

Vstup/výstup C-jazyk

10

Práca so súbormi – neobjektovo - orientovaná

Dátový prúd – zdroj->spotrebiteľ – buffer, transformácia dát

Vstupno-výstupná knižnica

Streamy

Práca so súbormi na **najnižšej úrovni** – identifikátor súboru

Binárne (ako v pamäti počítača) – komunikácia medzi počítačmi, programami

Textové súbory – znaková reprezentácia dát – pomalšie

- počítač človek
- xml-značkovacie jazyky (čitateľné, hľadanie chýb)

Otvorenie súboru – textový alebo binárny

Vstupno-výstupná knižnica

- Prostriedky vstupu a výstupu (V/V) nie sú súčasťou syntaxe jazyka C
- Existuje štandardná knižnica vstupno-výstupných funkcií
- Každý program, ktorý chce využívať túto knižnicu, musí obsahovať niekde riadok:
#include <stdio.h> alebo #include <cstdio>
- V knižnici existuje veľké množstvo funkcií
- Väčšinou návratová hodnota -1 znamená chybu
- Globálna premenná errno je po volaní každej funkcie nastavovaná na kód chyby
- Pre použitie premennej errno je nutné vložiť riadok:

#include <errno.h>

printf

- Výstup ide na štandardný výstup (väčšinou obrazovka)

int printf(format, prem1, prem2, ...);

- kde

format:

%[priznak][sirka][.presnost][rozmer]typ

priznak:

- prem. bude zarovnaná vľavo
- + prem. bude zarovnaná vpravo
- # alternatívne vytlačiť (závisí od typu) – 0x, 0X, 0

sirka:

- počet znakov, na koľko bude prem. vytlačená

presnost:

- počet desatinných miest pri výstupe premennej, príp. minimálny počet znakov premennej (typ integer)
- musí začínať bodkou

F|N|h|l|L: - modifikátor typu

- F - ďaleký (far) smerník
- N - blízky (near) smerník
- h - short (int)
- l - long (int, float, double)
- L - long (float, double)

printf - typ

- typ - určuje typ premennej:

<u>zn.</u>	<u>typ</u>	<u>formát výstupu</u>
d	int	znamienkové desiatkové číslo
i	int	znamienkové desiatkové číslo
o	int	bezznam. osmičkové číslo
u	int	bezznam. desiatkové číslo
x	int	bezznam. hexadecimálne číslo
X	int	— " —
f	float	znam. hodnota [-]dddd.dddd
e	float	znam. hodnota [-]d.dddd e [+/-]ddd
g	float	bud' f alebo e (závisí od hodnoty prem.)
E	float	to isté ako e
G	float	to isté ako g
c	char	jeden znak
s	char*	tlačí znaky až kým nenarazí na znak '\0,
p	void*	tlačí premennú ako smerník

printf - typ

```
#include <stdio>

int main()
{
    int p(12);

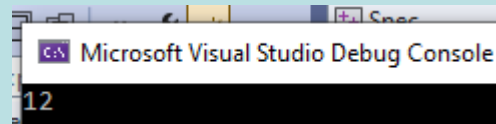
    printf("%d\n",p);
    return 0;
}
```

```
#include <iostream>

using namespace std;

int main()
{
    int p(12);

    cout<< p<<endl;
    return 0;
}
```



printf – príznak, šírka

príznak:

štandardne premenná bude zarovnaná vpravo

- prem. bude zarovnaná vľavo

alternatívny výpis (závisí od typu) – 0x, 0X, 0

šírka:

– počet znakov, na koľko bude premenná. vytlačená

```
#include <stdio>

int main()
{
    int p(12);

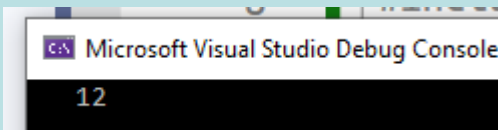
    printf("%5d\n", p);
    return 0;
}
```

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    int p(12);

    cout << setw(5) << right << p << endl;
    return 0;
}
```



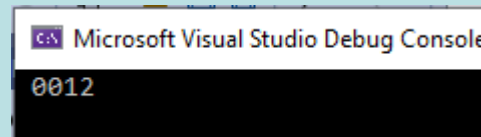
printf – príznak, šírka

presnost:

- počet desatinných miest pri výstupe double premennej, príp. minimálny počet znakov premennej (typ integer)
- musí začínať bodkou

