07

Języki programowania 1

Pętle programowe

Zakres:

- while (dopóki),
- do-while (wykonuj-dopóki),
- for (dla).

Pętla typu while

Słowo kluczowe używane jest w C do przetwarzania ujętego w nawiasy klamrowe bloku instrukcji (ciała funkcji) dopóki spełniony jest warunek wykonania pętli podany po słowie kluczowym while. Ogólna postać i budowa tej pętli może przyjąć postać:

```
while (wyrażenie_warunkowe)
{
 blok instrukcji;
}
```

Pętla działa w ten sposób, że najpierw sprawdzane jest wyrażenie warunkowe. Jeżeli jest ono niezerowe, to wykonywanie pętli jest kontynuowane (wykonywane są poszczególne instrukcje). Po wykonaniu wszystkich instrukcji wyrażenie jest sprawdzane ponownie. Jeżeli znów zwróci wartość niezerową, to pętla zostaje powtórzona. Dzieje się tak do momentu, aż kolejne sprawdzenie wyrażenia zwróci wartość zero.

Przykład pętli while, program *while.c*.
Pętla while może trwać nieskończenie długo jeżeli wpiszemy warunek, który będzie zawsze spełniony. Przykładowo:

```
while(1)
{
 instrukcja_1;
 instrukcja_2;
 .....
}
```

Wówczas wyjściem z pętli może być jakiś warunek występujący wewnątrz pętli.

Pętla do-while

W pętli do-while wyrażenie warunkowe znajduje się na końcu pętli (wykonuj-dopóki). Dzięki temu instrukcje będące wewnątrz tej pętli zostaną wykonane zawsze co najmniej jeden raz. Uogólniona postać pętli do-while:

```
do
{
 instrukcja_1;
 instrukcja_2;
 ...
}while(wyrażenie_warunkowe);
```

Jeżeli wyrażenie warunkowe zwraca wartość niezerową, wówczas pętla zostaje powtórzona. W przeciwnym wypadku program wykonywany jest dalej. W pętli tej zwrócić należy uwagę na to, że zakończona ona jest średnikiem.

Przykład pętli do-while, program *do_while.c*.

Pętle programowe z zastosowaniem instrukcji for

Ogólny format instrukcji for jest następujący:

```
for(wyrażenie_1; wyrażenie_2; wyrażenie_3)
{
 instrukcja_1;
 instrukcja_2;
 ...
}
```

Instrukcja for wykorzystuje 3 wyrażenia rozdzielone średnikami. W wyrażeniu_1 następuje zainicjowanie jednej, bądź wielu zmiennych. Ważnym jest to, że wyrażenie_1 wykonywane jest tylko jeden raz, na samym początku pętli. Wyrażenie_1 nazywane jest wyrażeniem inicjującym pętlę.
Wyrażenie_2 jest wyrażeniem warunkowym. Wyrażenie to sprawdzane jest zaraz po wykonaniu wyrażenia_1, a następnie po każdorazowym, udanym zakończeniu pętli. Jeżeli wyrażenie_2 zwraca wartość różną od zera, to pętla jest wykonywana. Jeżeli zwróci wartość 0, to wówczas pętla jest przerywana i wykonanie instrukcji for zostaje zakończone.
Wyrażenie_3 jest tzw. wyrażeniem_skoku. Nie jest one wykonywane w pierwszym wejściu do pętli, a dopiero po udanym przejściu pętli, przed przeprowadzeniem analizy wyrażenia_2.
Przykład instrukcji for, program *for.c*.

Pusta instrukcja for

Zazwyczaj instrukcja for nie kończy się średnikiem. Cóż zatem się stanie jeżeli wpiszemy:

for(i=1;1<10;i++);

lub

for(i=1;i<10;i++)
;

Pętla ta nie będzie wykonywała niczego, prócz samego zapętlania. Jest to tzw. pusta pętla.

Instrukcja for bardziej rozbudowana

Język C umożliwia w instrukcji for w miejscach nagłówków wpisać więcej wyrażeń. Muszą one być tylko oddzielone przecinkami. Przykładowo:

```
for(i=0,j=10;i<10,j>0;i++,j--)
{
 blok_instrukcji;
}
```

Przykład *for1.c*.