Prezentace a vysvětlení programového prostředí NXC

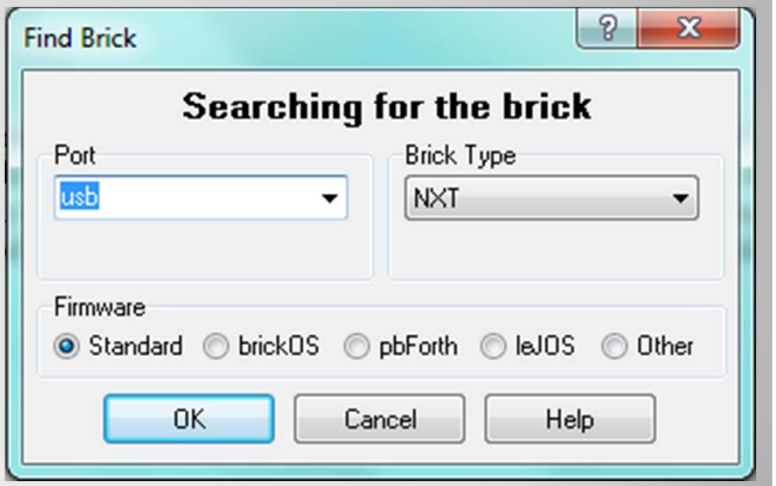
Úvod

- Další jazyk, který je možno použít pro programování NXT kostky je NXC – **N**ot e**X**atly **C**
- Na rozdíl od jazyku NXT-G, kde jsme vytvářeli program pomocí **grafických prvků** – přesněji řečeno pomocí programovacích bloků, tak jazyk NXC využívá instrukce v **textové podobě**
- Jazyk NXC je velmi podobný jazyku C
- Programy budete vytvářet v programovacím prostředí **Bricx Command Center - BricxCC**

