

# Informatika 3

## Knižnice

8

# Druhy knižníc

- Štandardné
- Doplnkové
  - doplnené tvorcami prekladačov
  - komerčné
- Funkcionálne
- Objektovo-orientované
- Statické - lib
- Dynamické – potrebná podpora systému (Windows – DLL, Linux - so)

# Štandardné

- Štandardná knižnica – sada štandardov pre knižnice a hlavičkové súbory C++ podľa normy ISO
  - Prevzaté z C jazyka – neobjektovo-orientované, funkcie
  - Objektovo-orientované knižnice C++
  - Hlavičkové súbory

# Prevzaté z C jazyka

- 24 hlavičkových súborov
  - Každý hlavičkový súbor obsahuje jednu či viac deklarácií funkcií a definície typov a makier
  - Poskytuje základnú sadu matematických funkcií, funkcie pre prácu s reťazcami a funkcie pre súborový a konzolový vstup a výstup
  - Jednoduchosť, prenositeľnosť na nové platformy

# Hlavičkové súbory z C jazyka

- **<assert.h> (cassert)** - obsahuje makra pre prácu s chybami - detekcia a odlaďovanie chýb.
- **<complex.h>** - práca s komplexnými číslami
- **<ctype.h> (cctype)** - prevody písmen
- **<errno.h> (cerrno)** - testovanie chybových kódov, hlásených funkciami knižníc
- **<fenv.h>** - práca s plávajúcou desatinnou čiarkou
- **<float.h> (cfloat)** - definície konštánt pre plávajúcu desatinnú čiarku
- **<inttypes.h>** - presné prevody celých čísel
- **<iso646.h>** - programovanie v znakovnej sade ISO 646
- **<limits.h> (climits)** - definícia konštánt a vlastností celočíselných typov
- **<locale.h>** - pre setlocale – výber lokalizácie
- **<math.h> (cmath)** - matematické výpočty
- **<setjmp.h> (csetjmp)** - pre setjmp a longjmp
- **<signal.h> (csignal)** - pre výnimky a signály
- **<stdarg.h> (cstdarg)** - prístup k argumentom funkcií
- **<stdbool.h>** - dátový typ boolean
- **<stdint.h>** - Definície ďalších typov int
- **<stddef.h> (cstddef)** - niekoľko užitočných dátových typov a makier
- **<stdio.h> (cstdio)** - podpora vstupu a výstupu (funkcionálna)
- **<stdlib.h> (cstdlib)** - generátor náhodných čísiel, pamäť, kontrola procesov, vyhľadávanie a triedenie
- **<string.h> (cstring)** - práca s reťazcami znakov
- **<tgmath.h>** - matematické funkcie pre prácu so zabudovanými typmi
- **<time.h> (ctime)** - konverzie časov a dátumov
- **<wchar.h> (cwchar)** - viacbytové znaky (UNICODE)
- **<wctype.h> (wcwctype)** - klasifikácia znakov

# Objektovo-orientované

- Knižnica tried a funkcií
- Založená na konvencií STL
- Hlavičkové súbory bez .h
- Vo vývoji, doplňovaná

# Kontajnery

- `<bitset>` - `std::bitset` – bitové pole
- `<deque>` - `std::deque` – obojstranný rad
- `<list>` - `std::list` – obojsmerne zreťazený zoznam
- `<map>` - `std::map` a `std::multimap` – utriedené asociatívne pole
- `<queue>` - `std::queue` – jednostranný rad
- `<set>` - `std::set` a `std::multiset` – množiny
- `<stack>` - `std::stack` – zásobník
- `<vector>` - `std::vector` – dynamické polia

# Všeobecné

- `<algorithm>` - definície kontajnerových algoritmov
- `<functional>` - funkcionálne objekty pre štandardné algoritmy
- `<iterator>` - práca s iterátormi
- `<locale>` - triedy a šablóny pre prácu s lokalizačnými dátami
- `<memory>` - správa pamäti
- `<stdexcept>` - štandardné výnimky
- `<utility>` - `std::pair` – práca s usporiadanými dvojicami



# Reťazce - string

- `<string>` - štandardné reťazcové triedy

# Vstup/Výstup - prúdy

- `<fstream>` - súborový vstup-výstup
- `<ios>` - typy a funkcie pre operácie s prúdmi
- `<iosfwd>` - preddeklarácie niektorých V/V šablón
- `<iomanip>` - formátovanie výstupu
- `<istream>` - triedy pre podporu vstupu
- `<ostream>` - triedy pre podporu výstupu
- `<sstream>` - triedy pre prácu s reťazcami
- `<streambuf>` - bufrovanie

# Numerické

- `<complex>` - komplexné čísla
- `<numerics>` - numerické spracovanie
- `<valarray>` - pole pre numerické spracovanie

# Podpora jazyka

- **<exception>** - spracovanie výnimiek
- **<limits>** - základné číselné typy – vlastnosti
- **<new>** - operátory new a delete – správa pamäti
- **<typeinfo>** - runtime typová informácia

