

Wstęp do programowania w języku C

Marek Piotrów - Wykład 2
Standardowe typy liczbowe. Filtry.

12 października 2016

Jednostki leksykalne języka C

- Słowa kluczowe: `int`, `float`, `if`, `else` `while`, `do`, `return`, itd.
- Identyfikatory: `xx`, `a1`, `Tab`, `_napocz`, itd.
- Stałe: znakowe: `'A'`, całkowite: `12`, `-5`, `0xBF`, zmiennopozycyjne `1.0`, `-0.21`, `1e10` lub napisowe `"Wlazl"` .
- Operatory i separatory: `+`, `*`, `(`, `)`, `->`, `<=`, `;`
- Komentarze: wielowierszowe `/* ... */` lub jednowierszowe `// ...`

Pliki nagłówkowe i funkcje

- Pliki nagłówkowe definiują interfejsy do bibliotek funkcji w języku C.
- `#include <stdio.h>` pozwala na korzystanie ze standardowych funkcji wejścia/wyjścia.
- `#include <stdlib.h>` pozwala na korzystanie z wielu użytecznych funkcji z biblioteki standardowej.
- Definicje funkcji opisują podstawowe moduły funkcjonalne programu. Funkcje powinny być sparametryzowane i wykonywać jedno dobrze zdefiniowane zadanie.
- Każdy program musi zawierać funkcję `main`.

- Deklaracje nazywają obiekty, których używamy w programie i podają ich typy. Mogą przypisywać obiektom wartości początkowe.
- Instrukcje opisują sekwencję operacji, które będą wykonywane na obiektach.
- Podstawowe instrukcje to przypisanie zmiennej wartości podanego wyrażenia, instrukcje pętli i instrukcje warunkowe. Instrukcją jest też wywołanie funkcji.

Podstawowe typy standardowe w języku C

Typy stałopozycyjne

- **znakowe:** `char`, `unsigned char`, `signed char`;
- **całkowite** `int`, `unsigned int`;
- **długie całkowite:** `long int`, `long`, `unsigned long int`;
- **bardzo długie całkowite:** `long long int`, `unsigned long long int`;

Typy zmiennopozycyjne

- **podstawowy:** `float`;
- **podwójnej dokładności:** `double`;
- **maksymalnej dokładności:** `long double`.

Przykład 1 - kopiowanie wejścia na wyjście

ZADANIE: Napisać program, który skopiuje wszystkie znaki ze standardowego wejścia na standardowe wyjście.

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

```
/* Bezposrednie kopiowanie standardowego wejścia na wyjście */
```

```
int main(void)
```

```
{  
    int c;
```

```
    c=getchar();
```

```
    while (c != EOF) {
```

```
        putchar(c);
```

```
        c=getchar();
```

```
    }
```

```
    return 0;
```

```
}
```

Przykład 1 - kopiowanie wejścia na wyjście

ZADANIE: Napisać program, który skopiuje wszystkie znaki ze standardowego wejścia na standardowe wyjście.

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

```
/* Bezpośrednie kopiowanie standardowego wejścia na wyjście */
```

```
int main(void)
```

```
{
```

```
    int c;
```

```
    c=getchar();
```

```
    while (c != EOF) {
```

```
        putchar(c);
```

```
        c=getchar();
```

```
    }
```

```
    return 0;
```

```
}
```

Przykład 1a - kopiowanie z użyciem scanf/printf

```
#include <stdio.h>

/* Bezpośrednie kopiowanie standardowego wejścia na wyjście */

int main(void)
{
    int wynik;
    char c;

    wynik=scanf("%c",&c);
    while (wynik == 1) {
        printf("%c",c); // Użycie scanf/printf zamiast getchar/putchar
        wynik=scanf("%c",&c); // powoduje około 4-krotne wolniejsze kopiowanie
    }
    return 0;
}
```


