PROGRAMOVANIE

PROGRAMOVANIE
Poznámky z prednášok 2014

Miroslav Biňas
Technická univerzita Košice, Slovensko

WILEY-INTERSCIENCE

A JOHN WILEY & SONS, INC., PUBLICATION

Obsah

CHAPTER 1

INTRODUCTION TO C PROGRAMMING

1.1 Úvod

- Predmet sa volá *Programovanie* a naším cieľom je z vás vychovať programátorov. Slovo *programovanie* vyjadruje činnosť a preto aj tento kurz bude veľmi praktický.
- Motivačne: na konci kurzu sa budete môcť pochváliť svojou (možno) prvou hrou, ktorú vytvoríte.
- Ak hľadáte literatúru pre tento predmet, môžete použiť titul *Učebnice jazyka C* (`http://www.martinus.sk/?uItem=74741`) od *Pavla Herouta*.
- Stránka celého predmetu sa nachádza na adrese `http://it4kt.cnl.sk/c/pvjc` - tu nájdete cvičenia, prednášky, príklady použité na prednáškach a aj všetko potrebné pre úspešné absolvovanie tohto kurzu.
- Osnova kurzu a prehľad všetkých tém sa nachádza na podstránke osnovy kurzu. Tento rok je tento kurz v špeciálnom režime, takže je možné je sa podoba bude "kozmeticky" upravovať počas behu.

- Za predmet môžete získať spolu *40* bodov, ktoré pozostávajú zo štyroch zadaní po *10* bodoch.
- Ak ste opakujúci, môžete si nechať uznať zápočet z minulého roku v jeho minimálnej podobe - *21* bodov. Odporúčame ale ho absolvovať znova (väčšia šanca predmet úspešne absolvovať).
- Pri absolvovaní predmetu budeme dbať na to, aby ste postupovali podľa etického kódexu a to hlavne pri práci na vašich zadaniach. Včas vás naň budeme opätovne upozorňovať.

1.2 Hello world!

- Začneme príkladom, ktorý sa používa už od roku 1974 a bol adaptovaný do mnohých podôb, resp. mnohých programovacích jazykov. V roku 1974 to bol *Brian Kernighan: Programming in C: A Tutorial* (`http://cm.bell-labs.com/cm/cs/who/dmr/ctut.pdf`). Tento príklad vypíše na obrazovku len text: *"Hello world!"* a slúži ako ukážka najjednoduchšieho programu v danom jazyku (alebo na overenie potrebného softvérového vybania pri programovaní). (zdroj: *Wikipedia: Hello world program*)
- Na vypísanie textu na obrazovku potrebujeme poznať funkciu, ktorá vypísanie textu na obrazovku zabezpečí.
- Kombinácia `\n` predstavuje nový riadok (escape character).
- To však nestačí - jazyk C nie je skriptovací a jeho príkazy sa nevykonávajú v poradí zhora nadol - od začiatku súboru po jeho koniec. Každý program v jazyku C obsahuje funkciu `main()`, ktorá označuje štartovacie miesto programu.
- Ak by sme sa pokúsili program spustiť v tomto momente, neuspeli by sme. Jazyk C totiž funkciu `printf()` nepozná a musíme mu povedať, kde sa jej definícia nachádza - musí sa ju "naučiť". Na to slúži `#include <stdio.h>`
- Výsledná podoba tohto programu bude teda vyzerať takto (hello.c):

Algorithm 1.1

```
#include <stdio.h>

int main(){
   printf("Hello world!\n");
}
```