# Doplnkové

- Vylepšenie štandardných knižničných funkcií – strcpy – strcpy\_s (doplňované tvorcami prekladačov)
- Podpora konkrétneho operačného systému – systémovo závislé – OWL, MFC, wxWidgets, Qt atď.
- Aplikačno-špecifické

# 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 jazyka C
- Existuje štandardná knižnica vstupno-výstupných funkcií
- Každý program, ktorý chce využívať túto knižnicu, musí obsahovať niekde riadok:  
**#include <stdio.h>**
- 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][F|N|h|l|L]typ

**priznak:**

- prem. bude zarovnaná vľavo
- + prem. bude zarovnaná vpravo
- # alternatívne vytlačiť (závisí od typu)

**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               |

# scanf

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

- kde

**format:**

**%[sirka][h|l|L]typ**

**sirka**

maximálna veľkosť pri čítaní zo vstupu

**F|N|h|l|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

# Formátová konverzia v pamäti

- **sprintf:**

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

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

- **sscanf:**

- podobné ako scanf, ale vstup ide zo string-u
- 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`, ...
- Pri spustení programu sú otvorené prúdy streamy  
`stdin` – štandardný vstupný stream  
`stdout` – štandardný výstupný stream  
`stderr` – štandardný chybový stream

# 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);  
}
```

- Čo ak sa v príklade zmení „int ch“ na „char 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 konštánt:**

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);
}
```

# **SDL Simple DirectMedia Layer**

# Inicializácia

- Inicializácia

```
#include <sdl.h>
```

```
int main(int argc, char *argv[])
{
    SDL_Init( SDL_INIT_EVERYTHING );
    SDL_Surface *screen;
    screen = SDL_SetVideoMode( 800, 600, 32, SDL_SWSURFACE);
    if ( screen == NULL ) {
        fprintf(stderr, "Problem: %s\n", SDL_GetError());
        exit(1);
    }
    /* ... */
    SDL_Quit( );
    return 0;
}
```

# Kreslenie

- Vymazanie obrazovky

```
SDL_FillRect( SDL_GetVideoSurface(), NULL, color);
```

- Kreslenie obdĺžnika

```
SDL_Rect box;
```

```
box.x=...; box.y=...; box.w=...; box.h=...;
```

```
SDL_FillRect(screen, &box, color);
```

- Tvorba farby

```
SDL_MapRGB(screen->format,R,G,B));
```

```
// R, G, B – farebné zložky <0..255>
```

- Aktualizácia obrazovky

```
SDL_UpdateRect(screen,0,0,0,0);
```

# Obrázok

- Načítava štandardne BMP

```
SDL_Surface* bmp = SDL_LoadBMP( „subor.bmp“ );
```

- Iné typy (PNG, JPG, ...) pomocou knižnice SDL\_Image

```
SDL_Surface* img = IMG_Load( „subor.png“ );
```

- Uvolnenie pamäti na konci programu

```
SDL_FreeSurface( bmp );
```

```
SDL_FreeSurface( img );
```

- Vykreslenie obrázku

```
DrawImage( screen, bmp, x, y ); // nie je funkcia v SDL
```

# Vykreslenie obrázku

```
void DrawImage( SDL_Surface* dest_surface, SDL_Surface*
    src_surface, int x, int y )
{
    SDL_Rect src_rect ;
    src_rect.x = 0 ;
    src_rect.y = 0 ;
    src_rect.w = src_surface->w ;
    src_rect.h = src_surface->h ;

    SDL_Rect dest_rect ;
    dest_rect.x = x ;
    dest_rect.y = y ;

    SDL_BlitSurface( src_surface, &src_rect, dest_surface,
        &dest_rect ) ;
}
```

# Vstupy

- Čakanie na vstup

```
SDL_Event event;  
SDL_WaitEvent(&event);
```

- Náhľad na vstup

```
SDL_Event event;  
SDL_PollEvent(&event);
```

- event.type

```
SDL_MOUSEMOTION  
SDL_MOUSEBUTTONDOWN  
SDL_KEYDOWN  
SDL_QUIT
```



# Hlavná slučka

```
while(1){  
    while ( SDL_WaitEvent(&event) ) {  
        switch (event.type) {  
            /* obsluha vstupov */  
            case SDL_QUIT:  
                exit(0);  
        }  
        /* vykreslenie sceny */  
        SDL_UpdateRect(screen,0,0,0,0);  
    }  
}
```

# Text

- Potrebujeme ďalšiu knižnicu „SDL\_ttf“
- Natiahnutie TrueType fontu

```
TTF_Font* font;  
font = TTF_OpenFont(„arialn.ttf“, 24);  
if (font == NULL){  
    printf(„TTF problem: %s %s \n“, file, TTF_GetError());  
}
```

- Prevod textu na SDL\_Surface

```
SDL_Surface* txt_surface;  
SDL_Color farba={255,0,0,0};  
txt_surface = TTF_RenderText_Solid(font, text, farba);
```

- Vykreslenie

```
DrawImage( screen, txt_surface, x, y );
```