Programovací jazyk C++

Hodina 1

Používané překladače

- Bloodshed Dev C++
  - http://www.bloodshed.net/devcpp.html
- CodeBlocks
  - http://www.codeblocks.org
    - pokud nemáte již nainstalovaný překladač, stáhněte si instalátor s MINGW v názvu
      - jinak se Vám nainstaluje pouze vývojové prostředí
- event. MS Visual C++
- (CodeGear C++ Builder)

Opakování

- Co je vyšší programovací jazyk?
  - kompilační vs. interpretační, příklady
- Co je překlad, překladač a jak funguje?
  - syntaxe, sémantika jazyka
  - lexikální symboly, lexikální a syntaktická analýza
  - jazyk relativních instrukcí (soubory .obj)
  - sestavení programu (linkování, linker), knihovny

Jazyk C

- 70. léta - AT&T Bell Laboratories
- Kernighan a Ritchie - verze K&R C
- silná vazba na UNIX
- 1988 - standardizace ANSI normou - ANSI C
  - ANSI 99
  - ANSI C1X (2011)

Jak probíhá překlad v jazyce C ?

- hlavičkové soubory .h (.hpp)
  - obsahují prototypy funkcí (hlavičky)
- preprocesor
  - provádí textové úpravy zdrojového kódu (náhrady, vložení hlavičkových souborů)
  - direktivy preprocesoru – začínají znakem #, neukončují se středníkem
  - příklad:

```
#define MAX 10
#include <stdio.h>
```

Jak probíhá překlad v jazyce C ?

Poznámka:

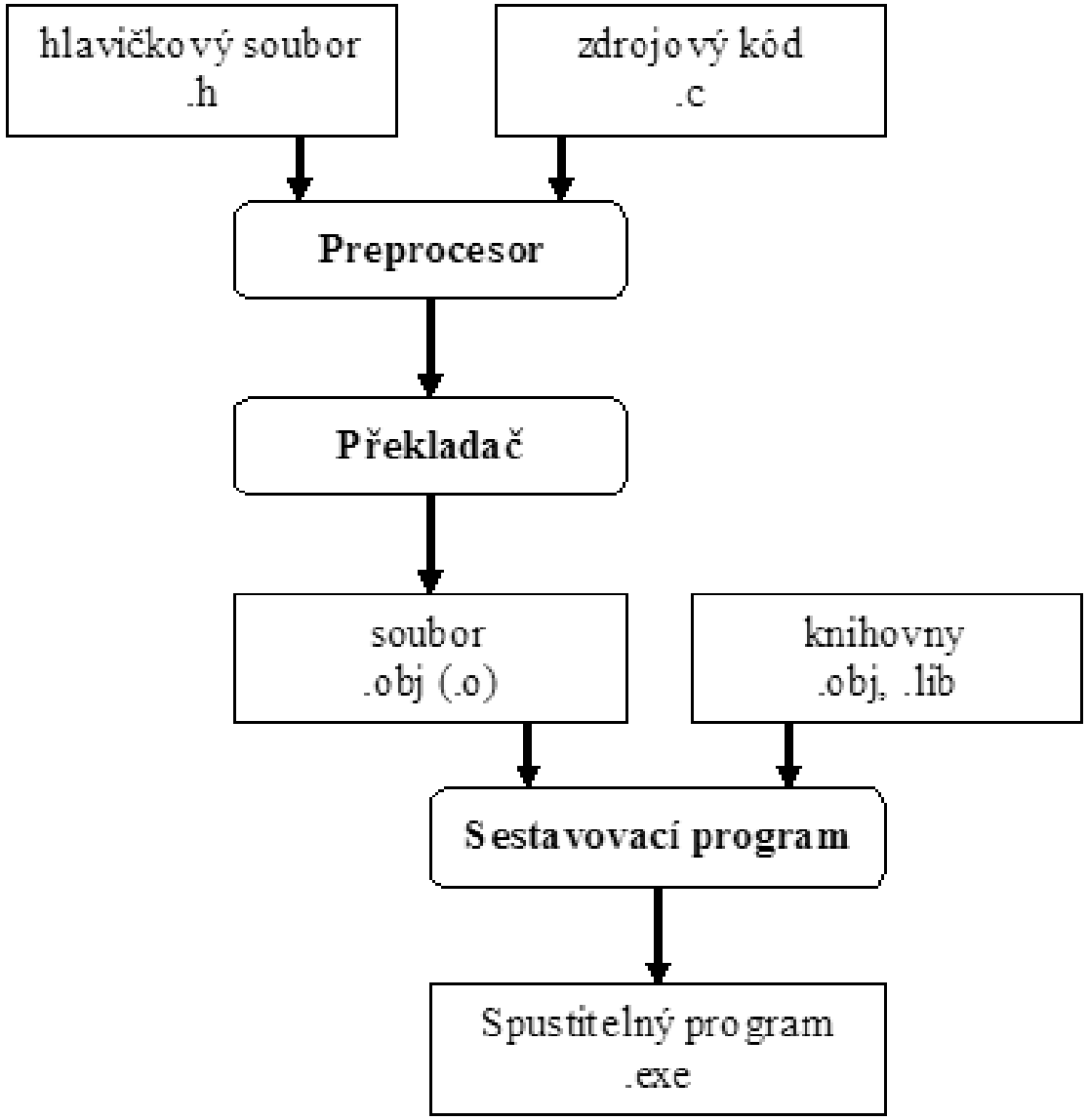
```
#include <stdio.h>
```

- preprocesor hledá soubor `stdio.h` ve standardních adresářích překladače (zpravidla podadresář `include`)

```
#include "soubor.h"
```

- preprocesor hledá soubor `soubor.h` v aktuálním adresáři (např. kde je uložen projekt)

```
#include "incl/soubor.h„
#include <ole/access.h>
```


hlavičkový soubor
.h
zdrojový kód
.c
Preprocesor
Překladač
soubor
.obj (.o)
knihovny
.obj, .lib
Sestavovací program
Spustitelný program
.exe

- pomocí příkazů preprocesoru lze realizovat *podmíněný překlad:*

```
#define PRACOVNI


#ifdef PRACOVNI
  printf("Kontrolni tisk");
#endif
```

- pomocí příkazů preprocesoru lze realizovat *podmíněný překlad:*

```
#define PRACOVNI


#ifdef PRACOVNI
  printf("Kontrolni tisk");
#else
  printf("Program pro vypocet");
#endif
```

Programovací jazyk C++

- 1983 - AT&T Bell Laboratories
  Bjarne Stroustrup - „Jazyk C s objekty“
- 1986 Bjarne Stroustrup

  The C++ Programming Language
- 1990 Bjarne Stroustrup:

  The Annotated C++ Reference Manual

- 1995 M.Ellis, B.Stroustrup
  The Annotated C++ Reference Manual
- 1997 - ANSI/ISO standardizace C++
- 2011 – poslední standard C++11
- jazyk C++ je rozšířené C-čko o objektové rysy, ale *to není zcela přesné:*
  - C není podmnožinou C++, některé neobjektové rysy C++ jsou jiné oproti standardnímu jazyku C

- **přesto** lze většinu programů zapsaných v jazyce C přeložit pod překladačem C++, zejména pokud jde o zápis podle ANSI C
- některé neobjektové změny oproti C:
  - odstraněn starý způsob deklarace hlaviček funkcí podle K&R
  - f() = f(void)
  - `inline` funkce
  - implicitní parametry funkcí

Opakování základů jazyka C

- jazyk se slabou typovou kontrolou (v C++ je zesílená)
- case sensitive (rozdíl mezi malými a velkými písmeny
- rysy funkcionálního jazyka
- označení bloku (těla funkce) - { }
- hlavní program: funkce `main`
  - `void main()`
  - `int main(int argc, char* argv)`

Literály:

- *konstanty*
  - desítkové konstanty: 13, -5, 15L
  - osmičkové konstanty: 056
  - šestnáctkové konstanty: 0x20, 0X1F
  - desetinná čísla: 5.3, -4E-3
  - znakové: `'a','\n','\t','0x0A'`
  - řetězcové: `"Ahoj, světe!"`
- *indentifikátory proměnných*
  - musí začínat písmenem, uvažovaná délka závisí na překladači (zpravidla 31), C++ vše

Datové typy - jednoduché

- *celočíselné*
  - char (8 bitů)
  - short, int, long (unsigned, signed), long long
  - sizeof(short) ≤ sizeof(int) ≤ sizeof(long) ≤ sizeof(long long)
- *plovoucí řádová čárka (desetinná čísla)*
  - float, double, long double (ANSI)
  - v některých překladačích i long long double
- *výčtový typ* - enum
- *jak je to s typem boolean?*

- v jazyce C není, nahrazuje se typem **`int`**
  - jakákoliv nenulová hodnota je považována za „pravdu“,
- v C++ je definován datový typ **`bool`** s hodnotami **`true, false`**

```
bool je_otevren = true;
if (je_otevren) …
if (je_otevren == false) …
```