```
int main()
{
    int p(12);

    printf("%5.4d\n",p);
    return 0;
}
```

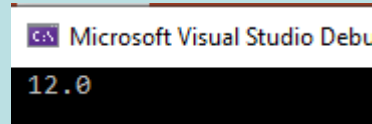


Microsoft Visual Studio Debug Console

0012

```
int main()
{
    double p(12);

    printf("%5.1f\n",p);
    return 0;
}
```



Microsoft Visual Studio Debu

12.0

printf

rozmer – kvôli kompatibilite typov

```
int main()
{
    int p(12);

    printf("%5.1I32d\n", p);
    return 0;
}
```

scanf

int scanf(format, &prem1, &prem2, ...);

- kde

format:

%[sirka][h||L]typ

sirka

maximálna veľkosť pri čítaní zo vstupu

F|N|h||L:

– modifikátor typu

h - short

l - double

L - long double

typ

– podobný ako pri printf

& bude pred každou premennou okrem poľa resp. smerníkom do poľa, ktoré chceme načítať

scanf

& bude pred každou premennou okrem poľa resp. smerníkom do poľa, ktoré chceme načítať

```
int main()
{
    int premenna;
    int* smernik = new int;
    int res = scanf("%d%d", &premenna, smernik);
    delete smernik;
    return 0;
}
```

Formátová konverzia v pamäti

- **sprintf:**

- podobné ako printf, ale výstup ide do reťazca
- formátová konverzia v pamäti
- formát taký istý ako pri printf

int sprintf(char *buf, formát, prem1, prem2, ...)

- **sscanf:**

- podobné ako scanf, ale vstup ide z reťazca
- formátová konverzia v pamäti
- formát taký istý ako pri scanf

int sscanf(char *buf, formát, &prem1, &prem2, ...)

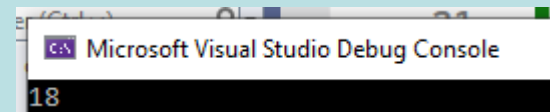
Neobjektové prúdy

- Práca so súbormi na vyššej úrovni
- Prúd/stream je tok dát (zo súboru alebo vstupného zariadenie, alebo do súboru alebo výstupného zariadenia)
- Keď chceme so súborom pracovať musíme ho otvoriť a na konci zatvoriť
- Otvorenie súboru: `fopen`
- Zatvorenie súboru: `fclose`
- Pre prácu s prúdmi množstvo funkcií f....
`fputc`, `fputs`, `feof`, `fgetc`, `fgets`, `fflush`, `fread`, `fwrite`, ...
- Max. počet súčasne otvorených súborov (`FILES`, `ulimit`)

Neobjektové prúdy

- Pri spustení programu sú otvorené prúdy (streamy)
 - stdin – štandardný vstupný prúd
 - stdout – štandardný výstupný prúd
 - stderr – štandardný chybový prúd

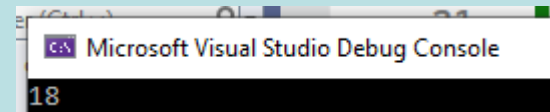
```
int main()
{
    int premenna(18);
    fprintf(stdout, "%d", premenna);
    return 0;
}
```



Neobjektové prúdy

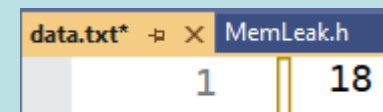
```
void Vypis(FILE* prud, int data)
{
    fprintf(prud, "%d", data);
}

