Informatyka 1

Politechnika Białostocka - Wydział Elektryczny

Elektrotechnika, semestr II, studia stacjonarne I stopnia

Rok akademicki 2018/2019

Wykład nr 7 (12.04.2019)

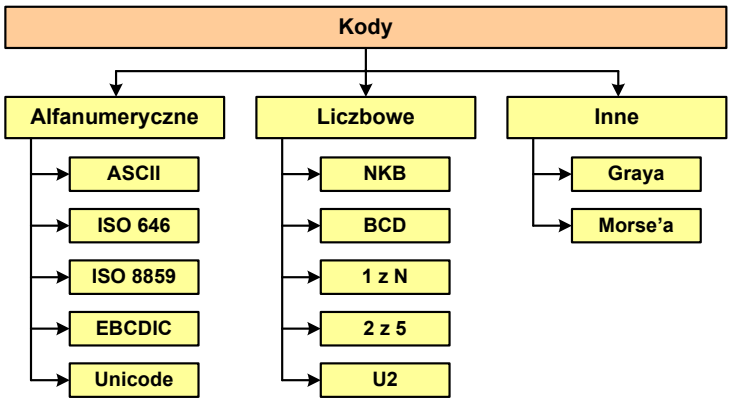
dr inż. Jarosław Forenc

Plan wykładu nr 7

- Kodowanie liczb
  - kod 2 z 5, kod Graya
- Język C
  - pętla for, operatory ++ i –
- Reprezentacja liczb całkowitych
  - liczby bez znaku

Kodowanie

- Kodowanie - proces przekształcania jednego rodzaju postaci informacji na inną postać



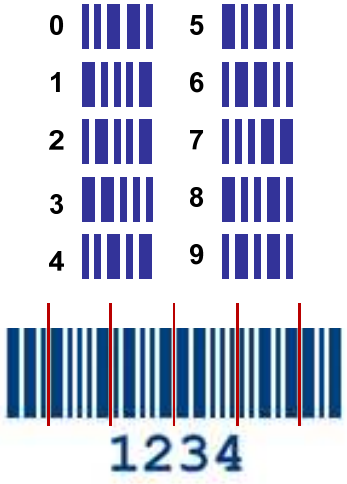
Kody liczbowe - Kod 2 z 5

- Kod 5-bitowy: 2 bity zawsze równe 1, a 3 bity zawsze równe 0
- Koduje 10 znaków (cyfry dziesiętne), kody nie są wzajemnie jednoznaczne (ta sama wartość może być zakodowana w różny sposób)
- Kod stałowagowy
- Kod detekcyjny
- Stosowany głównie w kodach kreskowych

Liczba dziesiętna	2 z 5 (01236)	2 z 5 (01234)	2 z 5 (74210)
0	01100	01100	11000
1	11000	11000	00011
2	10100	10100	00101
3	10010	10010	00110
4	01010	01010	01001
5	00110	00110	01010
6	10001	10001	01100
7	01001	01001	10001
8	00101	00101	10010
9	00011	00011	10100

Kody liczbowe - Kod 2 z 5 Industrial (1960 r.)

- Jednowymiarowy kod kreskowy kodujący cyfry: 0 ÷ 9
- Znak to 5 pasków: 2 szerokie i 3 wąskie
- Szeroki pasek jest wielokrotnością wąskiego, szerokości muszą być takie same dla całego kodu
- Struktura kodu:
  - start: 11011010
  - numer
  - stop: 11010110



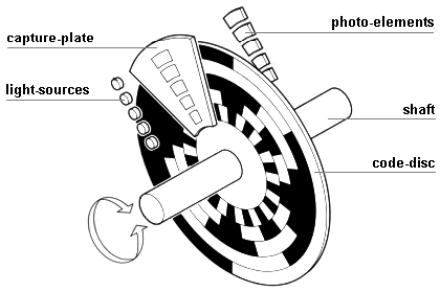
Kod Graya (refleksyjny)