- řetězec
  - není, řetězce jsou pole znaků (char*)
  - v C++ je již definován v knihovně STL typ `string`

Typické velikostí typů v 32 bit. překladači

Typ	Velikost [bit]	Rozsah
`char`	8	-128 až +127
`short`	16	-32 768 až +32 768
`int`	32	-2 147 483 648 až +2 147 48 3647
`long`	32 (64)	-2 147 483 648 až +2 147 48 3647
`long long`	64	-9 223 372 036 854 775 808 až +9 223 372 036 854 775 807
`float`	32	$-3{,}402823 \cdot 10^{+38}$ až $+3{,}402823 \cdot 10^{+38}$ nejnižší kladná hodnota $1{,}175494 \cdot 10^{-38}$ počet platných cifer 7 až 8
`double`	64	-1, 7976931348623157·10^{+308} až +1, 7976931348623157·103^{+308} nejnižší kladná hodnota $2{,}225073858507202 \cdot 10^{-308}$ počet platných cifer 15 až 16
`long double`	80 (96)	$-3{,}4 \cdot 10^{+4932}$ až $+3{,}4 \cdot 10^{+4932}$ nejnižší kladná hodnota $1{,}1 \cdot 10^{-4932}$ počet platných cifer 19

Typické velikostí typů v 32 bit. překladači

- **standard jazyka C nedefinuje velikost typů, záleží to na překladači (platformě)**
- **musí pouze platit nerovnost:**

`sizeof(short) ≤ sizeof(int) ≤ sizeof(long) ≤ sizeof(long long)`

- rozsahy typů jsou definovány jako konstanty v hlavičkových souborech `limits.h` a `float.h`
  - `INT_MAX, INT_MIN, LONG_MAX, ...`
  - `FLT_MAX, DBL_MAX, LDBL_MIN, ...`

Komentáře

- víceřádkové komentáře v klasickém C

```
/* Toto je víceřádkový
   komentář  */
```

- jednořádkové komentáře v C ++

```
// Toto je jednořádkový komentář
```

Proměnné:

- globální
- lokální (v bloku)
- deklarace proměnné:

```
typ proměnná;
int x=10; // počát. inicializace
char c;
float polomer, obsah;
TBarvy barva_auta;
```

```
/* Proměnná volba je globální, viditelná ve
   funkci obsah i v hlavním programu */
int volba;
int obsah()
{/* Proměnné x a y jsou lokální ve funkci
   obsah */
   float x, y;
}
int main(int argc, char **argv)
{
/* Proměnná  a je lokální viditelná pouze v
   hlavním programu */
   int a;
}
```

Kde je možné v jazyce C deklarovat proměnnou?

- klasické C-čko
  - pouze na počátku bloku, před kódem
- C++
  - kdekoliv, až před použitím
    - důsledek: náročnější překlad, ale přehlednější kód

Platnosti dekrarací

```
int main()
{  int i;
   …
   for(i=0;i<10;i++)
   {  int x;
      …
   }
   /* zde je platné i, x již ne */
}
```

Překrývání

```
int main()
{  int x;
   …
   for(i=0;i<10;i++)
   { int x; /* toto x překryje předchozí
     …
   }
   /* zde je platné „původní“ x */
}
```

Definice nového (uživatelského) typu

- klíčové slovo **`typedef`**
- jako příklad si ukážeme definici tzv. výčtového typu
  - v příštím cvičení definici strukturovaného typu

Problém:

- v programu chceme uchovávat informaci o barvě, např. barvě auta

Řešení

1. prostými číselnými hodnotami

```
int barva;
barva = 0;
if (barva == 0)
```

–někde na papír si poznamenám: 0 znamená černá, 1 znamená červená, ..

- nejhorší možný způsob

–o něco lépe: tuto informaci zapíši do programu jako komentář

```
/* 0 - černá, 1 - červená */
```

Řešení

2. definice konstant v programu

```
#define CERNA 0
#define CERVENA 1
int barva;
barva = CERVENA;
if (barva == CERVENA)
```

- přehledné řešení

3. definice výčtového typu

- na začátku programu nebo do samostatného hlavičkového souboru napíši definici deklaraci

```
typedef enum
{CERNA, CERVENA, BILA} Barvy;
```

- v programu definuji proměnou typu `Barvy`

```
Barvy barva;
barva = BILA;
if (barva == CERVENA)
```

- jak je výčtový typ implementován vnitřně?

```
typedef enum
{CERNA, CERVENA, BILA} Barvy;
```

  - celočíselně, `CERNA` hodnotou 0, `CERVENA` hodnotou 1, `BILA` hodnotou 2

  - hodnoty může předepsat programátor

```
typedef enum {CERNA, CERVENA,
BILA=6} Barvy;
```

Konstanty

Jak je to v klasickém C s konstantami?

