Środowisko programisty + Języki Programowania

Wykład 4

dr Maciej Dziemiańczuk

Instytut Informatyki,
Wydział Matematyki, Fizyki i Informatyki, Uniwersytet Gdański

Rok akademicki 2022/2023

Zanim zaczniemy...

Co było na ostatnim wykładzie?

- stałe
- instrukcje sterujące
- `if`, `if-else`

Co będzie dzisiaj?

- instrukcja `switch`
- pętla `while`, `do-while`
- pętla `for`,
- tablice zmiennych.

Słowa kluczowe w języku ANSI C99 (34)

Jakie słowa kluczowe języka C poznaliśmy ostatnio?

```
auto
break
case
char
const
continue
default
do
double
else
enum
extern
float
for
goto
if
inline
int
long
register
restrict
return
short
signed
sizeof
static
struct
switch
typedef
union
unsigned
void
volatile
while
```

Instrukcja sterująca `switch`

Instrukcja decyzyjna wielowariantowa switch

Składnia

```
switch (wyrazenie_calkowitoliczbowe)
{
    case wyr_stale1: instrukcje;
    case wyr_stale2: instrukcje;
    ...
    default: instrukcje;
}
```

Często używana z instrukcją break do przerywania działania instrukcji sterującej.

Przykład

```
int n = ...;
switch (n)
{
    case 2: printf("Liczba 2 "); break;
    case 5: printf("Liczba 5 "); break;
    case 9: printf("Liczba 9 "); break;
    default: printf("Inna liczba niz 2, 5, 9 "); break;
}
```

Instrukcja sterująca `switch`

Uwaga! *W instrukcji* ***switch*** *możemy łączyć warunki.*

Przykład

Kod programu: switch.c

```
1  #include <stdio.h>
2
3  int main()
4  {
5      char znak = '?';
6      switch (znak)
7      {
8          case 'A': printf("A"); break;
9          case 'B': printf("B");
10         case 'C':
11         case 'D': printf("D"); break;
12         case 'E': printf("E");
13         default: printf("?"); break;
14     }
15
16     return 0;
17 }
```

Co zostanie wyświetlone na ekranie dla różnych wartości zmiennej `znak`?

Pętla `while`

Pętla while

Składnia

```
while (wyrazenie)
   instrukcja;
```

Najpierw obliczane jest **wyrazenie**. Jeśli ma wartość niezerową to wykonywana jest **instrukcja** (lub blok instrukcji). Następnie ponownie sprawdzane jest **wyrażenie** i jeśli ma wartość niezerową to wykonywana jest **instrukcja**. Itd., itd... Jeśli **wyrazenie** ma wartość zerową, następuje wyjście z pętli.

Przykład

```
// Kod programu: while.c
int n = 0;
while (n < 10)
{
   printf("liczba %d \n", n);
   n = n + 1;
}
```

Schemat blokowy

Uwaga! *na pętle nieskończone!*

Pętle nieskończone

Przykład

```
int n = 0;
while (n >= 0)
{
    printf("liczba %d \n", n);
    n = n + 2;
}
```

Wydruk

```
liczba 0
liczba 2
liczba 4
...
```

Czy aby na pewno to pętla nieskończona?

Schemat blokowy

Pętla while

Przykład pętli while

Kod programu: while-number.c

```c
#include <stdio.h>

int main()
{
    int k, n;
    scanf("%i", &n);

    while (n)
    {
        k = n%10;
        n = n / 10;

        printf("%i,", k);
    }

    return 0;
}
```

Co wypisze ten program dla wczytanej liczby n=1052?

Uwaga na błędy

Znajdź i wyjaśnij wszystkie błędy.

Przykład 1

```c
// Kod programu: while-blad-1.c
float n = 0.0f;
while ( n != 1.0f )
{
    printf("liczba %f \n", n);

    n += 0.1f;
}
```

Przykład 2

```c
// Kod programu: while-blad-2.c
int n = 10;
while (n > 0);
{
    printf("liczba %i \n", n);
    n--;
}
```

Pętla `do-while`

Pętla do-while

Składnia

```
do
    instrukcja;
while (wyrazenie);
```

Podobnie jak przy pętli `while` z tą różnicą, że najpierw wykonywana jest `instrukcja` (lub blok instrukcji), a następnie sprawdzane jest `wyrazenie`, które ewentualnie uruchamia kolejne przebiegi pętli.

