Základy C++

doc. Mgr. Jiří Dvorský, Ph.D.

Katedra informatiky
Fakulta elektrotechniky a informatiky
VŠB – TU Ostrava

Prezentace ke dni 12. února 2020





Osnova přednášky

Základy C++

- Počítač
  - Hardware – základní struktura počítače
  - Software
  - Osobní počítače
- Programovací jazyk
- Jazyk C++
  - Vznik spustitelného programu
- Základní stavební bloky C++
  - Komentáře v C++
  - Datové typy v C++
- Proměné

Počítač

Počítač

- zařízení schopné provádět výpočty a rozhodnutí založená na logice.

Hardware (HW)

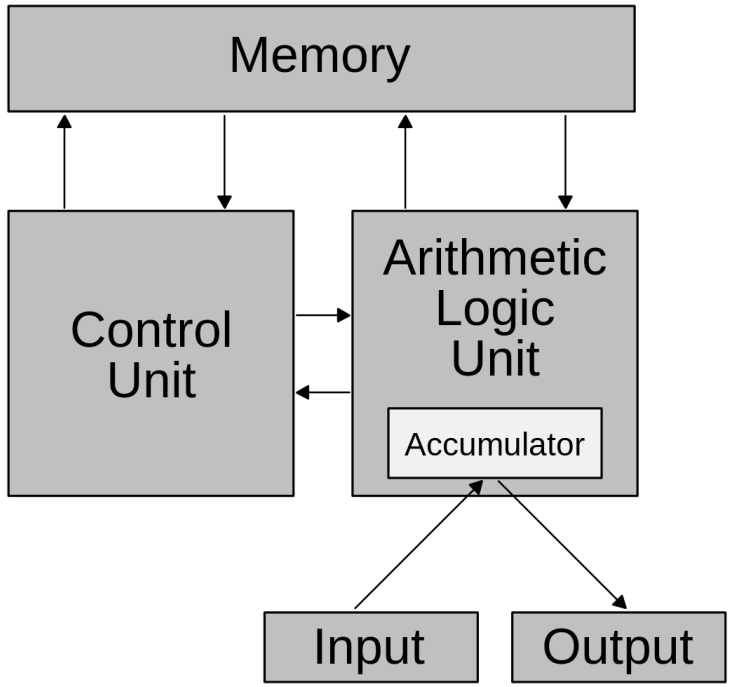
- technická část počítače, zařízení z nichž je počítač sestaven,
- např. klávesnice, monitor, pevný disk, paměť, procesor, sběrnice ...

Software (SW)

- programové vybavení počítače, „oživuje“ hardware,
- mnoho úrovní software:
  - ovladače zařízení,
  - jádro operačního systému,
  - moduly operačního systému (správa uživatelů, správa disků atd.),
  - grafické uživatelské rozhraní,
  - uživatelské aplikace.

Základní struktura počítače

- vstup,
- výstup,
- paměť,
- řadič a
- aritmeticko logická jednotka (ALU).



Poznámky

- O takto koncipovaném počítačí říkáme, že má **von Neumannovskou architekturu**.
- Řadič a ALU tvoří dohromady to, čemu se běžně říká procesor.

Základní struktura počítače

Vstup

- zařízení pomocí kterého se dostávají informace do počítače,
- klávesnice, myš, scanner, počítačová síť ...

Výstup

- předává informace z počítače jeho periferiím – výsledky výpočtů a řídící informace.

Aritmeticko logická jednotka (ALU)

- centrální výkonná část počítače,
- provádí veškeré výpočty, logické operace atd.
- Charles Babbage nazýval ALU slovem „mlýnice“

Řadič

- řídící část počítače,
- řídí a koordinuje ostatní části počítače.

Základní struktura počítače

Primární paměť

- vnitřní paměť počítače,
- Random Access Memory (RAM),
- velice rychlá, relativně malá kapacita,
- obsahuje zpracovávané programy, jejich data,
- data ze vstupu,
- data čekající na výstup.

Sekundární paměť

- vnější paměti počítače,
- relativně pomalé, velká kapacita,
- obsahuje všechny programy počítače,
- data pro tyto programy.

Program versus proces

Program

- posloupnost instrukcí které řídí počítač při zpracování dat,
- **statický**.

