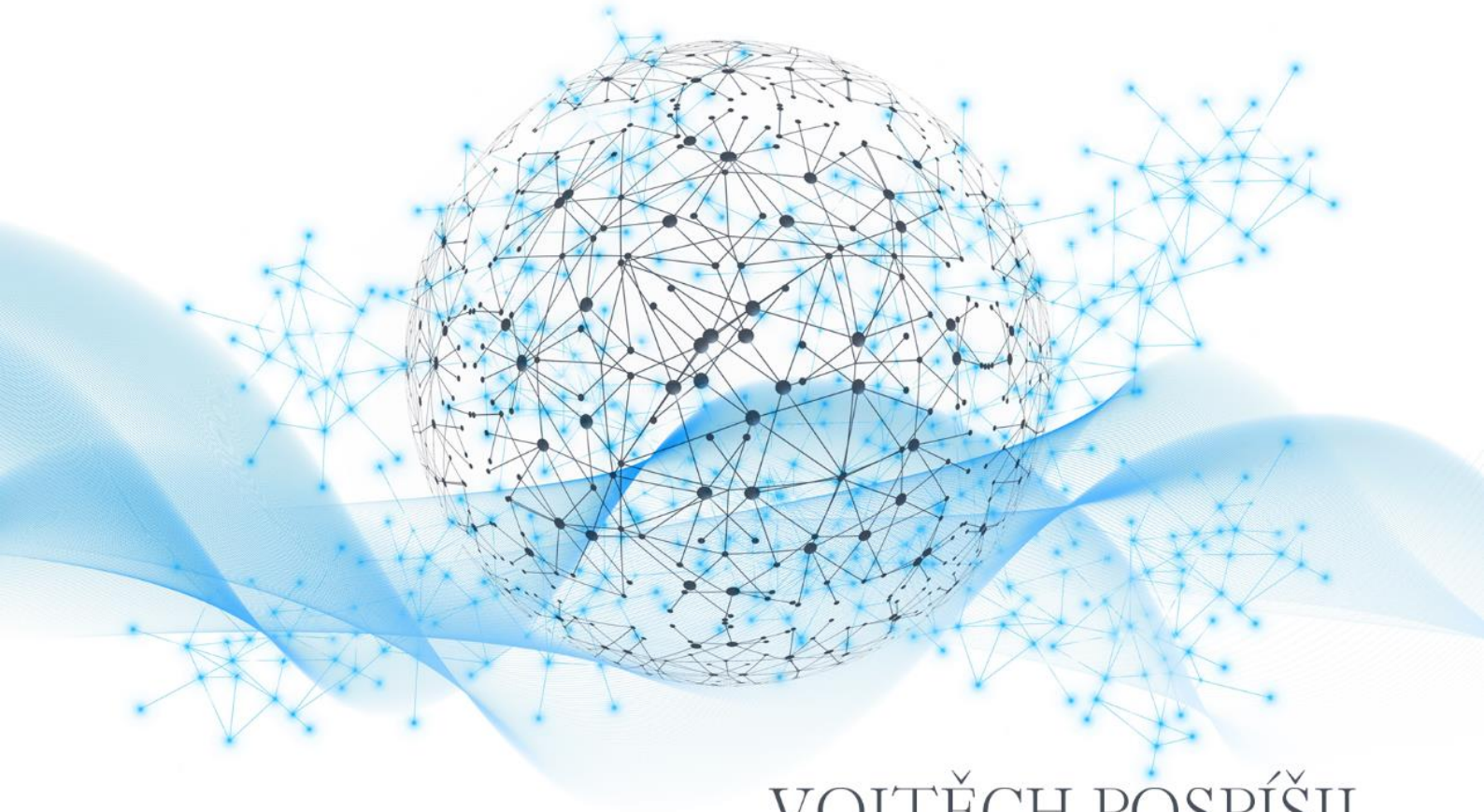
GOTO

aneb

OSAHÁVÁNÍ JAZYKA C



VOJTĚCH POSPÍŠIL

GOTO

aneb

Osahávání jazyka C

Poznejte podstatu programu a důležité okamžiky jeho tvorby:

