Programowanie strukturalne - wykład 2

Język C

Sposób działania C

1. Kod źródłowy
2. Kompilacja
3. Kod wynikowy

Hello World

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

int main()
{
    printf("Hello world!\n");
    return 0;
}
```

Sprawdzenie wersji języka C

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

int main()
{
    printf("%d",__STDC_VERSION__);
    return 0;
}
```

Wbudowane typy danych

https://en.wikipedia.org/wiki/C_data_types

https://devdocs.io/c/language/arithmetic_types

Deklaracja zmiennej

```
typ nazwa_zmiennej;
```

Przykład:

```
int a;
```

Przykład z nadaniem wartości:

```
int a=7;
```

Instrukcje wyjścia/wejścia

- Szczegółowo będą później

`print` - wyjście

https://en.cppreference.com/w/c/io/fprintf

`scanf` - wejście

https://en.cppreference.com/w/c/io/fscanf

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

int main()
{
    int a =0;
    scanf("%d",&a);
    printf("%d",a);
    return 0;
}
```

https://cpp0x.pl/dokumentacja/tekst-sformatowany-printf/736

Różnica między %d a %i?

Dotyczy tylko `scanf` przy `%i` mamy warianty:

- `12` - system dziesiętny
- `012` - system ósemkowy
- `0x12` - system szesnastkowy

Prezentacja debuggera

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

int main()
{
    int a = 5;
    a= -2;
    a=7;
    return 0;
}
```

Zasięg zmiennej

- globalne - zadeklarowane poza `main`, dostępne dla wszystkich funkcji.
- lokalne - zadeklarowane w innym miejscu

```c
#include <stdio.h>

int a,b;

void func1 ()
{
    a=3;
}

int main ()
{
    b=3;
    a=2;
    return 0;
}
```

```c
int a=1; /* zmienna globalna */

int main()
{
    int a=2;          /* to już zmienna lokalna */
    printf("%d", a); /* wypisze 2 */
}
```

Typy zmiennych

https://en.wikipedia.org/wiki/C_data_types

https://devdocs.io/c/language/arithmetic_types

Zasięg zmiennej

- globalne - zadeklarowane poza `main`, dostępne dla wszystkich funkcji.
- lokalne - zadeklarowane w innym miejscu

```c
#include <stdio.h>

int a,b;

void func1 ()
{
    a=3;
}

int main ()
{
    b=3;
    a=2;
    return 0;
}
```

```c
int a=1; /* zmienna globalna */

int main()
{
    int a=2;          /* to już zmienna lokalna */
    printf("%d", a); /* wypisze 2 */
}
```

Czas życia

Czas życia to czas od momentu przydzielenia dla zmiennej miejsca w pamięci (stworzenie obiektu) do momentu zwolnienia miejsca w pamięci (likwidacja obiektu).

```
int main()
{
 int a = 10;
 {          /* otwarcie lokalnego bloku */
     int b = 10;
     printf("%d %d", a, b);
 }          /* zamknięcie lokalnego bloku */

    printf("%d %d", a, b);  /* BŁĄD: b juz nie istnieje */
}
```

Stałe

Stała, różni się od zmiennej tylko tym, że nie można jej przypisać innej wartości w trakcie działania programu. Wartość stałej ustala się w kodzie programu i nigdy ona nie ulega zmianie.

```
const typ nazwa_stałej=wartość;
```

#define

`#define` - dyrektywa preprocessora.

Linia pozwalająca zdefiniować stałą, funkcję, słowo kluczowe lub makro, które będzie potem podmienione w kodzie programu na odpowiednią wartość lub może zostać użyte w instrukcjach warunkowych dla preprocesora.

```
#define NAZWA_STALEJ WARTOSC
```

```
#define NAZWA_STALEJ
```

```c
#include <stdio.h>

#define SIX 1+5
#define NINE 8+1

int main(void)
{
    printf("%d * %d = %d\n", SIX, NINE, SIX * NINE);
    return 0;

}
```

Operator przypisania

Operator przypisania (“=”), jak sama nazwa wskazuje, przypisuje wartość prawego argumentu lewemu.

```
int a = 3, b;
b = a;
printf("%d\n", b); /* wypisze 3 */
```