Przykład

```
// Kod programu: do-while.c
int pin = 0;
do
{
    printf("podaj pin: ");
    scanf("%d", &pin);
} while (pin != 4321);

printf("witamy w systemie\n");
```

Schemat blokowy

Uwaga! *Pętla `do-while` wykona się zawsze co najmniej jeden raz.*

Pętla `for`

Pętla for

Składnia

```
for (wyr_A; wyr_B; wyr_C)
    instrukcja;
```

Najpierw wykonywane jest wyrażenie `wyr_A`. Następnie sprawdzany warunek `war_B` i jeżeli jego wartość jest niezerowa to wykonywana jest `instrukcja`. Po zakończeniu wykonywania instrukcji, wykonywane jest `wyr_C` i ponownie sprawdzany jest warunek `wyr_B`. Jeśli jego wartość jest niezerowa to wykonywana jest ponownie `instrukcja`, a jeżeli jest zerowa to opuszczana jest pętla `for`.

Przykład

```
// Kod programu: for.c
int i, sum = 0;

for (i = 0; i < 20; i++)
{
    sum = sum + i;
    printf("i = %i \n", i);
}
printf("sum = %i \n", sum);
```

Pętla for

Składnia pętli for

```
for (wyr_A; wyr_B; wyr_C)
    instrukcja;
```

Schemat blokowy:

Równoważna pętla while

```
wyr_A;
while (wyr_B)
{
    instrukcja;
    wyr_C;
}
```

Przykład użycia pętli for oraz równoważnej pętli while

```
for (n = 5; n <= 25; n += 3)
{
    printf("liczba %i \n", n);
}
```

```
n = 5;
while (n <= 25) {
    printf("liczba %i \n", n);
    n += 3;
}
```

Pętla for

Składnia pętli for

```
for (wyr_A; wyr_B; wyr_C)
   instrukcja;
```

Uwaga! *Wszystkie trzy wyrażenia pętli* ***for*** *są opcjonalne, średniki są obowiązkowe, brak* ***wyr_B*** *zastępowany jest wartością niezerową.*

Przykład

Kod programu: for-1.c

```
1  #include <stdio.h>
2  int main()
3  {
4      int i = 0, sum = 0;
5
6      for ( ; i < 20; )
7      {
8          sum = sum + i;
9          printf("i = %i \n", i);
10         i += 2;
11     }
12
13     printf("sum = %i \n", sum);
14     return 0;
15 }
```

Operatora przecinka ,

```
wyr_A, wyr_B;
```

Wyrażenia obliczane są od lewej do prawej, a wartością całego wyrażenia jest prawa strona.

Przykład

```
1   int a=1, b=2;                // to nie jest operator przecinka
2   int c = (a += 2, a + b);     // operator przecinka
3   printf("c = %i\n", c);
```

Przykład dla pętli `for`

```
1   // Kod programu: for-comma.c
2   int i, j;
3   for (i=0, j=0; i < 10; i++,j+=5)
4   {
5       printf("%i %i\n", i, j);
6   }
```

Dla odważnych: *zobacz "sequence point problem" w C.*

Instrukcja `break` i `continue`

Instrukcja break

Instrukcja **break** powoduje natychmiastowe opuszczenie pętli (lub **switcha**).

Schemat

```
while/for (...) {
    instrukcja_A;

    if (warunek_B)
        break;

    instrukcja_B;
}
```

Przykład dla pętli while

```
1  // Kod programu: while-break.c
2  int n = 0, sum = 0;
3  while (n < 30)
4  {
5      n = n + 1;
6      if (n == 13) break;
7      sum = sum + n;
8  }
9  printf("%i\n", sum);
```

Pętla nieskończona i operacja break

```
// Kod programu: while-inf-break.c
int n = 0;
while (1)
{
    printf("liczba %d \n", n);

    if (n > 10) break;

    n = n + 1;
}
```

Przykład zastosowania pętli nieskończonej

```
Menu programu kalkulator:
1 - dodawanie dwoch liczb
2 - mnozenie dwoch liczb
3 - potegowanie
4 - pierwiastkowanie
x - wyjscie z programu

Wybierz:
```

Operacja continue

Operacja **continue** powoduje pominięcie wykonywania kolejnych instrukcji pętli oraz przechodzi do kolejnego obrotu pętli, który zaczyna się od sprawdzenia warunku pętli.

```
for/while {
    instrukcje;

    if (wyrazenie)
        continue;

    inne_instrukcje;
}
```

Przykład pętli **for** i operacji **continue**

```
1  // Suma liczb od 1 do 20 bez 13
2  int suma = 0, n = 1;
3  for (n = 1; n < 20; ++n)
4  {
5      if (n == 13) continue;
6      suma = suma + n;
7  }
8  printf("Suma %i \n", suma);
```

Operacja continue i pętla while

Uwaga na błędy!