· modularita programu a interface

· datové vstupy a klávesnicový buffer

· příkaz GOTO a jeho náhražky, řídící struktury

· datové struktury a operační paměť

· pointery - cesta k hardwarovým adresám paměti

· bytové a bitové programování

· odolnost programu

· ladění programu - hledání chyb

. vícevláknové programování na vícejádrových procesorech

C jazyk patří mezi čtyři světově nejpopulárnější programovací jazyky. A to spolu se svými dvěma potomky, jazyky C++ a C#. A se svým příbuzným, jazykem Java. C jazyk je jedničkou mezi jazyky první volby, tedy těmi, které se programátoři učí jako první. Ať už je to díky školním osnovám nebo v C jazyce napsaným systémovým programům. Příkladem jsou operační systémy UNIX, LINUX, ANDROID nebo webový server APACHE. C jazyk má široké spektrum použití. Od programování mikročipů přes mobilní aplikace až k aplikacím desktopovým. Díky vynikajícím kompilátorům a jejich optimalizačním schopnostem dokáže být výsledný program rychlejší než systémové programy napsané v assembleru.

Obsah:

Všech 78 odladěných programových kódů, které jsou součástí této publikace, si stáhněte na následujícím odkazu:

http://eknihyjedou.cz/content/GOTO_kody.zip

Motto: Program je sada proměnných, které mění svůj stav.

Program bez počítače:

Máme tři nekalibrované nádoby o objemech osm, pět a tři litry. Největší z nádob je po okraj naplněná vínem, ostatní jsou prázdné. Rozděl postupným přeléváním osm litrů vína tak, aby v osmilitrové nádobě byly čtyři litry vína a v pětilitrové nádobě rovněž čtyři litry vína.

Programy a počítač:

Kniha obsahuje 78 odladěných programových kódů C jazyka. K dispozici jsou i na datových nosičích.

Kapitola 1: O konci let, kdy srdce bylo osamělý lovec

Milí mladí kolegové,

řečeno s Williamem Wordsworthem: „Pryč je ten čas, kdy kopretinám v trávě plál k slávě třpyt…“ a legendární samotáři psali své originální programy coby pionýři divokého IT světa.

Dnešní programy vznikají „rychle“ a „efektivně“. Jeden program vytváří několik specializovaných programátorů a program se dokonce kompiluje ze zdrojových kódů několika programovacích jazyků.

Jazyk C řeší modularitu na dvou úrovních, na úrovni knihovny a na úrovni funkce. Tento extrémně rychlý a paměťově nenáročný jazyk je založen na pouhých dvaatřiceti slovech, takzvaných klíčových slovech C jazyka. Komu třicet dva slov nestačí, hledá další nástroje v systémových knihovnách, které si přibalí na začátku zdrojového kódu formulkou #include. Knihovny obsahují přidané stavební kameny C jazyka – funkce. Rovněž dobrý programátor rozděluje své programy do modulů – vlastních funkcí.

Uchopit velký program jiným způsobem než rozdělením na moduly snad ani nejde. Jenže co se kvůli přehlednosti rozděluje, to se zase musí kvůli funkčnosti propojit. A jsme u Achillovy paty celého systému, kterou je interface. Interface neboli rozhraní musí zajišťovat rychlou a bezproblémovou komunikaci oddělených částí systému navzájem.

Jak tedy spolu komunikují funkce C jazyka? Každá funkce je vybavena závorkou, do které může ten, kdo tuto funkci žádá o službu, vložit údaje, které má volaná funkce zpracovat. Po zpracování údajů musí volaná funkce předat výsledky zpět funkci volající. Za tím účelem je vybavena kapsou, která má stejné jméno jako funkce.

Představme si, že funkce B provádí výpočet mocniny. Zápis alfa=B(2, 3) kopíruje do lokální proměnné alfa obsažené ve funkci A výsledek umocnění dvě na třetí uložený v kapse funkce B. Kapsu B plní po výpočtu v těle funkce B klíčové slovo ***return***. Tělo funkce se nachází ve složených závorkách za deklarací funkce a před jejím voláním v programu. Ostatní moduly funkci využívají tak, že volají její kapsu, jak je uvedeno výše.

Toto rozhraní má jednu nevýhodu. Každá funkce má na předání dat jen jednu kapsu. Je-li z výpočtu více výstupů, neumí je funkce předat. Musí je proto vložit do globálních proměnných, které jsou deklarovány hned na začátku programu, aby je všechny funkce deklarované až za nimi znaly. A kterýkoliv modul si může z těchto globálních proměnných výsledky zkopírovat. Nevýhodou tohoto řešení je, že globální proměnné blokují místo v paměti po celou dobu běhu programu.