Prezentace a vysvětlení programového prostředí NXC

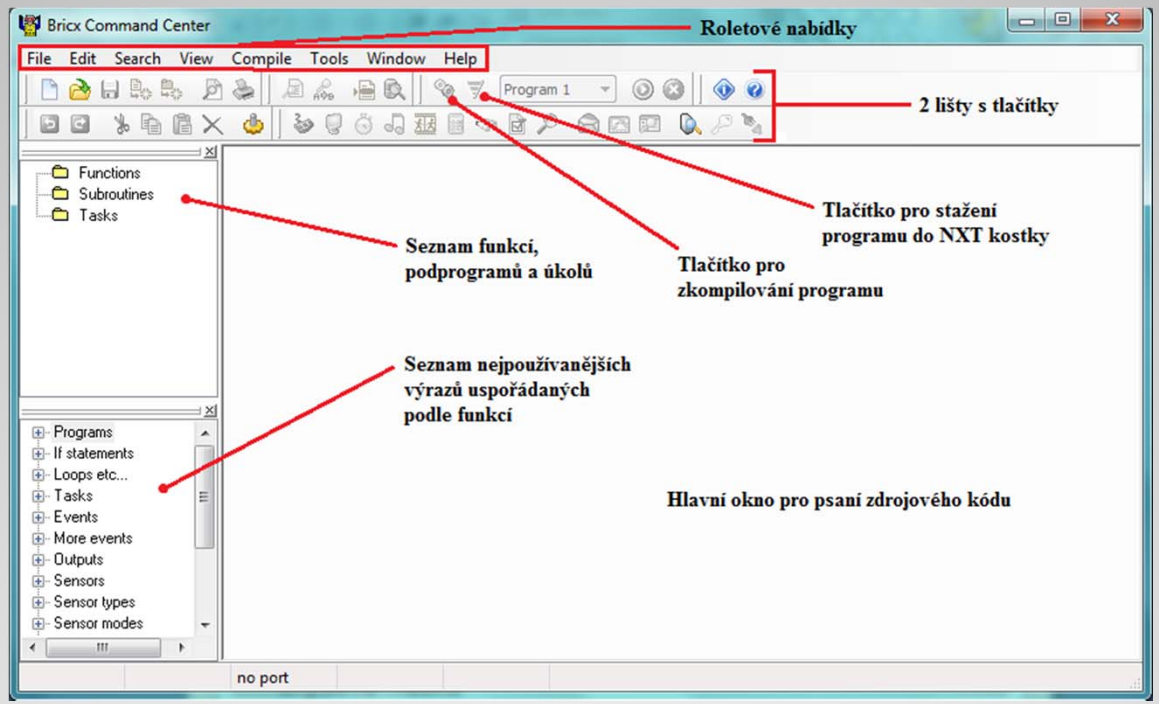
Bricx Command Center

- Při každém spuštění BricxCC se objeví okno pro identifikaci připojené NXT kostky
- V roletovém menu *Port* zvolte buď *Automatic* nebo *usb*, v sekci *Brick Type* zvolte *NXT* a v sekci *Firmware* vyberte *Standard*
- Kliknutím na tlačítko *OK* přejdete k programovacímu prostředí
- Pokud programovací kostka není připojena nebo špatně zvolíte parametry, tak se objeví chybová hláška



Prezentace a vysvětlení programového prostředí NXC

Bricx Command Center



- Pokud programovací kostku připojíte později a chcete ji inicializovat, tak okno pro inicializace vyvoláte následujícím postupem:
    1. Klikněte na roletovou nabídku *Tools*
    2. Vyberte volbu *Find Brick*

Prezentace a vysvětlení programového prostředí NXC

Obecné informace o NXC

- NXT kostka má byte-kódový interpretr, který slouží k vykonávání programů
- NXC překladač překládá zdrojový program do NXT byte-kódu, který potom může být vykonán
- Jazyk NXC vychází z jazyka C
- NXC není obecný programovací jazyk => Existuje mnoho omezení, které vyplývají z omezení byte-kódového interpretru NXT kostky

Prezentace a vysvětlení programového prostředí NXC

Obecné informace o NXC

- NXC je jazyk „*case-sensitive*“, jako jazyk C nebo C++ - rozlišují se velká a malá písmena; např. : „AbC“ není stejné jako „abc“
- Každá deklarace je zakončena středníkem
- Každá struktura je uzavřena dvojicí složených závorek

Prezentace a vysvětlení programového prostředí NXC

Lexikální pravidla

- Jedná se o souhrn pravidel, které popisují, jak má vypadat struktura zdrojového kódu – jak psát komentáře, manipulovat s mezerami, nebo jaké jsou povolené znaky pro identifikátory

Komentáře

- Existují dvě podoby komentářů
  - Klasický „Céčkovský" komentář – začíná „ /* " a končí „ */ ".
    Tento typ dovoluje vytvářet víceřádkové komentáře, ale nemohou být do sebe vnořeny.

```
/* toto je například komentář */

/* toto je dvouřádkový
                komentář */
```

Prezentace a vysvětlení programového prostředí NXC

Lexikální pravidla

Komentáře

- Druhý typ komentářů podporovaný v NXC začíná „ // “ a pokračuje do konce aktuálního řádku

```
// jednořádkový komentář
```

- Překladač komentáře ignoruje
- Slouží programátorovi k vytváření poznámek ve zdrojovém kódu, usnadňují orientaci v kódu

Prezentace a vysvětlení programového prostředí NXC

Lexikální pravidla

Prázdné místo

- Prázdné místo se skládá ze všech mezer, tabulátorů a nových řádků
- Slouží k oddělení znaků a udělání programu více přehledným
- Pokud jsou znaky rozlišitelné, tak přidání nebo odebráni mezery nemá žádný vliv na význam programu
- Následující kus kódu má stejný význam:

```
x=2;
x = 2;
x   = 2;
```