Pętla `while` wraz z operacją `continue`.

Przykład

```
1   // Kod programu: while-continue.c
2   // Suma liczb od 1 do 20 bez 13
3   int suma = 0;
4   int n = 1;
5   while( n < 20 )
6   {
7       if (n == 13) continue;
8
9       suma = suma + n;
10      n ++;
11  }
12  printf("Suma %i \n", suma);
```

Jak sobie radzić z niedziałającym kodem?

Pętle zagnieżdżone

Pętla w pętli

Przykład

```
// Kod programu for-for.c
int i,j;
for (i=1; i<=10; i++)
{
    for (j=1; j<=10; j+=2)
    {
        printf("%i  %i \n", i, j);
    }
}
```

Przykład

```
// Kod programu for-while.c
int n,k,i;
for (i=1; i<=100; i++)
{
    n = i;
    k = 1;
    while (n = n/2)
        k++;
    printf("%i,", k);
}
```

Program generujący choinkę w konsoli

Pętla w pętli w pętli w pętli ...

Przykład

Kod programu: for-for-b.c

```
1   #include <stdio.h>
2   #define SIZE   2
3   int main()
4   {
5       int a, b, c, d, e;
6
7       for (a=0; a<SIZE; a++)
8       for (b=0; b<SIZE; b++)
9       for (c=0; c<SIZE; c++)
10      for (d=0; d<SIZE; d++)
11      for (e=0; e<SIZE; e++)
12      {
13          printf("%i%i%i%i%i\n",a,b,c,d,e);
14      }
15
16      return 0;
17  }
```

Pytanie: *Co ten program robi?*

Operacja `goto`...

Używać czy nie używać?

Operacja goto i etykiety

Bezwarunkowe przejście do instrukcji etykietowanej.

```
printf("A");
goto finish;        // <-- goto

printf("B");
...
finish:          // <-- etykieta
   printf("C");
```

Uwaga! *Unikamy instrukcji* ***goto*** *w programach.*

Przykład usprawiedliwionego zastosowania instrukcji goto w zagnieżdżonych pętlach

```
1  // Kod programu: goto.c
2  for (i=0; i<30; i++) {
3     for (j=0; j<45; ++j) {
4        printf("%d x %d \n", i, j);
5        if ( i == 13 && j == 20)
6           goto koniec;
7     }
8  }
9  koniec:
```

Tablice

Tablice – przykład

Zadanie
Napisz program, który wczytuje 5 liczb i wypisuje je w odwrotnej kolejności.

Przykład rozwiązania

```
1  int a, b, c, d, e;
2
3  printf("Podaj piec liczb calkowitych: ");
4  scanf("%i %i %i %i %i", &a, &b, &c, &d, &e);
5
6  printf("%i %i %i %i %i", e, d, c, b, a);
```

Pytanie: *Jak zmodyfikować program, aby działał dla 20 liczb?*

1. Deklaracja zmiennej tablicowej

```
TYP nazwa_zmiennej[ROZMIAR];
```

Przykład

```
int tablica_liczb[10];
float tab2[100];
unsigned char tablica_znakow[60];
```

2. Odwoływanie się do elementów tablicy

Elementami tablicy o nazwie `nazwa` i rozmiarze N są

$$\texttt{nazwa[0]}, \texttt{nazwa[1]}, \ldots, \texttt{nazwa[}N-1\texttt{]}.$$

Przykład

```
int tablica[10];

tablica[0] = 1;
tablica[1] = tablica[0];
tablica[2] = tablica[0] + tablica[1];
tablica[9] = ...; // ostatni element tablicy
```

3. Tablica i pętla `for`

- **Deklaracja zmiennej tablicowej**

```
#define SIZE 20
int Numbers[SIZE];
```

- **Ustawianie** elementów tablicy

```
for (int i=0; i<SIZE; i++)
    Numbers[i] = i*2;
```

Ciekawostka: *od wersji C99 możemy deklarować zmienną w parametrach pętli* ***for***.