- původně jen symbolické

```
#define MAX 10
```

- v ANSI C a C++ klíčové slovo **`const`**
  - typované a netypované
  - nelze využít všude (meze polí)

```
const int MAX = 10;
```

Přiřazení:

- přiřazovací výraz:

```
y = 3*a + 12    i=j=1
```

- přiřazovací příkaz
  - výraz ukončený ; je příkaz

```
y = 3*a + 12; a = 10; c = 'a';
  barva_auta = BILA;
```

Operace:

- aritmetické: +, - ,* , /
  - % - zbytek po celočíselném dělení
  - pozor: dělení celočíselné, neceločíselné
    - typ operace záleží na typu operandů
    - `5/3, 5/3.0`
    - `int a,b; a/b, (double) a/b`
    - detaily viz Cvičení 5 předmětu Úvod do programování
- bitové: &, |, ~, ^

Bitové operátory

Operátor	**Operace**
&	Bitový součin (AND)
|	Bitový součet (OR)
^	Exlusivní bitový součet - výlučné NEBO (XOR)
~	Bitová negace
<<	Bitový posuv vlevo
>>	Bitový posuv vpravo

```
unsigned char a = 0x85;
/* 133 desitkove, 10000101 dvojkove */
unsigned char b = 0x46;
/* 70 desitkove, 01000110 dvojkove */
unsigned char c,d,e,f,g,h;
c = a & b;
d = a | b;
e = a ^ b;
f = ~ a;
g = a << 2;
h = b >> 3;
```

Výsledky bitových operací

10000101	10000101	10000101	10000101
&	|	^	~
01000110	01000110	01000110	
00000100	11000111	11000011	01111010

Výsledky bitových operací

```
10000101   01000110

  << 2       >> 3
```

```
00010100   00001000
```

K čemu je to dobré ?

- určitá funkce vrací např. 8 bitovou hodnotu a každý bit znamená nějakou chybu; číslo je uloženo v proměnné chyba

```
00010100
```

- potřebuji zjistit, zda je tento bit nastaven na 1
- k testu použiji číslo, kterému se říká maska:

```
00000100
00000100 bin = 04 hex
```

- **if** (chyba & x04 != 0)

- unární inkrementace, dekrementace (post, pre): `++, --`

```
a++; ++a; a--; --a;
y = 3*a++;  z = 5/--a;
```

  - detaily viz Cvičení 5 předmětu Programování
- přiřazovací operátor `op=`

```
y *= 3;
```
stejné jako `y = y*3;`

```
y += 3;
y >>= 2;
```

Příkazy

- každý příkaz ukončen středníkem
- podmínka:

```
if (a < 5) chyba();
if (x != 0) vypocet(); else err();
```

- relační operátory:

  `<, <=, >, >=, !=, ==`

- logické operátory:

  `&&, ||, !` - menší priorita než aritmetické

Pozor:

```
if (a == 5) ...
```
porovnání

```
if (a = 5) ...
```
přiřazení

```
int c;
if ((c=getchar())=='\n')
```

```
if (a) ...
```
porovnání s 0

je to samé jako `if (a != 0) ...`

- **cykly:**

```
while(výraz) příkaz;
do příkaz; while(výraz);
do { příkaz1; příkaz2; }
  while(výraz);
for(výraz1;výraz2;výraz3) příkaz;
```

ekvivalentní s:

```
výraz1;
while(výraz2) { příkaz;  výraz3; }
```

– nejčastější použití:

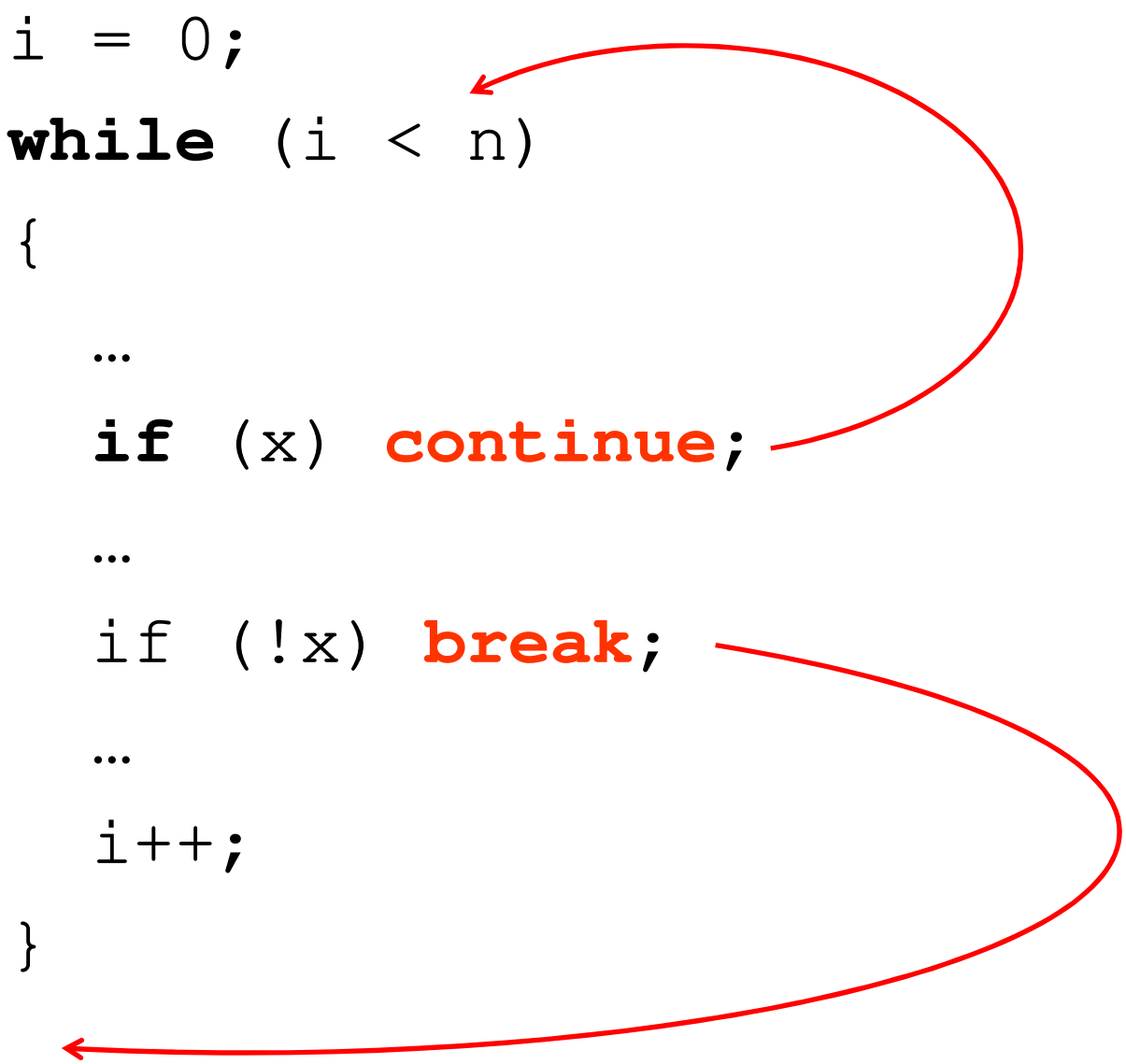
```
for(i=1;i<n;i++)
```

- lze i:

```
for(i=1;i<n;i=i+2)
```

- pozor na středník!
- co kdybych napsal čárku?
  - čárka v jazyce C je speciální operátor
  - detaily Přednáška 2 předmětu Programování 1

- **`continue`** a **`break`** v cyklu
  - `break` předčasně ukončí cyklus
  - `continue` ukončí provádění těla a skočí na testovací podmínku
  - mohou být použity ve všech cyklech


i = 0;
while (i < n)
{
…
if (x) continue;
…
if (!x) break;
…
i++;
}

- **větvení:**

```
switch (x)
{
   case 1: příkaz1; break;
   case 2:
   case 3: příkaz3;
   default: příkaz4;
}
```

- pozor na **break**
- **continue** do příkazu **switch** nepatří!

Konzolový vstup a výstup

- knihovna `stdio.h`
- `printf(const char*format,...)`
- `scanf(const char*format, ...)`
- pozor u scanf: operátor &

```
scanf("%d",&x)
```

Příklad č. 1

Napište program v „klasickém C“, který vyzve uživatele k zadání velikosti dvou stran obdélníka, zkontroluje, zda se data správně načetla a zda jsou větší než 0. Program vytiskne obsah a obvod obdélníka.

Program okomentujte /* */. Pro konzolový vstup a výstup použijte funkce z knihovny stdio.h

Příklad č. 2

- pomocí podmíněného překladu vytvořte dvě jazykové verze programu (anglickou a českou)
- vyzkoušejte výstup preprocesoru
  - spusťte překladač z příkazové řádky:
  - `gcc obsah.c -E`