektrotechniki, Energoelektroniki i Elektroenergetyki

Instrukcja  
do pracowni specjalistycznej z przedmiotu

## **Informatyka 2**

Kod przedmiotu: **EZ1E3012**

(studia niestacjonarne)

# **JĘZYK C - OPERATORY BITOWE**

Numer ćwiczenia

**INF30Z**

Autor:

dr inż. Jarosław Forenc

Białystok 2022

# Spis treści

<b>1. Opis stanowiska .....</b>	<b>3</b>
1.1. Stosowana aparatura .....	3
1.2. Oprogramowanie .....	3
<b>2. Wiadomości teoretyczne.....</b>	<b>3</b>
2.1. Operatory bitowe .....	3
2.2. Zastosowania operatorów bitowych.....	6
<b>3. Przebieg ćwiczenia.....</b>	<b>8</b>
<b>4. Literatura.....</b>	<b>9</b>
<b>5. Pytania kontrolne .....</b>	<b>10</b>
<b>6. Wymagania BHP .....</b>	<b>10</b>

---

**Materiały dydaktyczne przeznaczone dla studentów Wydziału Elektrycznego PB.**

© Wydział Elektryczny, Politechnika Białostocka, 2022 (wersja 5.1)

Wszelkie prawa zastrzeżone. Żadna część tej publikacji nie może być kopiowana i odtwarzana w jakiegokolwiek formie i przy użyciu jakichkolwiek środków bez zgody posiadacza praw autorskich.

# 1. Opis stanowiska

## 1.1. Stosowana aparatura

Podczas zajęć wykorzystywany jest komputer klasy PC z systemem operacyjnym Microsoft Windows 10.

## 1.2. Oprogramowanie

Na komputerach zainstalowane jest środowisko programistyczne Microsoft Visual Studio 2008 Standard Edition lub Microsoft Visual Studio 2008 Express Edition zawierające kompilator Microsoft Visual C++ 2008.

# 2. Wiadomości teoretyczne

## 2.1. Operatory bitowe

W języku C występuje 6 operatorów pozwalających na wykonywanie operacji na poszczególnych bitach liczb (Tabela 1). Operatory te można stosować jedynie do argumentów **całkowitych** typu **char**, **short**, **int**, **long** (ze znakiem lub bez).

Tabela 1. Operatory bitowe w języku C

Operator	Znaczenie	Opis
&	AND	dwuargumentowy operator koniunkcji bitowej
	OR	dwuargumentowy operator alternatywy bitowej
^	XOR	dwuargumentowy operator różnicy symetrycznej
~	NOT	jednoargumentowy operator uzupełnienia jedynekowego (zastępuje $0 \rightarrow 1$ , $1 \rightarrow 0$ )
>>		dwuargumentowy operator przesunięcia bitowego w prawo
<<		dwuargumentowy operator przesunięcia bitowego w lewo

Rozważmy następujący przykład deklaracji zmiennych:

```
unsigned char x = 106; /* 01101010 */
unsigned char y = 173; /* 10101101 */
unsigned char z;
```

Operator koniunkcji bitowej (&) ustawia jedynkę na każdej pozycji bitowej tam, gdzie oba bity są równe jeden. W pozostałych przypadkach ustawia zero.

<b>z = x &amp; y;</b>	<b>x</b> → 0 1 1 0 1 0 1 0
	<b>y</b> → 1 0 1 0 1 1 0 1
	-----
	<b>z</b> → 0 0 1 0 1 0 0 0

Operator alternatywy bitowej (|) ustawia jedynkę na każdej pozycji bitowej tam, gdzie przynajmniej jeden z bitów jest równy jeden.

<b>z = x   y;</b>	<b>x</b> → 0 1 1 0 1 0 1 0
	<b>y</b> → 1 0 1 0 1 1 0 1
	-----
	<b>z</b> → 1 1 1 0 1 1 1 1

Operator różnicy symetrycznej (^) ustawia jedynkę na każdej pozycji bitowej tam, gdzie bity są różne, a zero tam, gdzie bity są takie same.

<b>z = x ^ y;</b>	<b>x</b> → 0 1 1 0 1 0 1 0
	<b>y</b> → 1 0 1 0 1 1 0 1
	-----
	<b>z</b> → 1 1 0 0 0 1 1 1

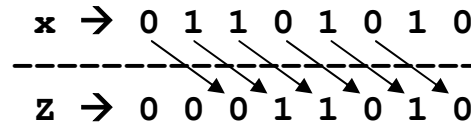
Operator uzupełnienia jedynekowego czyli negacji (~) zastępuje jedynkę zerem i zero jedynką.