- **Odwoływanie się** do elementów tablicy

```
int sum = 0;
for (int i=0; i<SIZE; i++)
    sum += Numbers[i];

printf("sum = %i\n", sum);
```

- **Wypisywanie** elementów tablicy

```
for (int i=0; i<SIZE; i++)
    printf("Numbers[%i] = %i \n", i, Numbers[i]);
```

Tablice - zadanie

Zadanie
Napisz program, który wczytuje 5 liczb i wypisuje je w odwrotnej kolejności.

Rozwiązanie "nieeleganckie"

```c
int a, b, c, d, e;
printf("Podaj piec liczb calkowitych: ");
scanf("%i %i %i %i %i", &a, &b, &c, &d, &e);
printf("%i %i %i %i %i", e, d, c, b, a);
```

Rozwiązanie korzystające z pętli i tablic

Kod programu: tablica-liczb.c

```c
#include <stdio.h>
#define SIZE 5
int main()
{
    int Numbers[SIZE];
    // TO_DO: dokonczyc ...
}
```

Pytanie z przyszłości: *czy do tego problemu koniecznie trzeba użyć tablic?*

Tablice – uwaga na błędy!

Uwaga na błędy!

- Nie wychodzimy poza zakres tablicy

```
1   int tablica_liczb[20];
2
3   tablica_liczb[20] = 4; // blad
4   tablica_liczb[-5] = 0; // blad
```

- Odwołujemy się do elementów tablicy za pomocą indeksów – liczb całkowitych

```
1   int tablica_liczb[10];
2   float a = 1.0;
3   ...
4   tablica_liczb[a] = 5;   // blad
```

- Rozmiar tablicy jest ograniczony i może być określony za pomocą zmiennej
  Od wersji C99 i o ile nie jest to zmienna globalna, ale o tym w przyszłości...

```
1   int rozmiar = 1000000;
2   int tablica_liczb[rozmiar]; // za duzo?
3   ...
```

4. Lista inicjalizacyjna

Przykład

```
int tabA[5] = { 5, 4, 3, 2, 1 };
int tabB[10] = { 1, 2, 3 }; // pozostale elementy tablicy to zera
```

Uwaga! *Elementy tablicy, które nie zostały ustalone w sposób jawny mają wartość zerową.*

- Wygodny sposób na zadeklarowanie tablicy z samymi zerami

```
int tabD[100] = { 0 };
```

- Korzystając z listy inicjalizacyjnej nie musimy podawać rozmiaru tablicy

```
float tabE[] = { 2.4f, 1.4f, 0.0f };
```

- Kolejność elementów listy możemy ustalić za pomocą desygnatora

```
int tabF[5] = { [0]=2, [4]=3, [1]=7,6,5 };
int tabG[] = { [0]=2, [20]=9, [10]=8,5,4 };
```

5. Tablice znaków i łańcuchy znaków

Łańcuch znaków to tablica znaków (zmiennych typu `char`), w której występuje co najmniej jeden znak `'\0'` reprezentujący koniec łańcucha.

```
1   char napis[20];
2   char slowo[6] = { 'n', 'a', 'p', 'i', 's', '\0' };
3   char imie[] = "Json";
```

Przykład

```
1   // Kod programu: string.c
2   char imie[30];
3
4   printf("Podaj swoje imie: ");
5   scanf("%s", imie);    // zwroc uwage na brak znaku &
6
7   printf("Pierwsza litera imienia %s to %c \n", imie, imie[0]);
8
9   int i=0;
10  while (imie[i] != '\0')
11     printf("%c, ", imie[i++]);
```

Tablicami znaków zajmiemy się dokładniej na kolejnych wykładach.

6. Tablice wielowymiarowe

Deklaracja zmiennej tablicowej wielowymiarowej:

```
TYP nazwa[X][Y][Z];
```

Elementami tablicy są `nazwa[i][j][k]`, gdzie $0 \leqslant i < X$, $0 \leqslant j < Y$, $0 \leqslant k < Z$.