int main()
{
    Vypis(stdout, 18);
    FILE *f = fopen("data.txt", "wt");
    Vypis(f, 18);
    fclose(f);
    return 0;
}
```



Microsoft Visual Studio Debug Console

18



data.txt* MemLeak.h

1 | 18

Max. počet súčasne otvorených súborov 512 (max. 8192)

fopen, fclose

FILE *fopen(char *meno, char *mod)

- funkcia vracia smerník na štruktúru FILE, ak sa súbor dá otvoriť a NULL ak sa súbor nedá otvoriť
- v premennej errno je dôvod, prečo sa nepodarilo otvoriť

meno:

- názov súboru, ktorý chcem otvoriť
- pozor na znak '\\'

mod:

- mód v ktorom má byť súbor otvorený
- r (read) otvorenie súboru iba pre čítanie
- w (write) vytvorenie - " - pre zápis, ak už súbor existuje, bude zrušený
- a (append) otvorenie súboru pre zápis na koniec súboru, ak súbor neexistuje, tak ho vytvorí
- r+ otvorenie súboru pre update (čítanie aj zápis) ak existuje, bude ponechaný
- w+ vytvorenie súboru pre update, ak už súbor existuje, bude zrušený
- a+ otvorenie pre update (ako r+, ale ukazovateľ koniec súboru)

doplňkový mód

- b - otvoriť ako binárny súbor
- t - otvoriť ako textový súbor

int fclose(FILE*)

- zatvorenie súboru
- vracia -1 ak sa nepodarilo zatvoriť
- v premennej errno je dôvod, prečo sa nepodarilo zatvoriť

fprintf, fscanf

- funkcia fprintf - podobná ako printf, ale výstup ide do súboru (stream-u)

int fprintf(FILE *f, format, prem1, prem2, ...)

- funkcia fscanf - podobná ako scanf, ale vstup ide zo súboru (stream-u)

int fscanf(FILE *f, format, &prem1, &prem2, ...)

- Príklad:

```
main()
{
    FILE *f, *g;
    f = fopen( "jano.dat", "w+b" );
    if( f == NULL ) {
        /* chybové hlásenie */
    }
    g = fopen( "jozef.txt", "rt" );
    if( g == NULL ) {
        /* chybové hlásenie */
    }
    ...
    fprintf( f, "Vysledok: %i\n", vysl );
    ...
    fclose(f);
    fclose(g);
}
```

- Vždy sa musí testovať, či sa podarilo otvoriť súbor. Inak je to hrubá chyba.

fprintf(stdout, "ahoj"); je to isté ako **printf("ahoj");**

Čo je koniec súboru

- Koniec súboru nie je znak (aj keď sa niekedy testuje)

```
FILE* f_in = fopen("c:\\text.txt");  
int ch;  
while( (ch=fgetc(f_in)) != EOF ){  
    putchar(ch);  
}
```

```
while(!feof(f)) {  
    putchar(ch);  
}
```

Práca so súbormi na nižšej úrovni

- So súbormi môžeme v jazyku C pracovať na úrovni operačného systému
- Nepristupujeme ku súboru pomocou smerníka na štruktúru FILE, ale pomocou popisovača súboru (handler) - celé číslo typu int
- V zdrojovom súbore programu musí byť:

#include <io.h>

#include <fcntl.h>

- Otvorenie súboru:

int open(char *meno, int mod);

– kde

meno - názov súboru

mod - kombinácia symbolických konštant:

O_RDONLY, O_WRONLY, O_RDWR,

O_NDELAY, O_APPEND, O_CREAT, O_EXCL, O_TRUNC

O_BINARY, O_TEXT,

a ďalších pre zamykanie súborov

- Funkcia vracia popisovač súboru ak sa podarilo súbor otvoriť a -1 ak sa súbor nepodarilo otvoriť (v errno je kód chyby)
- Zatvorenie súboru:
int close(int);
- Pri spustení programu otvorené: vstup(0), výstup(1) a chybový výstup(2)

read, write

- Binárne čítanie zo súboru

int read(int handler, void* buf, unsigned len)

- Binárny zápis do súboru

int write(int handler, void* buf, unsigned len)

– kde

handler

– popisovač súboru

buf

– smerník na buffer, ktorý sa zapíše/prečíta do/zo súboru

len

– počet bytes, ktoré sa zapíšu/prečítajú do/zo súboru

- Zapisuje resp. číta dáta do/zo súboru v tvare v akom sú dáta v pamäti

Príklad

```
#include <io.h>
#include <fcntl.h>
#include <errno.h>

void main()
{
    int in, out;
    out = open( "jano.dat", O_WRONLY | O_CREAT | O_TEXT);
    if( out == -1 ) {
        fprintf(stderr,"Chyba pri otváraní súboru jano.dat: ");
        perror( strerror( errno ) );
        exit( 1 );
    }
    in = open( "jozef.txt", O_RDONLY | O_BINARY);
    if( in == -1 ) {
        fprintf(stderr,"Chyba pri otváraní súboru jozef.txt: ");
        perror( strerror( errno ) );
        close( out );
        exit( 1 );
    }
    char buf[512];
    int num;
    while( ( num = read( in, buf, 512) ) >0 ){
        write( out, buf, num);
    }
    if( num == -1 ) {
        perror( strerror( errno ) );
    }
    close(in);
    close(out);
}
```