1.3 Preklad a spustenie programu

- Práve sme napísali *zdrojový kód* v jazyku C. Aby sme však "ochutnali" výsledok, musíme zo zdrojového kódu vytvoriť spustiteľný kód procesom *prekladu* (kompilácie).
- Prekladač je nástroj, ktorý zo *zdrojového kódu* vytvorí *objektový kód*. Objektovému kódu rozumie samotný procesor a je možné ho už vykonať (spustiť).
- Pre preklad budeme používať prekladač *GNU C Compiler* (`gcc`) s nasledovnými parametrami:
  - `-std=gnu99` - budeme používať verziu jazyka C z roku 99
  - `-Wall` - zaujímať nás budú všetky upozornenia (warning), pretože nechceme, aby ste písali "len"kód, ktorý ide, ale aby ste písali "dobrý kódä prekladač vás na nič špeciálne nemusel upozorňovať (program vie fungovať aj napriek tomu, že vás prekladač niekoľkokrát upozornil)
  - `-Werror` - aby ste tieto upozrnenia neignorovali, táto voľba ich všetky zmení na chyby, čím nedôjde ku vytvoreniu výsledného spustiteľného programu
- Program môžete prekladať aj použitím nástroja `make`. Predvolené správanie však funguje bez použitia extra prepínačov. Ak chcete dosiahnuť správne správanie, potrebujete nastaviť minimálne premennú prostredia `CFLAGS` (v súbore `/.bashrc`):

```
export CFLAGS="-std=gnu99 -Wall -Werror -lm"
```

- Program následne spustíte napísaním príkazu `./hello` z príkazového riadku. Ak budete používať vývojové prostredie, budete mať možnosť spustiť výslednú aplikáciu priamo z neho.

1.4 How old are you?

- Vytvoríme program, pomocou ktorého sa používateľa opýtame, koľko má rokov a následne ho "pochválime";) Vytvoríme kostru:

```
#include <stdio.h>

int main(){
}
```

- Na to, aby sme tento program vedeli vytvoriť potrebujeme poznať dve veci:
  - ako prečítať hodnotu od používateľa, a
  - ako si zapamätať a kam uložiť hodnotu, ktorú používateľ zadal.

- Na načítanie vstupu od používateľa budeme používať funkciu `scanf()`.
- Na zapamätanie a uloženie načítanej hodnoty budeme používať *premenné*.
- *Premenná* je miesto v pamäti pre uloženie hodnoty pre neskoršie použitie; alebo je to pomenované miesto v pamäti.
- Ak chceme pracovať s pemennou, musíme ju *deklarovať*. "Systém"totiž musí vedieť, koľko pamäte má pre túto premennú vyhradiť. Množsto pamäte je určené *typom* premennej.
- Pri deklarácii určujeme názov premennej a typ údajov, ktoré sa budú v premennej nachádzať. Názov by mal odrážať povahu údajov, ktoré bude premenná uchovávať. Typ hovorí aj o tom, koľko miesta v pamäti bude pre tento údaj vyhradený.
- Základné typy, s ktorými budeme pracovať sú
  - `int` - celé čísla
  - `float` - reálne čísla
  - `double` - reálne čísla s dvojnásobnou presnosťou
  - `char` - znaky (jazyk C nemá typ na uchovávanie reťazcov)
- Znaky sú v pamäti reprezentované tiež ako čísla. Ak sa na tieto čísla chceme pozerať ako na znaky, musíme vedieť, ktoré číslo predstavuje ktorý znak. Pre toto mapovanie vznikla tabuľka *ASCII*, ktorá mapuje čísla na znaky.
- Deklarovanie premennej a načítanie vstupu od používateľa:

```
ing age;
scanf("%d", &age);
```

- Funkcia `scanf()` má dva parametre: formátovací reťazec, ktorý hovorí o tom, aký typ údajov má byť prečítaný; a adresu premennej, do ktorej má byť táto informácia zapísaná.
- Vypísanie informácie na obrazovku:

```
printf("Well␣done,␣you␣are␣age␣years␣old\n");
```

- Očakávanie, že ku premennej pristupujeme priamo pomocou jej názvu aj pri výpise. Funkcia `printf()` obsahuje formátovací reťazec, pomocou ktorého vieme povedať, že sa má na konkrétnom mieste vypísať hodnota takéhoto typu:

```
printf("Well␣done,␣you␣are␣%d␣years␣old\n", age);
```

  Nenachádza sa tu už znak `'&'` - pracujeme priamo s hodnotou.
- Výsledný kód (age1.c):

```
#include <stdio.h>

int main(){
    int age;
    printf("How old are you? ");
    scanf("%d", &age);
    printf("Good boy, you are %d years old\n", age);
}
```

1.5 Rozsah platnosti premennej

- Rozšírime program tak, aby nebolo možné zadávať iné hodnoty ako tie, ktoré sú v rozsahu $(0, 120)$ - používateľ bude hodnoty zadávať dovtedy, pokiaľ nezadá správnu hodnotu.