C umożliwia też skrócony zapis postaci a #= b;, gdzie # jest jednym z operatorów: +, -, *, /, %, &, |, ^, « lub ». Ogólnie rzecz ujmując zapis a #= b; jest równoważny zapisowi a = a # (b).

```
int a = 1;
a += 5;      /* to samo, co a = a + 5;       */
a /= a + 2; /* to samo, co a = a / (a + 2); */
a %= 2;      /* to samo, co a = a % 2;       */
```

Operatory arytmetyczne dwuargumentowe

- dodawanie (“+”),
- odejmowanie (“-”),
- mnożenie (“*“),
- dzielenie (“/”),
- reszta z dzielenia (“%”) określona tylko dla liczb całkowitych (tzw. dzielenie modulo).

Jednoargumentowe operatory arytmetyczne

- pre-inkrementacja (“++i”),
- post-inkrementacja (“i++”),
- pre-dekrementacja (“–i”),
- post-dekrementacja (“i–”).

```
int a, b, c;
a = 3;
b = a--; /* po operacji b=3 a=2 */
c = --b; /* po operacji b=2 c=2 */
```

Operator rozmiaru `sizeof()`

Zwraca rozmiar obiektu podany w wybranej jednostce miary, np. bajtach lub słowach maszynowych.

```
printf ("%d",sizeof(int));
```

```
4
```

Operatory bitowe

- negacja bitowa (NOT)(“~”),
- koniunkcja bitowa (AND)(“&”),
- alternatywa bitowa (OR)(“|”) i
- alternatywa rozłączna (XOR) (“^”)

Są zdefiniowane dla liczb całkowitych, działają na bitach i mogą być szybsze niż zwykłe operacje.

https://pl.wikibooks.org/wiki/C/Operatory#Operatory_bitowe

Operatory porównania

- równe (“==”),
- różne (“!=”),
- mniejsze (“<”),
- większe (“>”),
- mniejsze lub równe (“<=”)
- większe lub równe (“>=”)

Operatory logiczne

- negacja (zaprzeczenie): “!”
- koniunkcja (“i”): “&&”
- alternatywa (“lub”): “||”

Operator wyrażenia warunkowego

```
a ? b : c
```

Najpierw oceniana jest wartość logiczna wyrażenia `a`; jeśli jest ono prawdziwe, to zwracana jest wartość `b`, jeśli natomiast wyrażenie `a` jest nieprawdziwe, zwracana jest wartość `c`.

Kolejność operatorów

https://en.cppreference.com/w/c/language/operator_precedence

Debugowanie

Sesja - live coding.

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;
}
```

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 */
 }
```

Bibliografia

- http://www.aistudy.com/program/images/programming_language_family_tree.gif, dostęp online 20.02.2020.
- Wojciech Sobieski, Języki Programowania, http://pracownicy.uwm.edu.pl/wojsob/pliki/publikacje/jp-01.pdf, dostęp online 20.02.2020.
- https://pl.wikipedia.org/wiki/Asembler, dostęp online 20.02.2020.
- https://pl.wikipedia.org/wiki/J%C4%99zyk_wysokiego_poziomu, dostęp pnline 20.02.2020.
- http://wazniak.mimuw.edu.pl/index.php?title=Paradygmaty_programowania/Wyk%C5%82ad_1:_Co_to_jest_paradygmat_programowania%3F, dostęp online: 20.02.2020.
- https://mfiles.pl/pl/index.php/Programowanie_strukturalne, dostęp online 20.02.2020.

- https://pl.wikipedia.org/wiki/C_(j%C4%99zyk_programowania), dostęp online 20.02.2020.
- https://pl.wikibooks.org/wiki/C/Zmienne#Deklaracja_zmiennych, dostęp online 20.02.2020.
- https://pl.wikibooks.org/wiki/C/Operatory, dostęp online 20.02.2020.
- https://pl.wikibooks.org/wiki/C/Operatory#Operatory_bitowe, dostęp online 10.03.2020.
- 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.