Przykład 2 - kopiowanie z zamianą dużych liter na małe

ZADANIE: Napisać program, który skopiuje wszystkie znaki ze standardowego wejścia na standardowe wyjście zamieniając wszystkie duże litery na małe.

```
#include <stdio.h>

/* Przepisywanie wejścia na wyjście z zamiana dużych liter na małe */

int main(void)
{
    int c;

    while ((c=getchar()) != EOF) {
        if (c >= 'A' && c <= 'Z')
            c = c - 'A' + 'a';
        putchar(c);
    }
    return 0;
}
```

Przykład 2 - kopiowanie z zamianą dużych liter na małe

ZADANIE: Napisać program, który skopiuje wszystkie znaki ze standardowego wejścia na standardowe wyjście zamieniając wszystkie duże litery na małe.

```
#include <stdio.h>

/* Przepisywanie wejścia na wyjście z zamiana dużych liter na małe */

int main(void)
{
    int c;

    while ((c=getchar()) != EOF) {
        if (c >= 'A' && c <= 'Z')
            c=c-'A'+ 'a';
        putchar(c);
    }
    return 0;
}
```

Przykład 3 - kodowanie bajtów metodą Base64

- Kodowanie Base64 służy do jednoznacznego reprezentowania dowolnego ciągu bajtów za pomocą tekstu składającego się z 64 (plus 1 - wypełniacz (=)) widocznych znaków. Te znaki to 26 liter dużych (A-Z), 26 liter małych (a-z), 10 cyfr (0-9) oraz znaków plus (+) i dzielone (/). Kodowanie przypisuje tym znakom kolejno wartości $0, 1, \dots, 63 = 2^6 - 1$. Każdy ciąg sześciu bitów jest reprezentowany jednoznacznie przez jedną z tych wartości.
- Wejściowy ciąg bajtów dzielony jest na grupy po 3 bajty. Każda grupa składa się z 24 bitów, czyli może być reprezentowana jako 4 sekwencje 6-bitowe i zakodowana jako grupa 4 znaków z podanego powyżej zbioru. Jeśli długość ciągu wejściowego nie jest podzielna przez 3, ostatnią sekwencję jednego lub dwóch bajtów koduje się uzupełniając ciąg bitów zerami z prawej strony do długości podzielnej przez 6, a otrzymany ciąg znaków kodujących uzupełnia się do 4 wypełniaczem.

Przykład 3 - kodowanie bajtów metodą Base64

Kodowanie Base64 jest używane m.in. do:

- w poczcie elektronicznej do przesyłania załączników binarnych;
- kodowania haseł w protokole SMTP podczas uwierzytelniania metodami PLAIN i LOGIN.

ZADANIE: Napisać program, który zakoduje wszystkie bajty ze standardowego wejścia na standardowym wyjściu używając kodowania Base 64. W domu możecie spróbować zaprogramować odkodowywanie Base64.

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

```
/******
```

```
* Kodowanie ciągu znaków ze standardowego wejścia w kodzie Base64  
* i wypisywanie zakodowanego tekstu na standardowym wyjściu (po 76  
* znaków w wierszu). Przykład z wykładu z dnia 13.10.2015 (MPI).  
* Zadanie dla chętnych: Napisać program odkodujący dla Base64.  
*****/
```

```
#define WYPELNIACZ ' '
```

```
#define KOLUMN 76 // liczba musi być podzielna przez 4
```

```
int main(void)
```

```
{  
    char kod[64]="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";  
    int z1, z2, z3, w_wierszu=KOLUMN, kodow;  
    int bitow6[4];  
    unsigned long int buf64;
```

```
do {  
    if ((z1=getchar()) == EOF) break;  
    else if ((z2=getchar()) == EOF) { // z1 - ostatni znak  
        kodow=2;  
        buf64=z1*16;  
    }  
    else if ((z3=getchar()) == EOF) { //z1,z2 - ostatnie 2 znaki  
        kodow=3;  
        buf64=(z1*256+z2)*4;  
    }  
    else {  
        kodow=4;  
        buf64=(z1*256+z2)*256+z3; // powinno się użyć przesuniecie  
    }  
}
```

```

for (int i=kodow-1; i >= 0; i--) {
    bitow6[i]=buf64%64;
    buf64=buf64/64;
}
for (int i=0; i < kodow; i++)
    putchar(kod[bitow6[i]]);
for (int i=kodow; i < 4; i++)
    putchar(WYPELNIACZ);
w_wierszu=w_wierszu-4;
if (w_wierszu == 0) {
    putchar('\n');
    w_wierszu=KOLUMN;
}
}
while (kodow == 4);
if (w_wierszu != KOLUMN) putchar('\n');
return 0;
}