Proces

- program, který je v daném okamžiku již vykonáván počítačem,
- **dynamický**

Operační systém (OS)

- první počítače OS postrádaly, procesy si zajišťovaly vše samy,
- snaha usnadnit a sjednotit práci s počítačem, periferiemi,
- poskytnout podporu pro běžně prováděné činnosti – čtení dat ze vstupu, tisk na tiskárně atd.,
- zvýšit výkon počítače,
- jednoúlohový vs. víceúlohový OS,
- jednouživatelský vs. víceuživatelský OS,
- dnešní OS: UNIX/Linux, Windows, Mac OS X Mountain Lion atd.,
- Pozor! Grafické uživatelské rozhraní (GUI), které vidíte na obrazovce je jen „špičkou“ ledovce. OS je rozsáhlá infrasturktura skrytá za tímto rozhranním.

Osobní počítače

- v průběhu 70. let 20. století používá Apple pro své počítače označení osobní,
- v roce 1981 uvedlo IBM osobní počítač, PC s procesorem Intel 8088,
- IBM zvolilo cestu otevřeného standardu, podíleli se různí výrobci HW a SW.
- typicky procesory Intel či AMD, operační systém zprvu DOS, následně Windows a Linux.

Programovací jazyk

- prostředek pro zápis algoritmů ve formě použitelné na počítači,
- jde o formální jazyk – formálně tj. matematicky popsaná syntaxe a sémantika jazyka,
- jinak řečeno jde o jazyk, který má svou gramatiku, slova, věty, věty mají svůj význam,
- různé programovací jazyky jsou určeny pro různé účely.

Programovací jazyky

Nižší programovací jazyky

- jazyk plně přizpůsobený počítači,
- jazyk symbolických adres, strojový kód, částečně VHDL.

Vyšší programovací jazyky

- jazyk lépe pochopitelnější pro člověka, vyšší míra abstrakce,
- syntaxe podobná primitivní angličtině, užití např. běžných značek pro matematické operace,
- jednomu příkazu ve vyšším jazyce odpovídá obecně mnoho instrukcí ve strojovém jazyce,
- většina současně používaných jazyků.

Nižší programovací jazyky

Strojový kód

- jediný jazyk, kterému počítač (procesor) rozumí,
- sekvence 0 a 1,
- strojově závislý,
- pro člověka „nestravitelný“.

Příklad

```
0010 1101
1101 1000
1001 1000
0001 0010
0010 0000
```

Nižší programovací jazyky (pokrač.)

Jazyk symbolických adres

- nesprávně česky nazýván „assembler“, assembler – překladač do strojového kódu,
- assembly language,
- anglické zkratky názvů instrukcí,
- pro člověka „stravitelnější“,

Příklad

```
MOV reg0, Vyska
MOV reg1, Sirka
MUL reg0, reg1
MOV reg0, Plocha
```

Kompilované programovací jazyky

- program je nejprve kompletně přeložen překladačem do strojového kódu počítače,
- teprve po bezchybném překladu je možné přeložený tvar programu spustit,
- výhody:
  - výsledný spustitelný tvar programu může být přesně optimalizován na konkrétní HW a OS,
  - větší rychlost běhu takového programu, lepší využití prostředků HW.
- nevýhody:
  - závislost na konkrétním HW a OS,
  - pro každý HW a OS nutno zvlášť přeložit,
  - menší pružnost při odstraňování chyb v programech, nelze snadno měnit kód programu za jeho běhu.
- typické překládané jazyky – C, C++, Pascal (Delphi), Fortran.

Poznámky

Kompilované programovací jazyky (pokrač.)

- překladač se někdy označuje jako kompilátor (angl. compiler) a překlad jako kompilování (angl. to compile),
- překladač je program, který překládá program z jednoho jazyka do jiného jazyka, typicky například z C++ do strojového kódu procesoru,
- překladač se při překladuz řídí gramatikou zdrojového jazyka, de-facto provádí větný rozbor, tak jak jej znáte z výuky češtiny,
- překladače nemusí překládat jen „programy“, například tato prezentace je napsána v jazyku LaTeX a překladem textu v tomto počítačovém jazyce získáme PDF dokument, který právě čtete.