<b>z = ~x;</b>	<b>x</b> → 0 1 1 0 1 0 1 0
	-----
	<b>z</b> → 1 0 0 1 0 1 0 1

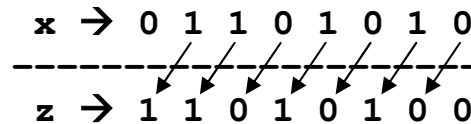
Operatory: << i >> przesuwają bity argumentu stojącego po lewej stronie operatora o liczbę pozycji określoną przez argument stojący po prawej stronie operatora. Drugi argument musi być liczbą dodatnią.

Przesunięcie w prawo powoduje pojawienie się na najstarszej pozycji 0 (dla liczb bez znaku) lub powielenie bitu znaku (dla liczb ze znakiem). Przy przesunięciu w lewo zwolnione (najmłodsze) bity wypełniane są 0.

`z = x >> 2;`

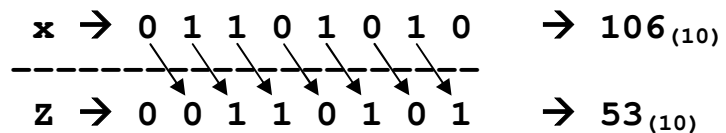


`z = x << 1;`

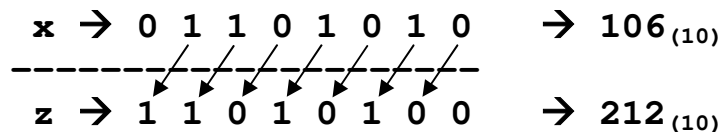


Przesunięcie w prawo o 1 pozycję odpowiada podzieleniu liczby przez 2, zaś przesunięcie w lewo o 1 pozycję - pomnożeniu przez 2.

`z = x >> 1;`



`z = x << 1;`



Nie należy mylić operatorów bitowych (`&` i `||`) z operatorami logicznymi (`&&` i `|||`). Zazwyczaj kompilator nie wykaże żadnego błędu, a jego wykrycie podczas wykonywania programu może być bardzo trudne.

```

int x = 1;
int y = 2;

if (x & y)                - wartość wyrażenia: 0 (fałsz),
    instrukcja;

if (x && y)               - wartość wyrażenia: 1 (prawda).
    instrukcja;

```

Język C udostępnia 5 złożonych (skrótowych) operatorów przypisania dotyczących operacji bitowych. Są to: `<<=`, `>>=`, `&=`, `|=`, `^=`.

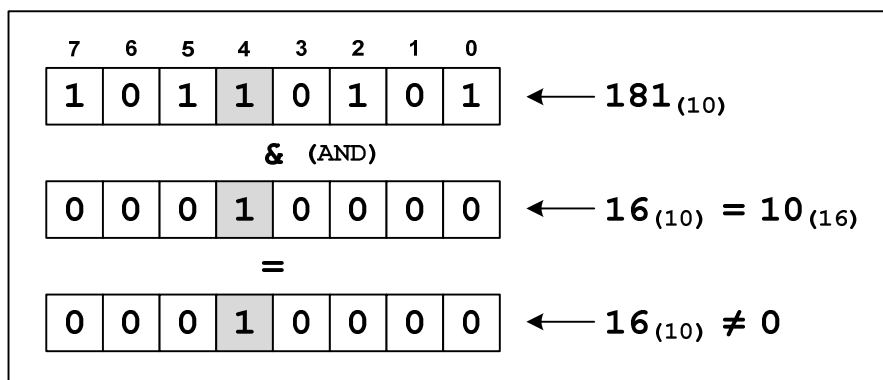
## 2.2. Zastosowania operatorów bitowych

Operatory bitowe wykorzystywane są do ustawiania i testowania wartości bitów w obszarach pamięci (programowanie niskopoziomowe).

Do sprawdzenia wartości wybranego bitu w bajcie stosowany jest operator koniunkcji bitowej (AND). Załóżmy, że chcemy sprawdzić wartość bitu nr 4 zmiennej **x** typu **unsigned char**. W tym celu wykonujemy operację koniunkcji bitowej zmiennej **x** i liczby mającej bit nr 4 równy 1 oraz wyzerowane pozostałe bity ( $0001\ 0000_{(2)} = 10_{(16)}$ ).

```
unsigned char x = 181; /* 10110101 */  
  
if ((x & 0x10) != 0)  
    printf("Bit nr 4 ma wartosc 1\n");  
else  
    printf("Bit nr 4 ma wartosc 0\n");
```

Jeśli w wyniku koniunkcji otrzymamy liczbę różną od zera, to bit nr 4 jest równy 1 (Rys. 1). W przeciwnym przypadku bit ten jest równy zero. Dodatkowe nawiasy w instrukcji warunkowej **if** są konieczne, gdyż operator logiczny **!=** ma wyższy priorytet niż operator bitowy **&**.