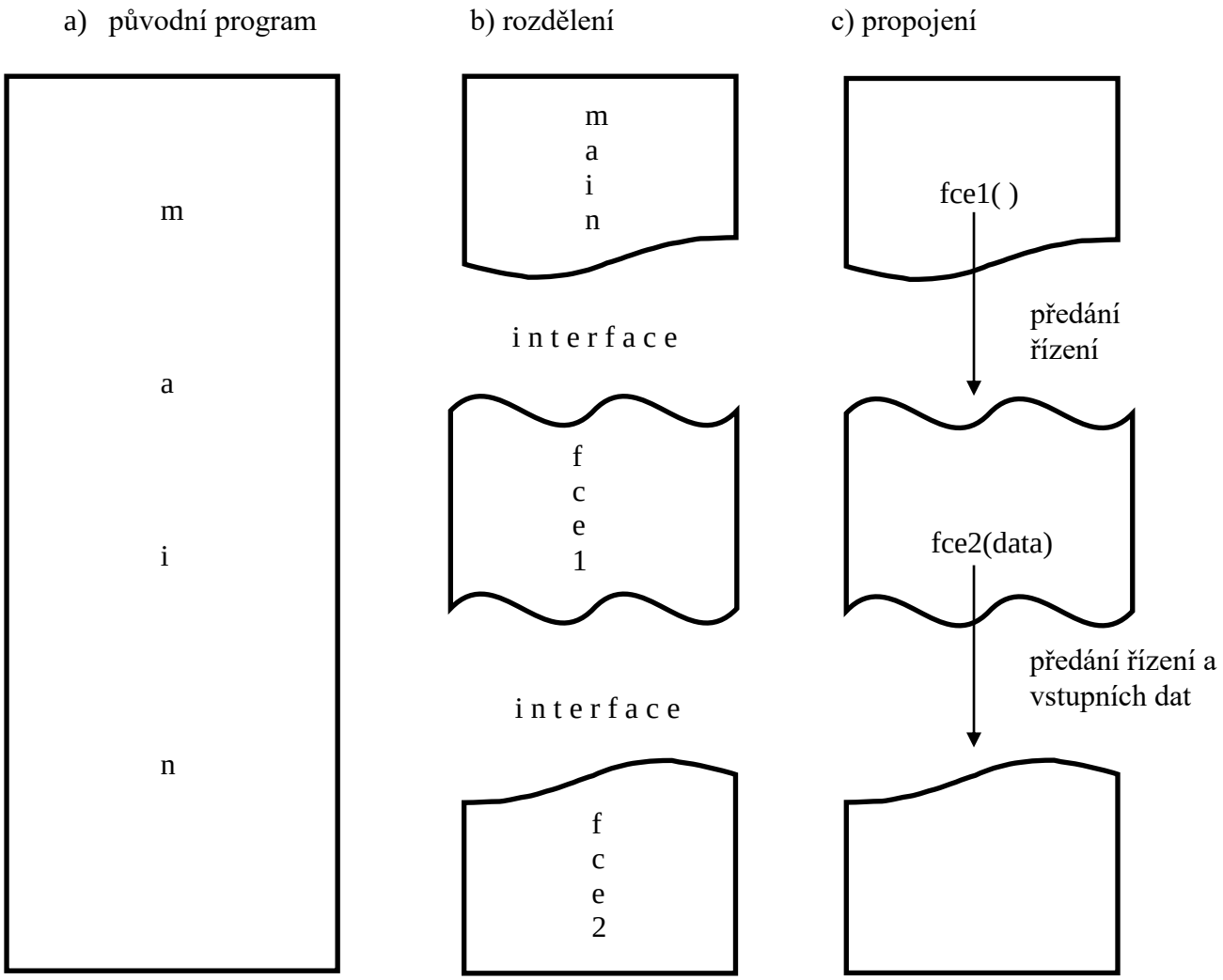
Existuje ještě třetí možnost. Do závorky volané funkce se kromě vstupních dat pro zpracování vloží navíc i hardwarové adresy lokálních proměnných funkce volající. Volaná funkce uloží (nikoliv zkopíruje) řešení přímo do lokálních proměnných volající funkce, i když jmény tyto proměnné nezná (jsou deklarovány jen uvnitř volající funkce). Zná ale jejich hardwarové adresy, které obdržela ve vstupní závorce jako takzvané pointery. Hardwarová adresa se nevytváří deklarací, je součástí paměti RAM již z výroby. Hardwarovou adresu paměťové buňky použité pro určitou proměnnou zjistíme tak, že před jméno proměnné napíšeme znak &. Naopak pokud si chceme adresu někam poznamenat, uložit kopii adresy do proměnné, musíme proměnnou deklarovat se znakem * před jménem. Hvězdička říká, že proměnná obsahuje adresu, nikoliv hodnotu.