Interpretované programovací jazyky – klasický přístup

- speciální prostředí pro běh programu,
- toto prostředí čte zdrojový kód programu a jakmile načte celý příkaz, okamžitě jej vykoná a pokračuje na další příkaz,
- výhody:
  - nezávislost programů na HW a OS, závislý je pouze interpret,
  - dynamičtější manipulace s programem, např. za běhu programu lze opravovat zdrojový kód, protože tento kód je interpretem opakovaně načítán.
- nevýhody:
  - obtížná optimalizace pro daný HW a OS,
  - horší využití prostředků HW a OS (na počítači musí běžet i interpret).
- typický interpretovaný jazyk je BASIC ve své klasické podobě, například na starých 8-bitových počítačích
- pozor – jazyk Visual Basic .NET nebo Java jsou uspořádány jinak!

Interpretované programovací jazyky – virtuální stroj

- členění na kompilované a interpretované není absolutní, tyto přístupy lze kombinovat,
- jazyk Java a jeho Java Virtual Machine (JVM):
  - JVM představuje virtuální procesor se svým specifickým strojovým jazykem,
  - zdrojový kód v jazyku Java je kompilován do strojového kódu JVM (výsledkem jsou class a jar soubory),
  - při spuštění JVM začne jednotlivé instrukce uložené např. v jar souboru interpretovat.
- obdobně se chová i Visual Basic .NET či Python,
- Just in Time (JIT) kompilace:
  - zdrojový kód programu je teprve při spuštění kompilován do strojového kódu počítače a okamžitě spuštěn,
  - přeložený tvar programu se nikde neukládá, program je vždy kompilován znovu.

Vyšší programovací jazyky

Lze dělit podle **programovacího paradigmatu**, čili stylu programování, na:

Procedurální jazyky tj. imperativní jazyky

- procedurální paradigma – program popisuje krok za krokem co se má provést,
- strukturované – C, BASIC,
- objektově orientované – Smalltalk, Java.

Neprocedurální jazyky tj. deklarativní jazyky

- neprocedurální paradigmata – popisujeme spíše výsledek, ne jak ho dostáhnout,
- funkcionální – Lisp, Haskell, Scheme,
- logické – Prolog.

Příklady vyšších programovacích jazyků

- FORTRAN (FORmula TRANslation) – první vyšší jazyk z roku 1957, vývoj trval tři roky, dodnes používán pro numerické výpočty,
- COBOL (COmmon Business-Oriented Language) – 1959, jazyk pro „hromadné zpracování dat“,
- Pascal – 1970, strukturované programování, výuka programování.
- Basic (Beginner's All-purpose Symbolic Instruction Code) – 1963, nyní převážně MS Visual Basic

Příklad

Plocha = Vyska * Sirka

Co je C++?

C++ je:

- vyšší programovací jazyk,
- určen pro všeobecné použití,
- určen pro profesionální programátory,
- zachovává zpětnou kompatibilitu s jazykem C.
- C++ je jazyk typicky překládaný do strojového kódu.

C++ dnes

V dnešní době plní jazyk C++ úlohu moderního assembleru.

Historie jazyka C

- autor – Dennis Ritchie, Bell Laboratories, v letech 1968 až 1973,
- vychází z jazyků BCPL a B,
- silná vazba na OS UNIX, jádro OS UNIX práno v C,
- Brian Kernighan a Dennis Ritchie vydávají v roce 1978 knihu *The C Programming Language*, jazyk označováno jako „K & R C“
- 1989 – ANSI standard
- 1990 – ANSI a ISO norma ANSI/ISO 9899: 1990
- 2000 – C99, ISO 9899:1999

Jazyk C++

- 1979 – Bjarne Stroustrup (Bell Laboratories) vyvíjí jazyk „C with classes“
- 1983 – název změněn na C++
- 1985 – vychází kniha *The C++ Programming Language*, zatím se nejedná o standard,
- 1998 – norma ISO/IEC 14882:1998
- 2011 – norma ISO/IEC 14882:2011, 12. srpna 2011,
- 2014 – současný standard označovaný C++14, podporovaný gcc, VS2017 atd.
- 2017 – připravovaný standard, aktuálně ve formě draftu.
- rozšíření jazyka C o OOP, plus další úpravy,
- jazyk není čistě objektový.

