Programowanie strukturalne - wykład 3

Język C

Instrukcje sterujące

Instrukcje warunkowe

```
if (wyrażenie) {
  /* blok wykonany, jeśli wyrażenie jest prawdziwe */
}
```

```
 if (wyrażenie) {
  /* blok wykonany, jeśli wyrażenie jest prawdziwe */
} else {
  /* blok wykonany, jeśli wyrażenie jest nieprawdziwe */
}
```

```
switch (wyrażenie) {
case wartość1:
 break;
case wartość2:
 break;
/* ... */
default:
 break;
}
```

Przykłady:

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int a =5;
    if (a>0)
    {
        printf("Liczba dodania\n");
    }
    else
    {
        printf("Liczba nie jest dodania\n");
    }
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int a =-1;
    if (a)
    {
        printf("Prawda\n");
    }
    else
    {
        printf("Fałsz\n");
    }
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int a =-1;
    if (a)
        printf("a");
    printf("bc");
    return 0;
}
```

Uwaga: porównywanie floatów nie za pomocą ==

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    float a=1.1;
    float b=3.3;
    printf("%d",b==3*a);
    return 0;
}
```

Pętle

for

```
for (wyrażenie1; wyrażenie2; wyrażenie3) {
   /* instrukcje do wykonania w pętli */
 }
```

do ...while

```
do {
  /* instrukcje do wykonania w pętli */
} while (warunek);
```

while

```
while (warunek) {
    /* instrukcje do wykonania w pętli */
 }
```

Przykłady pętli

Warunek początkowy może lub nie musi mieć deklarację zmiennej interującej (od C99).

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    for(int i=1;i<5;i++)
    {
        printf("%d\n",i);
    }
    return 0;
}
```

Przed C99

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int i;
    for(i=1;i<5;i++)
    {
        printf("%d\n",i);
    }
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    for(int i=0;i<5;i++)
    {
        printf("%d\n",i);
    }
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    for(int i=5;i>0;i--)
    {
        printf("%d\n",i);
    }
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    for(int i=5;i>0;i--)
    {
        printf("%d\n",i);
    }
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    for(int i=1;i*i<100;i+=2)
    {
        printf("%d\n",i);
    }
    return 0;
}
```

Nieskończona pętla

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    for(;;)
    {
        printf("endless loop!");
    }
    return 0;
}
```

Wnętrze jest opcjonalne:

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
    for(int n = 0; n < 10; ++n, printf("%d\n", n))
    ; // null statement
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int i =0;
    while (i<5)
    {
        printf("%d\n",i);
        i++;
    }
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int i =0;
    while (i<5)
    {
        printf("%d\n",i);
        if (i>2)
        {
            break;
        }
        i++;
    }
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int i =0;
    while (i<5)
    {
        i++;
        if (i==2)
        {
            continue;
        }
        printf("%d\n",i);
    }
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int i =0;
    do
    {
        printf("%d\n",i);
        i++;
    }
    while (i<-2);
    return 0;
}
```

Funkcja

Funkcje w C

Ogólna składnia funkcji

```
typ identyfikator (typ1 argument1, typ2 argument2)
{
  /* instrukcje */
}
```

procedura - podprogram nie zwracający wyniku, realizujący pewną funkcjonalność

```
void identyfikator (typ1 argument1, typ2 argument2)
{
  /* instrukcje */
}
```

Instrukcja return powoduje zakończenie wykonywania funkcji i zwrócenie wartości. Może być ona użyta dowolną ilość razy w kodzie funkcji.

Funkcja - podprogram zwracający wynik na podstawie przekazanych argumentów

```
typ identyfikator (typ1 argument1, typ2 argument2)
{
  /* instrukcje */
  return ...;
}
```

Funkcja czysta - funkcja, która nie ma żadnych skutków ubocznych, tzn. nie modyfikuje ani przekazanych argumentów, ani globalnego stanu programu.

Przykłady:

Funkcja bez argumentu zwracająca `int`:

```
int foo1()
{
    return 62;
}
```

Funkcja z jednym argumentem zwracająca `int`

```
int foo2(int a)
{
    return a+4;
}
```

Funkcja z dwoma argumentami zwracająca `float`

```
float foo3(int a, int b)
{
    return (b+a)/3.0;
}
```

Procedura bez argumentu:

```c
void foo4()
{
    printf("abc");
}
```

Procedura z jednym argumentem

```c
void foo5(int a)
{
    printf("%d",a*8);
}
```

Wszystkie wcześniejsze funkcje były “czyste”.

Czy ta funkcja jest czysta?

```c
#include <stdio.h>
#include <stdlib.h>

void foo10(int a)
{
    a+=5;
    printf("a%d\n",a);
}

int main()
{
    int a = 7;
    printf("b%d\n",a);
    foo10(a);
    printf("c%d\n",a);
    return 0;
}
```

Rekurencja

Rekurencja, zwana także rekursją (ang. recursion, z łac. recurrere, przybiec z powrotem) – odwoływanie się np. funkcji lub definicji do samej siebie.

Przykłady:

- silnia

$$0! = 1, \quad n! = (n-1)! \cdot n$$

- ciąg Fibonacciego

$$F_n := \begin{cases} 0 & \text{dla } n = 0, \\ 1 & \text{dla } n = 1, \\ F_{n-1} + F_{n-2} & \text{dla } n > 1. \end{cases}$$

```c
#include <stdio.h>
#include <stdlib.h>

int silnia(int n)
{
    if (n ==0)
    {
        return 1;
    }
    return n* silnia(n-1);
}

int main()
{
    int a = 5;
    printf("%d\n",silnia(a));
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>

int fib(int n)
{
    if (n ==0)
        return 0;
    if (n == 1)
        return 1;
    return fib(n-1)+fib(n-2);
}

int main()
{
    int a = 6;
    printf("%d\n",fib(a));
    return 0;
}
```

Bibliografia

- https://pl.wikibooks.org/wiki/C/Funkcje, dostęp online 10.03.2020.
- https://pl.wikipedia.org/wiki/Rekurencja, dostęp online 10.03.2020.
- https://pl.wikibooks.org/wiki/C/Instrukcje_steruj%C4%85ce, dostęp online 10.03.2020.
- Richard Reese, Wskaźniki w języku C, Wydawnictwo Helion 2014.
- https://pl.wikibooks.org/wiki/C/Wska%C5%BAniki, dostęp online 15.03.2020.
- http://wazniak.mimuw.edu.pl/index.php?title=Wst%C4%99p_do_programowania_w_j%C4%99zyku_C/Wska%C5%BAniki, dostęp online 15.03.2020.
- https://pl.wikibooks.org/wiki/C/Wska%C5%BAniki_-_wi%C4%99cej, dostęp online 15.03.2020.