```

Przykład 4 - zliczanie słów i wierszy w tekście

```
#include <stdio.h>

/* Zliczanie znakow, slow i wierszy w tekście wejściowym */

#define IN 1 /* wewnątrz słowa */
#define OUT 0 /* poza słowem */

int main(void)
{
    int z,lw,stan;
    long int ls,lz;

    stan=OUT; lw=0; ls=lz=0L;

    while ((z=getchar()) != EOF) {
        ++lz;
        if (z == '\n') ++lw;
        if (z == ' ' || z == '\n' || z == '\t')
            stan=OUT;
        else if (stan == OUT) {
            stan=IN; ++ls;
        }
    }
    printf("Wierszy: %d, slow: %ld, znakow: %ld\n",lw,ls,lz);
    return 0;
}
```

Przykład 5 - zliczanie znaków w tekście

```
#include <stdio.h>

/* Zliczanie liter, białych znaków i innych */

int main(void)
{
    int z;
    long int biale,inne;
    int litery[26];

    biale=inne=0L;
    for (int i=0; i < 26; ++i) litery[i]=0;

    while ((z=getchar()) != EOF)
        if (z >= 'A' && z <= 'Z')
            ++litery[z-'A'];
        else if (z >= 'a' && z <= 'z')
            ++litery[z-'a'];
        else if (z == ' ' || z == '\n' || z == '\t')
            ++biale;
        else ++inne;

    printf("Liter:");
    for (int i=0; i < 26; ++i)
        if (litery[i] > 0) printf(" %c%c=%d",'a'+i,'A'+i,litery[i]);
    printf("\nbiałych znaków: %ld, innych: %ld\n",biale,inne);
    return 0;
}
```


Przykład 6 - wyszukiwanie wzorca I

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

/* Wypisz wszystkie wiersze zawierające podany wzorzec. Tekst jest zadany
 * na standardowym wejściu, a wzorzec (dowolny ciąg znaków) jest podany jako
 * parametrem wywołania programu. Na standardowe wyjście kopiowane są
 * wiersze zawierające podany wzorzec. */

#define MAKS_DŁUGOSC_WIERSZA 1000

static bool wystapil(char wzor[], char buf[], int pozycja)
{
    int i,j;

    for (i=0, j=pozycja; wzor[i] == buf[j]; ++i, ++j)
        if (wzor[i] == ' \0') return true;
    return (wzor[i]==' \0');
}

static int dlugosc(char tekst[])
{
    int i=0;

    while (tekst[i] != ' \0') ++i;
    return i;
}

int main(int argc, char *argv[])
{
    char bufor[MAKS_DŁUGOSC_WIERSZA];
    int dlugosc_wzorca, dlugosc_wiersza;
```

Przykład 6 - wyszukiwanie wzorca II

```
if (argc <= 1)
{
    printf("Prawidlowe wywołanie programu:\n\t%s wzorzec\n",argv[0]);
    return 1;
}
dlugosc_wzorca=dlugosc(argv[1]);
while (fgets(bufor,sizeof(bufor),stdin) != NULL)
{
    dlugosc_wiersza=dlugosc(bufor);
    for (int i=dlugosc_wiersza-dlugosc_wzorca; i >= 0; --i) // nieoptymalnie
        if (wystapil(argv[1],bufor,i))
        {
            fputs(bufor,stdout);
            break;
        }
}
return 0;
}
```