- Kod dwójkowy, bezwagowy, niepozycyjny
- Dwa kolejne słowa kodowe różnią się stanem jednego bitu
- Kod cykliczny - ostatni i pierwszy wyraz również różnią się stanem jednego bitu

kod 1-bitowy	kod 2-bitowy	kod 3-bitowy
0	00	000
1	01	001
	11	011
	10	010
		110
		111
		101
		100

Kod Graya

- Stosowany w przetwornikach analogowo-cyfrowych, do cyfrowego pomiaru analogowych wielkości mechanicznych (np. kąt obrotu)



http://tams-www.informatik.uni-hamburg.de
/applets/hades/webdemos/10-gates/15-graycode/dual2gray.html

Język C - suma kolejnych 10 liczb: 1+2+...+10

```
#include <stdio.h>

int main(void)
{
    int suma;

    suma = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;

    printf("Suma wynosi: %d\n",suma);

    return 0;
}
```

```
Suma wynosi: 55
```

Język C - suma kolejnych 100 liczb: 1+2+...+100

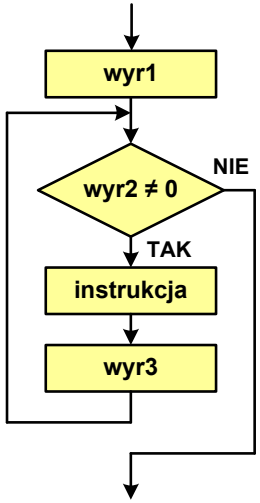
```
#include <stdio.h>

int main(void)
{
    int suma=0, i;

    for (i=1; i<=100; i=i+1)
        suma = suma + i;

    printf("Suma wynosi: %d\n",suma);

    return 0;
}
```

```
Suma wynosi: 5050
```


Język C - pętla for

```
for (wyr1; wyr2; wyr3)
    instrukcja
```

- wyr1, wyr2, wyr3 - dowolne wyrażenia w języku C
- Instrukcja:
  - prosta - jedna instrukcja zakończona średnikiem
  - złożona - jedna lub kilka instrukcji objętych nawiasami klamrowymi



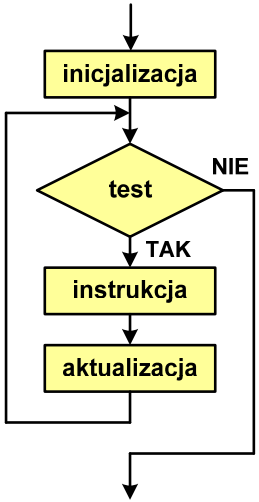
Język C - pętla for

- Najczęściej stosowana postać pętli for

```
int i;
for (i = 0; i < 10; i = i + 1)
    instrukcja
```

- Instrukcja zostanie wykonana 10 razy (dla i = 0, 1, 2, ... 9)
- Funkcje pełnione przez wyrażenia

```
for (inicjalizacja;test;aktualizacja)
    instrukcja
```



Język C - pętla for (wyświetlenie tekstu)

```
#include <stdio.h>

int main(void)
{
    int i;

    for (i=0; i<5; i=i+1)
        printf("Programowanie nie jest trudne\n");

    return 0;
}
```

```
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
Programowanie nie jest trudne
```

Język C - pętla for (suma liczb: 1 + 2 + … + N)

```
#include <stdio.h>
#define N 1234

int main(void)
{
    int i, suma=0;

    for (i=1; i<=N; i++)
        suma = suma + i;

    printf("Suma %d liczb to %d\n", N, suma);

    return 0;
}
```

```
Suma 1234 liczb to 761995
```