Vznik spustitelného programu – základní pojmy

- **zdrojový kód** – napsán programátorem, textový soubor obsahující konstrukce C++ o kterých programátor doufá, že budou vykonávat to co požaduje.
- **hlavičkový soubor** – deklarace funkcí, tříd, konstant atd. Buď napsán programátorem nebo dodán s překladačem.
- **projekt** – soubor popisující, ze kterých zdrojových kódů a knihoven vznikne výsledný program.
- **kód s relativními adresami** – anglicky object file, zdrojové kódy jsou přeloženy do instrukcí procesoru, ale nejsou dořešeny všechny vazby.
- **spustitelný kód** – soubor s binárním zápisem instrukcí, přímo vykonavatelný procesorem.

První program

Hello World

```
1 #include <iostream>
2
3 using namespace std;
4
5 // This is single line comment.
6 void main()
7 {
8    cout << "Hello World" << endl;
9 }
```

Čeština ve zdrojovém kódu

Nedoporučuji psát do C++ cokoliv česky s diakritikou, ani komentáře.

Vznik spustitelného programu

Vznik spustitelného programu ze zdrojového kódu prochází těmito fázemi:

1. **Edit** – editace zdrojového kódu,
2. **Preprocess** – rozvinutí maker, vložení hlavičkových souborů,
3. **Compile** – vlastní překlad, výsledkem je modul s relativními adresami nebo alternativně program v assembleru,
4. **Link** – sestavení výsledného programu, vstupem jsou moduly s relativními adresami, plus knihovny; výsledkem spustitelný program,
5. **Load** – OS zavede spustitelný kód do primární paměti,
6. **Execute** – vlastní vykonávání programu.

Vznik spustitelného programu

Ke vzniku spustitelného programu se využívají následující programy:

1. **Edit** – textový editor,
2. **Preprocess** – preprocesor,
3. **Compile** – překladač, kompilátor,
4. **Link** – linker (sestavovač, spojovač?),
5. **Load** – operační systém,
6. **Execute** – operační systém.

Ladění

- Proces odstraňování chyb v programu se nazývá **ladění**.
- K ladění slouží program zvaný **debugger**, který
  - umožňuje sledovat vykonávání programu,
  - vypisovat hodnoty proměnných,
  - zobrazuje zásobník volání funkcí,
  - umožňuje nastavovat tzv. breakpointy, tj. místa, kde se má přerušit vykonávání programu
  - krokování programu po jednom příkazu.

Základní stavební bloky C++

Data – údaje, které bude program zpracovávat,

Kód – instrukce programu prostřednictvím jsou zapisovány algoritmy,

Komentáře – pomocné informace pro programátora umístěné ve zdrojovém kódu.

Komentáře v C++

Existují dva druhy komentářů:

- jednořádkový – začíná sekvencí `//` a končí s koncem řádku
- víceřádkový – omezen sekvencemi `/*` a `*/`

```
1 // This is single line comment.
2 void main()
3 {
4    cout << "Hello World" << endl; // Single line comment.
5    /*
6    Multi line comment
7    suitable for long texts.
8    */
9 }
```

Komentáře v C++ (pokrač.)

Komentáře

- při překladu jsou kompilátorem ignorovány,
- určeny pouze pro programátora,
- zvyšují čitelnost kódu,
- slouží k zaznamenání informací, které nejsou přímo patrné ze zdrojového kódu – autorství, verze, datum revize atd.
- tzv. dokumentační komentáře lze využít pro zápis programátorské dokumentace přímo k příslušným částem kódu, generování dokumentace externím SW (např. Doxygen),
- platí zásada, že **všeho moc škodí**,
- jakmile je nutno v komentáři „dovysvětlit“ nesrozumitelný úsek kódu, je tento kód špatný!
- stejně tak je zcela nevhodné komentovat každý řádek kódu!

Datové typy v C++

Datový typ definuje dvě vlastnosti:

- rozsah povolených dat, čili jaké hodnoty jsou povolené a jaké ne,
- povolené operace, čili co s těmi hodnotami lze provádět.

Berte je vážně!

Datové typy jsou jedním z prostředků pro psaní kvalitních programů.

Datové typy v C++

Datový typy v C++ dělíme na:

- základní
  - celočíselné
  - s plovoucí řádovou čárkou
  - logické
- adresní
  - ukazatelé
  - reference
- strukturované
  - pole
  - struktura
  - třída

Celočíselné datové typy v C++

- `char` – jeden byte
- `short` – větší nebo roven `char`
- `int` – větší nebo roven `short`
- `long` – větší nebo roven `int`
- `long long` – větší nebo roven `long`

Všechny typy jsou automaticky se znaménkem, neznaménková verze se uvádí klíčovým slovem `unsigned`, např. `unsigned int`.