Prezentace a vysvětlení programového prostředí NXC

Lexikální pravidla

Prázdné místo

- Některé operátory se skládají z více znaků a nemůže se mezi nimi objevit mezera
- Na příkladu níže na prvním řádku je použit operátor „ >> “ pro bitový posun vpravo; na druhém řádku je ale vložena mezera za znak „ > “. Znaky jsou pak interpretovány jako dva samostatné a překladač nahlásí chybu.

```
// nastaví proměnnou x na číslo, které vznikne posunutím 1 o 4 bity doprava
x = 1 >> 4;

// chyba
x = 1 > > 4;
```

Prezentace a vysvětlení programového prostředí NXC

Lexikální pravidla

Číselné konstanty

- Mohou být v decimální - desítkové nebo hexadecimální - šestnáctkové formě
- Desetinné konstanty se skládají z desetinné čárky a z jedné nebo více desetinných míst za desetinnou čárkou
- Číselné konstanty v hexadecimální formě začínají „0x“ nebo „0X“ následované jednou nebo více šestnáctkovými číslicemi

```
// nastaví proměnnou x1 na hodnotu 10
x1 = 10;

// nastaví proměnnou x2 na hodnotu 16 (10 hexa)
x2 = 0x10;

// nastaví proměnnou x3 na hodnotu 1.5
x3 = 1.5;
```

Prezentace a vysvětlení programového prostředí NXC

Lexikální pravidla

Textové řetězce a charakteristické konstanty

- Konstanty ve formě textových řetězců jsou v NXC ohraničeny dvojitými uvozovkami
- Textové řetězce jsou na pozadí automaticky konvertovány do pole bytů, kde poslední byte v poli je nulový. Poslední bytová nula se obvykle označuje „null terminator"
- Například:

```
// Na první řádek displeje NXT kostky se vypíše: testovani
TextOut(0, LCD_LINE1, "testovani");
```

- Charakteristické konstanty jsou v NXC ohraničeny jednoduchými uvozovkami a mohou obsahovat jeden znak ASCII. Hodnota charakteristické konstanty je číselná ASCII hodnota znaku
- Například:

```
char ch = 'a'; // ch == 97
```

Prezentace a vysvětlení programového prostředí NXC

Lexikální pravidla

Identifikátory a klíčová slova

- **Identifikátory** se používají jako jména pro proměnné, úkoly, funkce a podprogramy
- První znak identifikátoru musí být velké nebo malé písmeno nebo podtržítko „ _ “. Zbývající znaky mohou být písmena, čísla a podtržítka.
- **Klíčová slova** jsou vyhrazené skupiny znaků v jazyku NXC, které nemohou být použity jako identifikátory.
- Kompletní seznam klíčových slov:

  asm, bool, break, byte, case, char, const, continue, default, do, else, enum, false, float, for, goto, if, inline, int, long, mutex, priority, repeat, return, safecall, short, start, stop, string, struct, sub, switch, task, true, typedef, unsigned, until, void, while

Prezentace a vysvětlení programového prostředí NXC

Struktura programu

- NXC program se skládá z kódových bloků a proměnných
- Existují dva odlišné typy kódových bloků:
  - Úlohy
  - Funkce
- Každý typ kódového bloku má jedinečné vlastnosti, ale struktura je společná
- Maximální počet kódových bloků a funkcí může být 256

Prezentace a vysvětlení programového prostředí NXC

Struktura programu

Úkoly - Tasks

- Úkol v jazyku NXC odpovídá jednomu vláknu
- Úkoly jsou definovány pomocí klíčového slova „task" a syntaxi můžete vidět na kódu níže

```
task jmeno()
{
  // zde je umístěn kód úkolu
}
```

- Název úkolu může být jakýkoliv povolený identifikátor
- Program musí mít vždy alespoň jeden úkol s názvem „main", který se spustí při každém spuštění programu
- Tělo úkolu se skládá ze seznamu příkazů