Kupříkladu v modulu A máme deklarovánu lokální proměnnou alfa pro ukládání výsledku výpočtu prováděného modulem B. Předáme proto modulu B jak kopie vstupních hodnot pro výpočet (2 a 3), tak i kopii hardwarové adresy proměnné alfa.

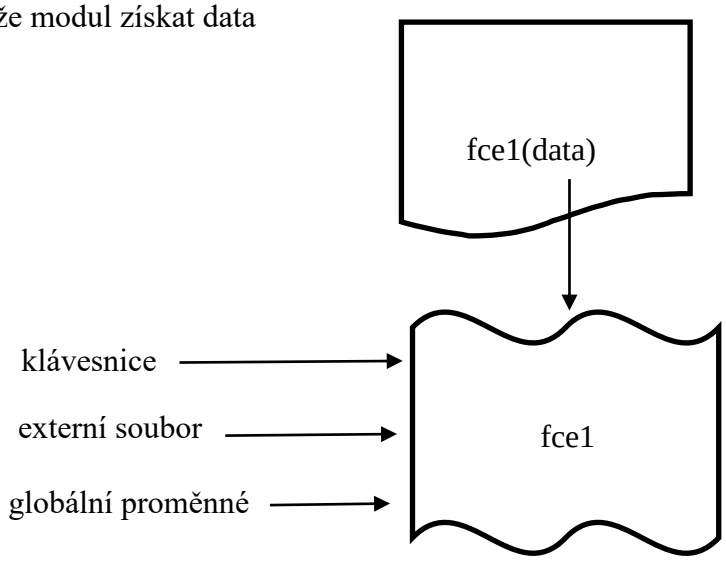
Výpočet voláme takto: B(2, 3, &alfa). Modul B má na příjem kopií dat připraveny tři proměnné: beta1, beta2 a *beta3. Obdrží do nich hodnoty 2, 3 a 8AF4, což je hexadecimální adresa paměťové buňky používané pro proměnnou alfa. Vnitřně se odehrají tato tři přiřazení: beta1=2, beta2=3 a beta3=&alfa. Výsledek výpočtu - číslo 8 - nelze vložit do alfa přiřazením alfa=8, protože modul B jméno alfa nezná. Ale lze jej vložit do alfa přes místní proměnnou beta3 takto: *beta3=8. Tímto přiřazením se číslo 8 vloží do proměnné, jejíž hardwarovou adresu obsahuje proměnná beta3, tedy do proměnné alfa.

Pro úplnost zmíním ještě rozhraní, které nepředává žádná data, ale předává pouze řízení programu. Použiji-li názvosloví z předchozího příkladu, pak modul A předá řízení modulu B takto: B( ) . Pokud by B(něco) byla kapsa funkce, musela by být na pravé straně přiřazovacího příkazu a na jeho levé straně by byla proměnná: alfa=B(něco). Funkce, která nepřebírá žádná data, ale přebírá jen řízení programu, se nazývá procedura.

obr. 1: Rozdělení programu na moduly



obr. 2: Čtyři způsoby, jak může modul získat data



Příklad 1

V roce 1833 navrhl Charles Babbage spolu se sedmnáctiletou Augustou Adou Kingovou, hraběnkou z Lovelace (první programátorkou našeho vesmíru a dcerou slavného romantického básníka lorda Byrona) samočinný počítač, jehož současná replika potvrzuje, „že to funguje“. Vstupními prvky byly klávesnice na naplnění paměti vstupními hodnotami a děrné štítky na řízení běhu programu (převzaté z automatického tkalcovského stavu francouzského vynálezce Josepha Maria Jacquarda). Výstupním zařízením byla tiskárna (knihtisk Johannese Gutenberga existoval už více jak 300 let). Počítač obsahoval adresovatelnou paměť na tisíc čísel (dnešní operační paměť RAM), počítací ústrojí (dnešní ALU) a pracovní registr na jedno číslo (dnešní střadač či aritmetický registr). „Instrukční sada procesoru“ tohoto prvního počítače obsahovala pouhých 7 instrukcí v desítkové soustavě:

1 – plnění střadače obsahem paměti (původní obsah střadače se vymaže)

2 – přičtení obsahu paměti ke střadači

3 – odečtení od střadače

4 – násobení střadače

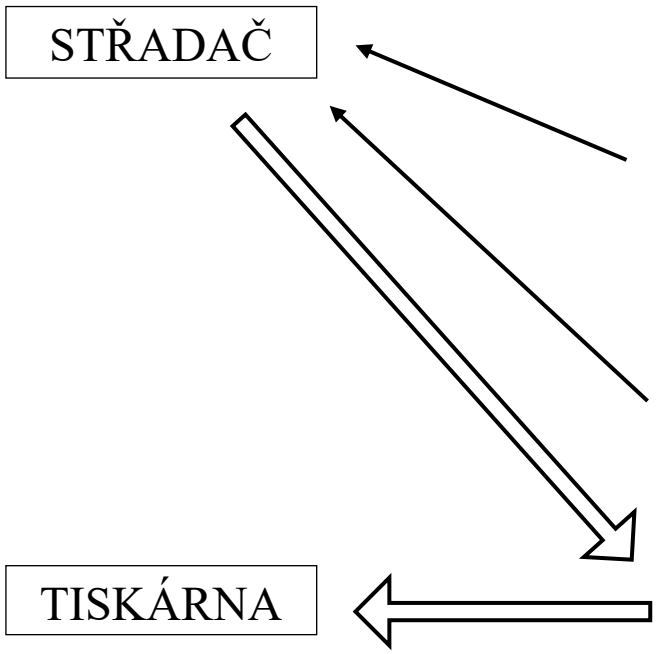
5 – dělení střadače

6 – uložení střadače do paměti

7 – tisk z paměti

Krok 1

Naprogramuj první Babbageův počítač. V editoru Notepad (Poznámkový blok) napiš program na výpočet součtu čísla 7 (uloženého v paměti s adresou 200) a čísla 8 (uloženého v paměti s adresou 150) s následným tiskem výsledku. Každou dvojici čísel (instrukce a adresa) piš na nový řádek.



adresa paměti	obsah paměti
150	8
200	7
?	?

Mezikrok – ukázkový **program Stisk klávesy** (***znakové tabulky v grafickém a řádkovém rozhraní*** *OS Windows 10 a v nestandardních funkcích C jazyka*):

---

1) Program Stisk klávesy zobrazuje ASCII kód právě stisknuté klávesy. Jaký ASCII kód klávesa vysílá, záleží na aktuálně navolené klávesnici (přepínač Windows + mezerník).

2) Ale nejen změna klávesnice mění ASCII kód stisknuté klávesy. Druhým měničem kódu je používaná znaková sada. Ta je implicitně definována operačním systémem a její výběr nemůže uživatel ovlivnit. S jedinou výjimkou popsanou níže.

3) Sada ISO-8859-2, zkráceně Windows Latin 2 a sada Windows 1250, zkráceně CP 1250, kterou používá OS W10 v oknech (v grafickém rozhraní), se liší jen ve znacích š (182/154) - ť (187/157) - ž (190/158). Ale v řádkových příkazech (konzolovém okně) používá OS W10 sadu CP 852, neboli MS DOS Latin 2, a to je úplně jiná znaková sada. Přepnout znakovou sadu konzoly na CP 1250 můžeme dosovským příkazem chcp 1250 a zpět příkazem chcp 852. Nastavení bude platit jen dočasně pro právě aktuálně otevřené konzolové okno.

3) Použitá funkce getche() je z nestandardní knihovny conio.h a bere klávesu okamžitě po stisku, ještě před výstupem na monitor, ze kterého by pak po Entru klávesa putovala do klávesnicového bufferu. Funkce getche() používá znakovou sadu ISO 8859-1, zkráceně Windows LATIN 1, aniž by žádala OS o dovolení. Současně program ukládá klávesu sejmutou funkcí getche() do proměnné znak. Z proměnné znak ji na monitor interpretuje standardní funkce printf() s použitím konzolové sady CP 852.