Język C - pętla for (przykłady)

```
for (i=0; i<10; i++)
    printf("%d ",i);
```

```
0 1 2 3 4 5 6 7 8 9
```

```
for (i=0; i<10; i++)
    printf("%d ",i+1);
```

```
1 2 3 4 5 6 7 8 9 10
```

```
for (i=1; i<=10; i++)
    printf("%d ",i);
```

```
1 2 3 4 5 6 7 8 9 10
```


Język C - pętla for (przykłady)

```
for (i=1; i<10; i=i+2)
    printf("%d ",i);
```

```
1 3 5 7 9
```

```
for (i=10; i>0; i--)
    printf("%d ",i);
```

```
10 9 8 7 6 5 4 3 2 1
```

```
for (i=-9; i<=9; i=i+3)
    printf("%d ",i);
```

```
-9 -6 -3 0 3 6 9
```


Język C - pętla for (break, continue)

- W pętli for można stosować instrukcje skoku: break i continue

```
int i;

for (i=1; i<10; i++)
{
    if (i%2==0)
        continue;
    if (i%7==0)
        break;
    printf("%d\n",i);
}
```

- continue przerywa bieżącą iterację i przechodzi do obliczania wyr3
- break przerywa wykonywanie pętli

```
1 3 5
```

Język C - pętla for (najczęstsze błędy)

- Postawienie średnika na końcu pętli for

```
int i;
for (i=0; i<10; i++);
printf("%d ",i);
```

```
10
```

- Przecinki zamiast średników pomiędzy wyrażeniami

```
int i;
for (i=0, i<10, i++)
    printf("%d ",i);
```

Błąd kompilacji!

```
error C2143: syntax error : missing ';' before ')'
```


Język C - pętla for (najczęstsze błędy)

- Błędny warunek - brak wykonania instrukcji

```
int i;
for (i=0; i>10; i++)
    printf("%d ",i);
```

- Błędny warunek - pętla nieskończona

```
int i;
for (i=1; i>0; i++)
    printf("%d ",i);
```

```
1 2 3 4 5 6 7 8 9 ...
```


Język C - pętla nieskończona

```
for (wyr1; wyr2; wyr3)
    instrukcja
```

- Wszystkie wyrażenia (wyr1, wyr2, wyr3) w pętli for są opcjonalne

```
for ( ; ; )
    instrukcja
```

- pętla nieskończona

- W przypadku braku wyr2 przyjmuje się, że jest ono prawdziwe

Język C - zagnieżdżanie pętli for

- Jako instrukcja w pętli for może występować kolejna pętla for

```
int i, j;
for (i=1; i<=3; i++)          // pętla zewnętrzna
    for (j=1; j<=2; j++)      // pętla wewnętrzna
        printf("i: %d   j: %d\n",i,j);
```

```
i: 1   j: 1
i: 1   j: 2
i: 2   j: 1
i: 2   j: 2
i: 3   j: 1
i: 3   j: 2
```

Język C - operator inkrementacji (++)

- Jednoargumentowy operator ++ zwiększa wartość zmiennej o 1 (nie wolno stosować go do wyrażeń)
- Operator ++ może występować jako przedrostek lub przyrostek

Zapis	Nazwa	Znaczenie
`++x`	preinkrementacji	wartość zmiennej jest modyfikowana przed jej użyciem
`x++`	postinkrementacji	wartość zmiennej jest modyfikowana po użyciu jej poprzedniej wartości

Język C - operator inkrementacji (++)

- Przykład

```
int x = 1, y;
y = 2 * ++x;
```

```
int x = 1, y;
y = 2 * x++;
```

- Kolejność operacji

```
++x          x = 2
2 * ++x      2 * 2
y = 2 * ++x  y = 4
```

```
2 * x        2 * 1
y = 2 * x    y = 2
x++          x = 2
```

- Wartości zmiennych

```
x = 2    y = 4
```

```
x = 2    y = 2
```


Język C - operator inkrementacji (++)