Prezentace a vysvětlení programového prostředí NXC

Struktura programu

Úkoly - Tasks

- Úkoly se nejčastěji spouští pomocí příkazu „*StartTask(jmeno ukolu)*“
- Úkoly v řadě za sebou se spustí příkazem „*Follows(ukol_1, ukol_2,…, ukol_N)*“ – *následující úkol je spuštěn poté, co skončil předchozí*
- Příkazem *Precedes(ukol_1, ukol_2,…, ukol_N*) se spustí úkoly současně
- Všechny aktuálně běžící úkoly se zastaví příkazem *StopTask(jméno úkolu)*
- Všechny úkoly se zastaví příkazem *StopAllTasks()* – nezastaví se ale samotný program, to se provede příkazem *Stop()*
- Příkazem *ExitTo(jmeno dalsiho ukolu)* se ukončí současný úkol a zároveň spustí definovaný úkol
- Úkol se ukončí jednoduše automaticky dosažením konce těla úkolu

Prezentace a vysvětlení programového prostředí NXC

Struktura programu

Funkce - Functions

- Občas je potřebné skupinu příkazů sloučit do jediné funkce, kterou může váš kód volat podle potřeby
- NXC podporuje funkce se vstupními argumenty a návratovými hodnotami
- Do funkce můžeme pokaždé poslat jiné vstupní argumenty, takže se funkce může pokaždé chovat jinak
- Syntaxe funkce je následující:

```
navratovy_typ jmenoFunkce(seznam argumentu)
{
  // tělo funkce
}
```

- **Návratový typ** je typ proměnné, kterou funkce vrací
- Pokud funkce nevrací žádnou hodnotu, tak používáme *void* nebo *sub*

Prezentace a vysvětlení programového prostředí NXC

Struktura programu

Funkce - Functions

- **Seznam argumentů** může být prázdný nebo může obsahovat jeden a více argumentů, které jsou od sebe odděleny čárkou „ , “
- Argument je představován *typem* a *jménem proměnné*
- Proměnná může být typu bool, char, byte, int, short, long, unsigned int, float, strings, struct typy nebo array
- Příklad funkce, která nemá žádný vstup a výstupem je proměnná typu int:

```
int cislo() //návratová hodnota je typu int
{
  int x = 1;
  return x; //příkazem return určujeme, kterou proměnnou funkce vrátí
}

//pokud chceme funkci zavolat, tak použijeme následující kód:
int a = cislo(); //do proměnné a se uloží výstup funkce
```

Prezentace a vysvětlení programového prostředí NXC

Struktura programu

Funkce - Functions

- Příklad funkce, která má dva vstupní argumenty a výstupem je proměnná typu int:

```
// funkce pro výpočet součinu dvou čísel
int soucin(int x, int y) //dva vstupní argumenty typu int
{
    return x * y; //funkce vrací součin dvou čísel
}

//ukázka volání funkce v příkazu pro zobrazení textu na displeji NXT kostky
task main()
{
     NumOut(0, LCD_LINE1, foo(10, 20)); // výstup funkce je 200
     NumOut(0, LCD_LINE2, foo(10, 5)); //výstup funkce je 50
     Wait(SEC_10); // čeká 10 sekund
}
```

Prezentace a vysvětlení programového prostředí NXC

Struktura programu

Funkce - Functions

- Příklad funkce, která má dva vstupní argumenty a nemá žádnou návratovou hodnotu:

```
// funkce pro výpis součinu dvou čísel na displeji NXT kostky
void vypis(int x, int y) //dva vstupní argumenty typu int
{
  NumOut(0, LCD_LINE1, x*y);
  Wait(SEC_10); // čeká 10 sekund
}

task main()
{
    vypis(10, 20); //volání funkce
}
```

Prezentace a vysvětlení programového prostředí NXC

Struktura programu

Proměnné - Variables

- Proměnné mohou být následujícího typu:

Klíčové slovo proměnné	Informace
`bool`	true/false
`byte`	0 až 255
`char`	Znaky
`unsigned int`	0 až 65535
`int`	-32768 až 32767
`unsigned long`	0 až 4 294 967 295
`long`	–2 147 483 648 až 2 147 483 647
`mutex`	Speciální proměnná
`string`	Textové řetězce
`struct`	Uživatelem definovaná struktura
`klíčové slovo proměnné``[]`	Pole jakéhokoliv typu