Przykład

```
1  // Kod programu: array2D.c
2  int tabliczka_mnozenia[10][10];
3
4  for (int i=0; i<10; i++)
5     for (int j=0; j<10; j++)
6        tabliczka_mnozenia[i][j] = i*j;
7
8  // wypisywanie
9  for (int i=0; i<10; i++)
10 {
11    for (int j=0; j<10; j++)
12       printf("%2i ", tabliczka_mnozenia[i][j]);
13    printf("\n");
14 }
```

Lista inicjalizująca dla tablicy wielowymiarowej

Kod programu: array-2D-int.c

```
1  #include <stdio.h>
2  int main()
3  {
4      int numbers[4][10] = {
5          {0, 1, 2, 3, 4, 5, 6, 7, 8, 9 },
6          {0, 2, 4, 6, 8, 10, 12, 14, 16, 18 },
7          {0, 3, 6, 9, 12, 15, 18, 21, 24, 27 },
8          {0, 4, 8, 12, 16, 20, 24, 28, 32, 36 }
9      };
10
11     for (int i=0; i<4; i++)
12     {
13         for (int j=0; j<10; j++)
14             printf("%2i ", numbers[i][j]);
15         printf("\n");
16     }
17
18     return 0;
19 }
```

Uwaga! *Deklarując listą inicjalizującą tablicę wielowymiarową można pominąć tylko rozmiar na pierwszej współrzędnej.*

Dwuwymiarowe tablice znaków

Kod programu: array-string.c

```c
#include <stdio.h>
int main()
{
    char tydzien[7][13] =
    {
        "poniedzialek",
        "wtorek",
        "sroda",
        "czwartek",
        "piatek",
        "sobota",
        "niedziela",
    };

    for (int i=0; i<7; i++)
        printf("%i dzien tygodnia to %s \n", i+1, tydzien[i]);

    return 0;
}
```

Pytanie z przyszłości: w jaki sposób przechowywane są tablice wielowymiarowe w pamięci?

Pytania?

Dodatkowe slajdy

Lista inicjalizacyjna i tablice VLA (variable length array)

Uwaga! *Możemy używać listy inicjalizacyjnej tylko dla tablic, których rozmiar jest znany na etapie kompilacji.*

```
1   int n = ...;
2   ...
3   int tabH[n] = { 1, 2, 3 }; // blad!
```

Określenie rozmiaru tablicy

```
1   int numbers[] = { 1, 4, 5, 1, 2, 3, 6, 8, 34, 2, 1, 5, 0, -23, 1, 4, 2, 5 };
2
3   // okreslenie liczby elementow tablicy
4   int size = sizeof(numbers) / sizeof(int);
```

Uwaga! *Operatora* ***sizeof*** *można użyć do określenia liczby elementów tablicy tylko tam, gdzie widoczna jest deklaracja zmiennej tablicowej. Do problemu wrócimy przy okazji wskaźników.*

Operacje bitowe

Operacje bitowe

Operacje wykonywane jedynie na wartościach całkowitoliczbowych (`char`, `int` itp.)

Lista operatorów bitowych

operator	opis
`&`	bitowa operacja `AND` (i)
`|`	bitowa operacja `OR` (lub)
`^`	bitowa operacja `XOR` (albo)
`~`	bitowa negacja (dopełnienie do jedności)
`>>`	przesunięcie bitowe w prawo
`<<`	przesunięcie bitowe w lewo

Przykład

```
1  unsigned char a = 18, b = 7, c = 0;
2  c = a|b;
3  c = a&b;
4  c = a^b;
5  c = ~a;
6  c = a>>2;
7  c = b<<3;
8  c = 0b11001;  // rozszerzenie kompilatora GCC
```

Operacje bitowe

Przykład programu wypisującego kolejne potęgi liczby 2.

```
1   unsigned int potega;
2   for (int i=0; i<10; i++)
3   {
4       potega = 1 << i;
5       printf("2^%i = %i \n", i, potega);
6   }
```

Przykład programu wypisującego postać binarną liczby dziesiętnej

```
1   int liczba;
2   scanf("%i", &liczba);
3
4   char bit0, bit1, bit2, ...;
5
6   // obliczanie kolejnych bitów
7   if (liczba & 1) bit0 = 1; else bit0 = 0;
8   if (liczba & 2) bit1 = 1; else bit1 = 0;
9   if (liczba & 4) bit2 = 1; else bit2 = 0;
10  ...
```

Pytanie: *Jak można poprawić ten program?*

Pytania?