Rys. 1. Sprawdzenie wartości wybranego bitu w bajcie

Warunek logiczny w powyższej instrukcji **if** można zapisać w prostszej, równoważnej postaci:

```

if (x & 0x10)
    printf("Bit nr 4 ma wartosc 1\n");
else
    printf("Bit nr 4 ma wartosc 0\n");

```

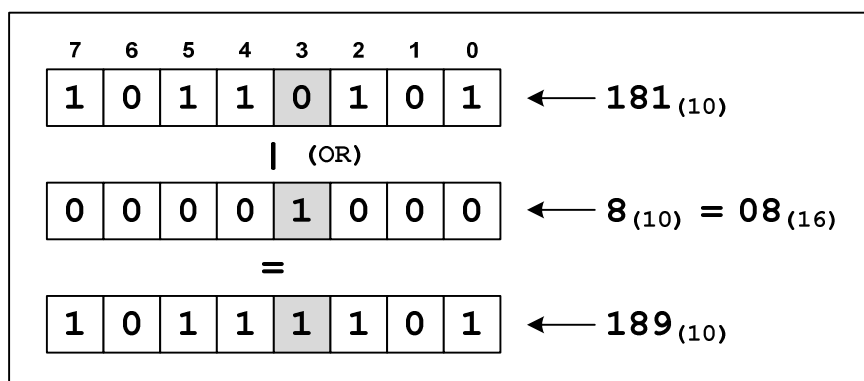
Do zapisania wartości 1 do wybranego bitu w bajcie (bez zmiany wartości pozostałych bitów) stosuje się operator alternatywy bitowej (OR). Załóżmy, że chcemy ustawić bit nr 3 zmiennej **x** typu **unsigned char**. W tym celu wykonujemy operację alternatywy bitowej zmiennej **x** i liczby mającej bit nr 3 równy 1 oraz wyzerowane pozostałe bity ( $0000\ 1000_{(2)} = 08_{(16)}$ ).

```

unsigned char x = 181; /* 10110101 */
x = x | 0x08;

```

Bit nr 3 otrzyma wartość 1, natomiast pozostałe bity nie ulegną zmianie (Rys. 2).



Rys. 2. Ustawienie wybranego bitu w bajcie

Do ustawienia bitu można zastosować skrócony operator przypisania |=.

```

x |= 0x08;

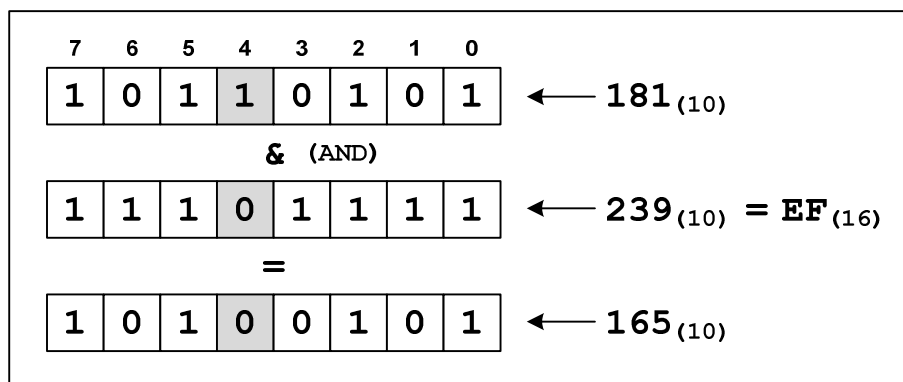
```

Do zapisania wartości 0 do wybranego bitu w bajcie (bez zmiany wartości pozostałych bitów) stosuje się operator koniunkcji bitowej (AND). Załóżmy, że chcemy wyzerować bit nr 4 zmiennej **x** typu **unsigned char**. W tym celu

wykonujemy operację koniunkcji bitowej zmiennej **x** i liczby mającej bit nr 4 równy 0 oraz ustawione na 1 wszystkie pozostałe bity (**1110 1111**<sub>(2)</sub> = **EF**<sub>(16)</sub>).

```
unsigned char x = 181; /* 10110101 */  
  
x = x & 0xEF;
```

Bit nr 4 otrzyma wartość 0, natomiast pozostałe bity nie ulegną zmianie (Rys. 3).



Rys. 3. Wyzerowanie wybranego bitu w bajcie

Do wyzerowania bitu można zastosować skrócony operator przypisania **&=**.

```
x &= 0xEF;
```

### 3. Przebieg ćwiczenia

Na pracowni specjalistycznej należy wykonać zadania wskazane przez prowadzącego zajęcia. W różnych grupach mogą być wykonywane inne zadania.