- Proměnné jsou deklarovány pomocí klíčového slova proměnné a jména proměnné, deklarace je ukončena středníkem „ ; “

Tabulka převzata z: Bakalářské práce Ivana Moce, studenta 3. ročníku ČVUT, FEL - 2010

Prezentace a vysvětlení programového prostředí NXC

Struktura programu

Globální a lokální proměnné

- Globální proměnné jsou deklarovány mimo jakýkoliv kódový blok
- Jedna deklarace může být použita ve všech úkolech, funkcích nebo podprogramech
- Její rozsah působnosti začíná v místě deklarace a končí s koncem programu

x

- Lokální proměnné jsou deklarovány v úkolech a funkcích
- Jsou přístupné jen v rámci kódového bloku, ve kterém jsou deklarovány
- Jejich rozsah platnosti začíná v místě jejich deklarace a končí s koncem bloku, kde jsou deklarovány
- Složený příkaz ohraničený symboly „{" a „}" je považována také za blok

Prezentace a vysvětlení programového prostředí NXC

Struktura programu

Globální a lokální proměnné

- Příklady deklarací:

```
int a; //globální proměnná

task main()
{
    int b; //lokální proměnná
    a = b; //OK deklarace
      {      //začátek složeného příkazu
        int c; //lokální proměnná
        int cc = c; //OK deklarace
      }      //konec složeného příkazu
    int ccc = c;  //ŠPATNĚ - c zde nemá platnost
}

task pokus()
{
   a = 10; //OK deklarace
   b = 20: //ŠPATNĚ - y není globální proměnná
}
```

Prezentace a vysvětlení programového prostředí NXC

Struktura programu

Struktury - structures

- Používají se pro sdružení proměnných, které mají něco společného
- Uplatní se ve složitějších programech
- Ukázka na příkladu:

```
struct osoba  //vytvoření nové struktury typu osoba
{
string jmeno;
string prijmeni;
string pohlavi;
int vek;
};
```

```
osoba osoba_pokus; //vytvoření proměnné typu osoba
//přístup k proměnné uvnitř struktury
osoba_pokus.jmeno = "Jaroslav" ;
osoba_pokus.prijmeni = "Marek" ;
osoba_pokus.vek = 22;
```

- Po vytvoření struktury můžeme přistupovat na její proměnné
- K proměnným struktury přistupujeme pomocí jména proměnné-struktury, tečky a jména proměnné uvnitř struktury

Prezentace a vysvětlení programového prostředí NXC

Struktura programu

Pole – Arrays

- Deklarují se stejným způsobem jako běžné proměnné, ale ještě je třeba doplnit hranaté závorky „[]“
- Ukázka deklarace:

```
int moje_pole[]; // deklarace pole typu int s 0 prvky
```

- Lze vytvořit i vícedimenzionální pole přidáním dalšího páru hranatých závorek – max. počet v NXC je 4
- Ukázka deklarace:

```
int moje_pole[][]; // 2-dimenzionální pole typu int s 0 prvky
```

- Prvky pole jsou číslovány od nuly – první prvek je na pozici s indexem 0, druhý prvek je na pozici s indexem 1 a tak dále

Prezentace a vysvětlení programového prostředí NXC

Struktura programu

Pole – Arrays

- Ukázky deklarací:

```
//deklarace pole délky 4, které obsahuje 4 prvky s výchozí hodnotou 0
int pole1[4];
//přiřazení hodnoty prvkům pole
pole1[0] = 10; //přiřadí do pole na nultou pozici hodnotu 10
pole1[1] = 20; //přiřadí do pole na první pozici hodnotu 20
// => na pozici 0 bude 10, na pozici 1 bude 20 a na pozici 2 a 3 bude 0

/*deklarace prvků pole výčtem hodnot - nemusíme zadávat délku pole
  v hranatých závorkách*/
int pole2[] = {1, 2, 3, 4}; //pole délky 4
string autaPole[] = {"Audi" , "Skoda" , " Ford" }; //pole délky 3
```