- Miejsce umieszczenia operatora ++ nie ma znaczenia w przypadku instrukcji typu:

```
x++;
++x;
```

równoważne

```
x = x + 1;
```

- Nie należy stosować operatora ++ do zmiennych pojawiających się w wyrażeniu więcej niż jeden raz

```
x = x++;
x = ++x;
```

- Zgodnie ze standardem języka C wynik powyższych instrukcji jest niezdefiniowany

Język C - operator dekrementacji (--)

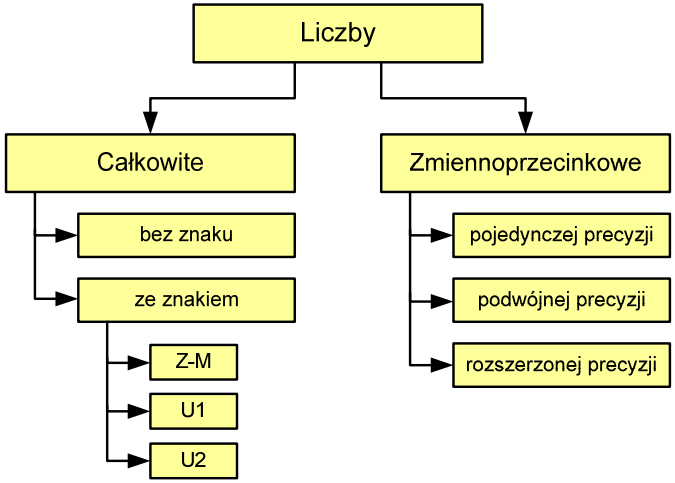
- Jednoargumentowy operator -- zmniejsza wartość zmiennej o 1 (nie wolno stosować go do wyrażeń)
- Operator -- może występować jako przedrostek lub przyrostek

Zapis	Nazwa	Znaczenie
`--x`	predekrementacji	wartość zmiennej jest modyfikowana przed jej użyciem
`x--`	postdekrementacji	wartość zmiennej jest modyfikowana po użyciu jej poprzedniej wartości

Język C - priorytet operatorów ++ i --

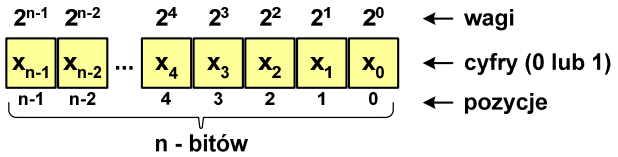
Priorytet	Operator / opis
1	`++` `--` (przyrostki) `()` `[]` `.` `->`
2	`++` `--` (przedrostki) `sizeof` `(typ)` `+` `-` `!` `~` `*` `&` (jednoargumentowe)
3	`*` `/` `%`
4	`+` `-` (dwuargumentowe)
5	`<<` `>>`
6	`<` `>` `<=` `>=`
7	`==` `!=`
8	`&` (bitowy)
9	`^`

Reprezentacja liczb w systemach komputerowych



Liczby całkowite bez znaku

- Zapis liczby w systemie dwójkowym:



- Używając n-bitów można zapisać liczbę z zakresu:

$$X_{(2)} = \langle 0,\ 2^n - 1 \rangle$$

$8 - bitów$	0 ... 255
$16 - bitów$	0 ... 65 535
$32 - bity$	0 ... 4 294 967 295
$64 - bity$	0 ... 18 446 744 073 709 551 615

18 trylionów 446 biliardów 744 biliony 73 miliardy 709 milionów 551 tysięcy 615

Liczby całkowite bez znaku w języku C

- Typy zmiennych całkowitych bez znaku stosowane w języku C:

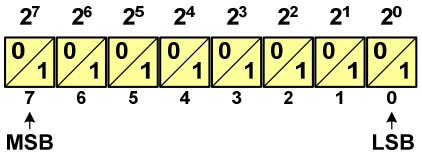
Nazwa typu	Rozmiar (bajty)	Zakres wartości
unsigned char	1 bajt	0 ... 255
unsigned short int	2 bajty	0 ... 65 535
unsigned int	4 bajty	0 ... 4 294 967 295
unsigned long int	4 bajty	0 ... 4 294 967 295
unsigned long long int	8 bajtów	0 ... 18 446 744 073 709 551 615