4) Při psaní programu v C jazyce v libovolném ID prostředí (grafické rozhraní) používá OS znakovou sadu CP 1250. Odladěný a zkompilovaný program běží v konzolovém okně používajícím sadu CP 852. Například ve funkci printf() zapsané 'š' má v CP 1250 ASCII kód 154, ale program v konzolovém okně přiřadí tomuto kódu znak podle CP 852, tedy 'Ü'. Podobně např. v CP 1250 napsané 'ž' má kód 158 a ten se v CP 852 zobrazuje jako '×'. Znak 'á' má kód 225, což je v CP 852 znak 'ß' atd.

5) Při programování v C jazyce vlož na začátek programu příkaz ***system(“chcp 1250“);***

a v konzolovém okně nastav font z rodiny vektorových (truetypových) písmen, nikoliv písmen bitmapových (rastrových). ***Pro nastavení písma ve výstupním okně programu klikni pravým tlačítkem myši na ikonu v levém horním rohu okna, vyber a potvrď levým tlačítkem volbu Vlastnosti, vyber roletu Písmo, vyber velikost 24 a typ písma Consolas.***

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

void main(void)
{
    int znak, i;

    system("chcp 1250"); system("color 9f"); system("mode con lines=29 cols=124");
    printf("1) Program Stisk klávesy zobrazuje ASCII kód právě stisknuté klávesy atd…");
    system("cls"); system("color a1"); system("chcp 852");
    printf("Takto CP 852 zmeni pismena s hackem s, z nebo dlouhe a na znaky š, ž, á.\n");
    system("pause"); printf("\n"); system("chcp 1250");
    printf("A takto je to vpořádku: š, ž, á.\n");
    system("pause"); printf("\n"); system("chcp 852");
    printf("Stisknete klavesu 'c' s hackem:"); znak=getche();
    printf("\nVybrali jste znak %c s ASCII kodem %d\n",znak,znak);
    printf("\nStisknete klavesu 's' s hackem:"); znak=getche();
    printf("\nVybrali jste znak %c s ASCII kodem %d\n",znak,znak);
    printf("\n"); system("chcp 1250");
    printf("Stiskněte klávesu 'č':"); znak=getche();
    printf("\nVybrali jste znak %c s ASCII kódem %d\n",znak,znak);
    printf("\nStiskněte klávesu 'š':"); znak=getche();
    printf("\nVybrali jste znak %c s ASCII kódem %d\n",znak,znak);
    printf("\nPřepněte si klávesnici z české na anglickou a do třetice stiskněte klávesu 'č': ");
    znak=getche();
    printf("\nVybrali jste znak %c s ASCII kódem %d\n",znak,znak);
    printf("\nA rovněž potřetí stiskněte klávesu 'š':"); znak=getche();
    printf("\nVybrali jste znak %c s ASCII kódem %d\n",znak,znak);
    system("pause"); system("cls"); system("color 9f");
    printf("\nA na závěr smyčka na hraní (smyčku ukončíte stiskem klávesy q). ");
    printf("\n Nezapomeňte si přepnout klávesnici zpět na českou.");
    do {
                                printf("\nStiskněte klávesu nebo kombinaci kláves:"); znak=getche();
                                    printf("\nVybrali jste znak %c s ASCII kódem %d\n",znak,znak);
    } while (znak != 'q');
}
```

6) Při výstupu textu na obrazovku se OS snaží všude prosazovat své znakové nastavení. Daleko pestřejší situace nastává při vytváření externích textových souborů. Řada textových editorů umožňuje nastavit výstupní znakovou sadu z několika možností. Z 8 bitových sad je to především Windows-1250, ISO-8859-2, DOS-CP852, bratři Kameničtí-KeybCS2 a MacCE. 16 a 32 bitové kódování ovládly univerzální znakové sady Unicode (UTF-8, UTF-16). Následující program vám ukáže ASCII kódy znaků externího textového souboru, jehož kopii jste si uložili pod názvem s1.txt do aktuálního adresáře programu. Pro vaše pokusy s textovými editory stačí uložit do souboru jediný znak, a to malé š. Pokud jste obdrželi externí soubor, jehož editor nemáte k dispozici, upravte si jeho kopii smazáním většiny jeho obsahu se zachováním některého charakterizujícího znaku z triumvirátu š, ž, ť nebo v něm bez úprav tyto znaky vyhledejte. Viz tabulka:

Znaková sada	Znak		
	š	**ž**	**ť**
Windows 1250	154	158	157
ISO-8859-2	185	190	187
CP-852	231	167	156
bratři Kameničtí	168	145	159
UTF-8	dvojkódy		
UTF-16	čtyř až pětikódy		