- Vytvoriť takýto kód (chybný kód) (age2.c):

```
#include <stdio.h>

int main(){
    do{
        int age;
        printf("How old are you? ");
        scanf("%d", &age);
    }while(age <= 0 || age > 120);
    printf("Good boy, you are %d years old\n", age);
}
```

- *Bug* - chyba v programe. *In 1946, when Hopper was released from active duty, she joined the Harvard Faculty at the Computation Laboratory where she continued her work on the Mark II and Mark III. Operators traced an error in the Mark II to a moth trapped in a relay, coining the term bug. This bug was carefully removed and taped to the log book. Stemming from the first bug, today we call errors or glitch's [sic] in a program a bug.* (zdroj: wikipedia: software bug (`http://en.wikipedia.org/wiki/Software_bug`))

- Prekladač hlási, že premenná `age` nie je deklarovaná. Prečo, keď je deklarovaná na riadku 5?

- Rozsah platnosti premennej (scope) - premenná platí od jej deklarácie po najbližšie uzatváracie zložené zátvorky "jej úrovne". Vhodným presunutím deklarácie premennej `age` je možné jej rozsah platnosti rozšíriť:

  (age3.c):

```c
#include <stdio.h>

int main(){
    int age;

    do{
        printf("How old are you? ");
        scanf("%d", &age);
    }while(age <= 0 || age > 120);

    printf("Good boy, you are %d years old\n", age);
}
```

- *Globálna premenná* - platí v celom programe. Umiestňuje sa mimo (nad) definíciu funkcie `main()`. Nepoužívať! Používanie globálnych premenných vedie k zlým návykom! (age4.c):

```c
#include <stdio.h>

int age;

int main(){
    do{
        printf("How old are you? ");
        scanf("%d", &age);
    }while(age <= 0 || age > 120);

    printf("Good boy, you are %d years old\n", age);
}
```

1.6 Unit Converter

- Vytvoríme program na prevod teploty zo stupňov celzia do kelvinov a fahrnheitov. Vzorce na prevod:
  - `kelvin = celsius + 273.15;`
  - `fahrnheit = 9/5 * celsius + 32;`
- Presnosť výpočtu závisí od presnosti čiastkových výsledkov: aký bude výsledok operácie $9/5$? Je potrebné s ním pracovať ako s typom `float` - $9.0/5.0$.
- Formátovanie výsledku na dve desatinné miesta pomocou zápisu `%.2f`.
- Výsledný program (converter.c):

```
#include <stdio.h>

int main(){
    printf("Enter the temperature in Celsius: ");
    float celsius;
    scanf( "%f", &celsius );

    float fahrnheit = 9.0/5.0 * celsius + 32;
    float kelvin = celsius + 273.15;

    printf("%.2f in Celsius is %.2f in Fahrnheit and %.2f in Kelvin\n", celsius , fah
}
```

1.7 Presnosť desatinných čísiel

- Vytvorme jednoduchý program (float1.c):

```
#include <stdio.h>

int main(){
    float f = 1/10;
    printf("%.2f\n", f);
}
```

- Po preklade a spustení je však miesto výsledku 0.10 výsledkom 0.00.
- Problém je, že delíme jeden integer druhým. Výsledkom operácie podielu dvoch integerov je ďalší integer.
- Riešením problému je spraviť z typu `int` typ `float` (float2.c):

```
#include <stdio.h>

int main(){
    float f = 1.0/10.0;
    printf("%.2f\n", f);
}
```

- Rovnako tak môžeme explicitne pretypovať typ `int` na typ `float` pomocou *cast operátora* (float3.c):

```
#include <stdio.h>

int main(){
    float f = (float)1/(float)10;
    printf("%.2f\n", f);
}
```

- Pokiaľ budete zväčšovať počet čísiel nachádzajúcich sa za desatinnou čiarkou, výsledkom už nebude hodnota 0.1, ale na konci budú pribúdať rozličné iné hodnoty. Je to dané tým, že typ `float` má konečnú veľkosť (je v pamäti reprezentovaný konkrétnym počtom bytov). To znamená, že v istom momente môžu byť výsledky vašich výpočtov nepresné.
- Presnosť vie narobiť rozličné problémy - viď toto video (`http://www.youtube.com/watch?v=EMVBLg2MrLs`).