- W nazwach typów short i long można pominąć słowo int:

unsigned short int	→	unsigned short
unsigned long int	→	unsigned long
unsigned long long int	→	unsigned long long

Liczby całkowite bez znaku w języku C

- Typ unsigned char (1 bajt):

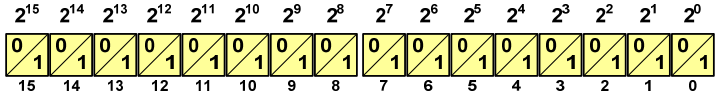


  - MSB (Most Significant Bit) - najbardziej znaczący bit, najstarszy bit, największa waga
  - LSB (Least Significant Bit) - najmniej znaczący bit, najmłodszy bit, najmniejsza waga

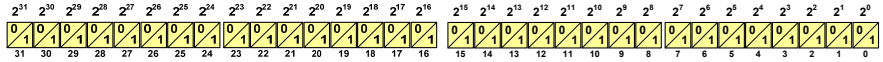
- Zakres wartości:
  - dolna granica: $0000\ 0000_{(2)} = 00_{(16)} = 0_{(10)}$
  - górna granica: $1111\ 1111_{(2)} = FF_{(16)} = 255_{(10)}$

Liczby całkowite bez znaku w języku C

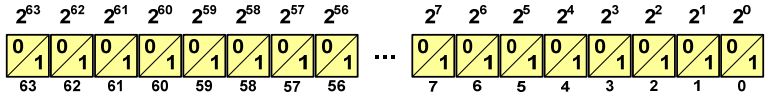
- Typ unsigned short int (2 bajty):



- Typy unsigned int (4 bajty) i unsigned long int (4 bajty):



- Typ unsigned long long int (8 bajtów):



Liczby całkowite bez znaku w języku C

```
unsigned short int:      65535 0 1
unsigned int:            4294967295 0 1
unsigned long int:       4294967295 0 1
unsigned long long int: 18446744073709551615 0 1
```

```
#include <stdi
int main()   /* przepełnienie zmiennej, ang. integer overflow */
{
   unsigned short int      usi = 65535;
   unsigned int             ui = 4294967295;
   unsigned long int       uli = 4294967295;
   unsigned long long int ulli = 18446744073709551615;

   printf("unsigned short int:     %hu %hu %hu\n",usi,usi+1,usi+2);
   printf("unsigned int:           %u %u %u\n",ui,ui+1,ui+2);
   printf("unsigned long int:      %lu %lu %lu\n",uli,uli+1,uli+2);
   printf("unsigned long long int: %llu %llu %llu\n",
                                    ulli,ulli+1,ulli+2);
   return 0;
}
```


Liczby całkowite bez znaku w języku C

```
unsigned short int:      1 0 65535
unsigned int:            1 0 4294967295
unsigned long int:       1 0 4294967295
unsigned long long int: 1 0 18446744073709551615
```

```
#include <stdi
int main()   /* przepełnienie zmiennej, ang. integer overflow */
{
   unsigned short int      usi = 1;
   unsigned int             ui = 1;
   unsigned long int       uli = 1;
   unsigned long long int ulli = 1;

   printf("unsigned short int:     %hu %hu %hu\n",usi,usi-1,usi-2);
   printf("unsigned int:           %u %u %u\n",ui,ui-1,ui-2);
   printf("unsigned long int:      %lu %lu %lu\n",uli,uli-1,uli-2);
   printf("unsigned long long int: %llu %llu %llu\n",
                                    ulli,ulli-1,ulli-2);
   return 0;
}
```

Koniec wykładu nr 7

Dziękuję za uwagę!