- Pokud vytváříme prázdné pole, tak před použitím ho musíme inicializovat funkcí *ArrayInit* a určit jeho velikost

```
int array[];
ArrayInit(array, 1, 10); //inicializace pole - do pole se uloží 10 jedniček
```

Prezentace a vysvětlení programového prostředí NXC

Tvrzení

Přiřazení a další operace s proměnnými

- Dosud jsme se setkali s nejpoužívanějším operátorem pro přiřazení hodnoty výrazu do proměnné – „ = “
- Existují další operátory, které modifikují hodnotu proměnné jinými způsoby:

Operátor	Akce
=	Přiřazení výrazu do proměnné
+=	Přičtení výrazu k proměnné
-=	Odečtení výrazu od proměnné
*=	Vynásobení proměnné výrazem
/=	Vydělení proměnné výrazem
%=	Zbytek po dělení proměnné výrazem

Prezentace a vysvětlení programového prostředí NXC

- Pokračování tabulky

Operátor	Akce
&=	Bitový AND výrazu a proměnné
|=	Bitový OR výrazu a proměnné
^=	Bitový XOR výrazu a proměnné
||=	Absolutní hodnota výrazu
+-=	Do proměnné nastaví signum výrazu (-1,0,1)
>>=	Bitový posun doprava o počet bitů určených výrazem
<<=	Bitový posun doprava o počet bitů určených výrazem
++	Zvýšení hodnoty proměnné o 1
--	Snížení hodnoty proměnné o 1

Prezentace a vysvětlení programového prostředí NXC

- Příklad:

```
int x = 1; //deklarace proměnné x
int y = 2; //deklarace proměnné y

x += 2;  // k x se přičte 2, takže x = 3
y++;     // y se navýší o jedna, takže y = 3
x *= y;  // vynásobení x a y mezi sebou, takže x = 9
x %=5;   // zbytek po dělení 5 bude 4, takže x = 4
```

Prezentace a vysvětlení programového prostředí NXC

Tvrzení

- Operátory pro porovnávání proměnných:

Operátor	Funkce
==	Je rovno
<	Je menší
>	Je větší
<=	Je menší nebo rovno
>=	Je větší nebo rovno
!=	Není rovno
True	Pravda (ano)
False	Nepravda (ne)
&&	A zároveň
||	Nebo

Prezentace a vysvětlení programového prostředí NXC

Preprocesor

- NXC obsahuje preprocesor, který je modelovaný podle preprocesoru jazyka C
- Preprocesor je část programu, která se zpracovává ještě před samotným překladem kódu
- Slouží k implementaci externích knihoven, definování maker nebo proměnných
- Odlišnosti:
  - Neuvádíme typ proměnné
  - Příkaz není ukončen středníkem
  - Hodnota proměnné nejde během běhu programu měnit
- Ukázka deklarace:

```
#include "knihovna.h" //implementace knihovny
#define cas 60 //definování proměnné či makra
```

Prezentace a vysvětlení programového prostředí NXC

Seznam zdrojů:

[1] HANSEN, John. NXC Programmer's Guide [online]. 2010-05-30 [cit. 2010-05-23]. The NXC Guide. Dostupné z WWW: <http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC_Guide.pdf>

[2] MOC, Ivan. Využití robota LEGO MINDSTORMS – návrh a realizace úloh, návod na programování v NXC. Praha, 2010. 64 s. Bakalářská práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra řídící techniky.

[3] BENEDETTELLI, Daniele. Programming LEGO NXT Robots using NXC [online]. 2007-06-07 [cit. 2010-05-23]. The NXC Tutorial. Dostupné z WWW: <http://bricxcc.sourceforge.net/nbc/nxcdoc/NXC_tutorial.pdf>