1. Zadeklaruj w programie zmienną typu **unsigned char**. Przypisz zmiennej liczbę z przedziału od **0** do **255**. Wyświetl wartości kolejnych bitów tej liczby.
2. Wczytaj z klawiatury liczbę całkowitą. Stosując operacje bitowe sprawdź, czy liczba jest parzysta czy nieparzysta.



3. Napisz program zawierający funkcję **GetBit()**, która dla liczby typu **unsigned int** (pierwszy argument funkcji) zwraca wartość bitu o numerze będącym drugim argumentem tej funkcji. Zakładamy, że najmłodszy bit liczby ma numer 0. Jeśli numer bitu jest niepoprawny funkcja powinna zwrócić wartość **-1**.
4. Napisz program zawierający funkcje **Set\_1()** i **Set\_0()**. Funkcja **Set\_1()** powinna w liczbie typu **unsigned int** (pierwszy argument funkcji) zapisać wartość **1** w bicie o numerze będącym drugim argumentem tej funkcji. Natomiast funkcja **Set\_0()** powinna w liczbie typu **unsigned int** (pierwszy argument funkcji) zapisać wartość **0** w bicie o numerze będącym drugim argumentem tej funkcji. Pozostałe bity nie powinny zmieniać się. Zakładamy, że najmłodszy bit liczby ma numer **0**. Dodaj w funkcjach sprawdzenie poprawności numeru bitu.

## 4. Literatura

- [1] Prata S.: Język C. Szkoła programowania. Wydanie VI. Helion, Gliwice, 2016.
- [2] Kernighan B.W., Ritchie D.M.: Język ANSI C. Programowanie. Wydanie II. Helion, Gliwice, 2010.
- [3] Deitel P.J., Deitel H.: Język C. Solidna wiedza w praktyce. Wydanie VIII. Helion, Gliwice, 2020.
- [4] Kochan S.G.: Język C. Kompendium wiedzy. Wydanie IV. Helion, Gliwice, 2015.
- [5] King K.N.: Język C. Nowoczesne programowanie. Wydanie II. Helion, Gliwice, 2011.
- [6] <http://www.cplusplus.com/reference/clibrary> - C library - C++ Reference
- [7] <https://cpp0x.pl/dokumentacja/standard-C/1> - Standard C

## 5. Pytania kontrolne

1. Scharakteryzuj operatory bitowe występujące w języku C.
2. Wyjaśnij różnice pomiędzy operatorami bitowymi i logicznymi.
3. W jaki sposób można zapisać 1 lub 0 do dowolnego bitu liczby (bez zmiany wartości pozostałych bitów)?

## 6. Wymagania BHP

Warunkiem przystąpienia do praktycznej realizacji ćwiczenia jest zapoznanie się z instrukcją BHP i instrukcją przeciw pożarową oraz przestrzeganie zasad w nich zawartych.

W trakcie zajęć laboratoryjnych należy przestrzegać następujących zasad.

- Sprawdzić, czy urządzenia dostępne na stanowisku laboratoryjnym są w stanie kompletnym, nie wskazującym na fizyczne uszkodzenie.
- Jeżeli istnieje taka możliwość, należy dostosować warunki stanowiska do własnych potrzeb, ze względu na ergonomię. Monitor komputera ustawić w sposób zapewniający stałą i wygodną obserwację dla wszystkich członków zespołu.
- Sprawdzić prawidłowość połączeń urządzeń.
- Załączenie komputera może nastąpić po wyrażeniu zgody przez prowadzącego.
- W trakcie pracy z komputerem zabronione jest spożywanie posiłków i picie napojów.
- W przypadku zakończenia pracy należy zakończyć sesję przez wydanie polecenia wylogowania. Zamknięcie systemu operacyjnego może się odbywać tylko na wyraźne polecenie prowadzącego.
- Zabronione jest dokonywanie jakichkolwiek przełączeń oraz wymiana elementów składowych stanowiska.

- Zabroniona jest zmiana konfiguracji komputera, w tym systemu operacyjnego i programów użytkowych, która nie wynika z programu zajęć i nie jest wykonywana w porozumieniu z prowadzącym zajęcia.
- W przypadku zaniku napięcia zasilającego należy niezwłocznie wyłączyć wszystkie urządzenia.
- Stwierdzone wszelkie braki w wyposażeniu stanowiska oraz nieprawidłowości w funkcjonowaniu sprzętu należy przekazywać prowadzącemu zajęcia.
- Zabrania się samodzielnego włączania, manipulowania i korzystania z urządzeń nie należących do danego ćwiczenia.
- W przypadku wystąpienia porażenia prądem elektrycznym należy niezwłocznie wyłączyć zasilanie stanowiska. Przed odłączeniem napięcia nie dotykać porażonego.