Obvyklé rozsahy celočíselných datových typů

Typ	Počet bitů	Rozsah hodnot
`char`	8	$2^7 \ldots 2^7 - 1$
`short`	16	$-2^{15} \ldots 2^{15} - 1$
`int`	32	$-2^{31} \ldots 2^{31} - 1$
`long`	32	$-2^{31} \ldots 2^{31} - 1$
`long long`	64	$-2^{63} \ldots 2^{63} - 1$

Celočíselné datové typy v C++ (pokrač.)

Číselné soustavy používané v C++

- desítková – základ soustavy je číslo 10, číslice 0 až 9, např. 158
- osmičková – základ soustavy je číslo 8, číslice 0 až 7, např. 0236
- šesnáctková – základ soustavy je číslo 16, číslice 0 až 9, písmena A až F, kde A je deset, B jedenáct atd., např. 0x9E

Celočíselné datové typy v C++ (pokrač.)

Poznámky

- Nejběžnější se pochopitelně desítková soustava.
- Binární soustava se nepoužívá ve zdrojovém kódu přímo nepoužívá (dlouhý zápis čísel).
- Osmičková soustava je spíše historický přežitek.
- Šestnáctková soustava se používá pro manipulaci s bity – vždy 4 bity na jednu šestnáctkovou číslici.
- Šestnáctková soustava se také často nazývá jako **hexadecimální** soustava.

Číselné soustavy – příklad

Číslo 158 můžeme vyjádřit v jednotlivých číselných soustavách takto:

- binárně

$$10011110_2 = 1\times2^7+0\times2^6+0\times2^5+1\times2^4+1\times2^3+1\times2^2+1\times2^1+0\times2^0$$

- osmičkově

$$236_8 = 2 \times 8^2 + 3 \times 8^1 + 6 \times 8^0$$

- desítkově

$$158_{10} = 1 \times 10^2 + 5 \times 10^1 + 8 \times 10^0$$

- hexadecimálně

$$9E_{16} = 9 \times 16^1 + 14 \times 16^0$$

Datové typy s plovoucí řádovou čárkou v C++

Datový typ	Počet bitů	Rozsah hodnot	Platné číslice
`float`	32	$\pm 3,4 \times 10^{\pm 38}$	7
`double`	64	$\pm 1,7 \times 10^{\pm 308}$	15
`long double`	ve Visual Studio 2013 shodné s `double`		

Reálná čísla vs. čísla s plovoucí řádovou čárkou

Čísla s plovoucí řádovou čárkou nejsou totožná s reálnými čísly z matematiky – problém s periodickými čísly, iracionálními čísly!

Logický datový typ v C++

- jazyk C++ zavádí pro logické hodnoty datový typ `bool`,
- konstanty `true` pravda a `false` nepravda,
- platí že `true` $\neq$ `false`,
- neexistuje žádná souvislost mezi `bool` a `int`!

Logické hodnoty v ANSI C

Jazyk ANSI C používá pro uložení logických hodnot typ `int` včetně svérázné definice hodnot true a false. V jazyce C++ důrazně doporučuji nepoužívat tento zastaralý přístup a doporučuji používat typ `bool`.

Logický datový typ v C++

Platné vztahy

$$\texttt{true} \neq \texttt{false}$$

$$\neg\texttt{true} = \texttt{false}$$

$$\neg\texttt{false} = \texttt{true}$$

Proměné

Data v programu jsou reprezentována pomocí proměnných.

Proměnná

- identifikátor,
- datový typ,
- hodnota.

Identifikátor

- obecný název pro pojmenování čehokoliv v programu co lze pojmenovat,
- konečná neprázdná posloupnost písmen, číslic, znaků podtržítko, která nesmí začínat číslicí,
- rozlišují se malá a velká písmena.
- délka identifikátoru není teoreticky omezena

Příklad

```
a A a1 ab _a _a1 a_b VeryLongIdentifier
```

Jak správně volit identifikátory?

- Buď česky nebo anglicky, ne mixovat! `NextMove` nebo `DalsiTah` ne `NextTah`.
- Raději anglicky.
- Identifikátor musí popisovat co obsahuje! `NumberOfStudents` nebo `PocetStudentu` ne `A28`.

Děkuji za pozornost