STU FIIT

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Základy procedurálneho programovania 1

Programujeme

Pripravte si papier (A4) a písatko!

10. 10. 2016

zimný semester 2016/2017

Úloha: Objem kvádra

- Napíš program v jazyku C, ktorý načíta zo vstupu tri celé čísla do 1000, a vypíše na výstup objem kvádra s rozmermi, ktoré boli na vstupe. (3 body)

```c
#include <stdio.h>

int main(void)
{
    int x,y,z;
    scanf("%d %d %d", &x, &y, &z);
    printf("%d\n", x*y*z);
    return 0;
}
```

Úloha: Súčet N čísel

- Napíš funkciu **sucet**, ktorá vypočíta súčet čísel v poli: vstupné argumenty pole N čísel (int).
- Postup: výsledok si musím niekde pamätať, označme to ako premennú **vysledok**, potom prejdem všetky čísla a pripočítam ich do premennej **vysledok**
- **Program:**

```
int sucet(int* pole, int n)
{
    int i, vysledok = 0;
    for (i = 0; i < n; i++)
        vysledok += pole[i];
    return vysledok;
}
```

Úloha: Vymeň hodnotu premenných int

- **Napíš (jednorázový) príkaz „niekde“ v programe:**

```
int tmp, u = 10, v = 20;
tmp = u; u = v; v = tmp;
```

- **Napíš (znovupoužiteľnú funkciu) na výmenu hodnôt dvoch premenných typu int.**

```
void swap(int *x, int *y)
{
    int tmp = *x;
    *x = *y;
    *y = tmp;
}
// pouzitie
int u = 10, v = 20;
swap(&u, &v);
```

Úloha: Minimum z poľa čísel

- Napíš funkciu **min**, ktorá vypočíta index najmenšieho čísla v poli rôznych čísel: vstup pole N čísel (int).
- Najskôr vymyslíme a napíšeme signatúru funkcie:

```
int min(int* pole, int n)
{
    if (n <= 0)
        return -1;
    int i, x = 0;
    for (i = 1; i < n; i++)
        if (pole[x] > pole[i])
            x = i;
    return x;
}
```

Úloha: Súčet N čísel zo vstupu

- Znovupoužitie programu pre súčet N čísel v poli
- Postup: Pridáme načítanie čísel zo vstupu
- Funkcia **nacitaj_pole**, vstupné argumenty adresa pola (s dostatkom miesta) kam načíta čísla
- **Program**:

```
void nacitaj_pole(int *pole, int *n)
{
    int i;
    scanf("%d", n); // nacitam pocet prvkov pola ... N
    for (i = 1; i <= *n; i++) // nacitam N cisel
        scanf("%d", &pole[i-1]); // i-te cislo do pamate pole[i-1]
}
```

STU FIIT

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Základy procedurálneho programovania 1

Reťazce (char*)

10. 10. 2016

zimný semester 2016/2017

Ako v programe pracovať so znakom?

- Opakovanie: Premenné sú schopné uchovať nejaký počet rôznych stavov (napr. short 65536, char 256)
- Koľko rôznych znakov chceme v programe uchovať?
  - Znaky anglickej abecedy: a, b, c, ... z (26)
  - Malé písmená (a až z)
  - Veľké písmená (A až Z)
  - Číslice (0 až 9)
  - Iné znaky: + - * = ! ? , ; :
- Postačuje dátový typ **char**
- **Rôzne stavy premennej typu char budeme interpretovať ako rôzne písmená**

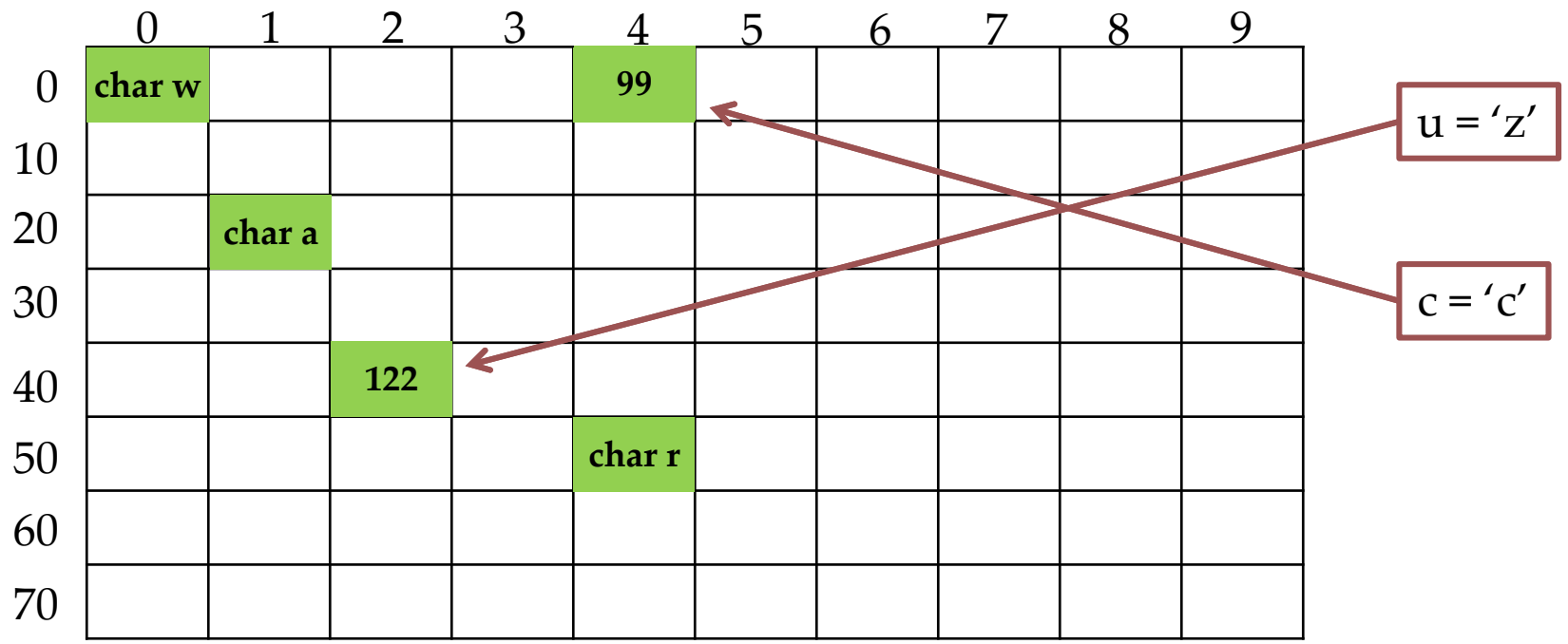
Ako v programe pracovať so znakom?

- Rôzne stavy sú vlastne rôzne hodnoty premennej
- **Napr.** písmeno **(malé) a** zodpovedá 97
- **Premennú typu char, ktorá má hodnotu 97 interpretujeme ako znak abecedy 'a'**

```
char c = 'a';
printf("%d", c); // vypise 97
```

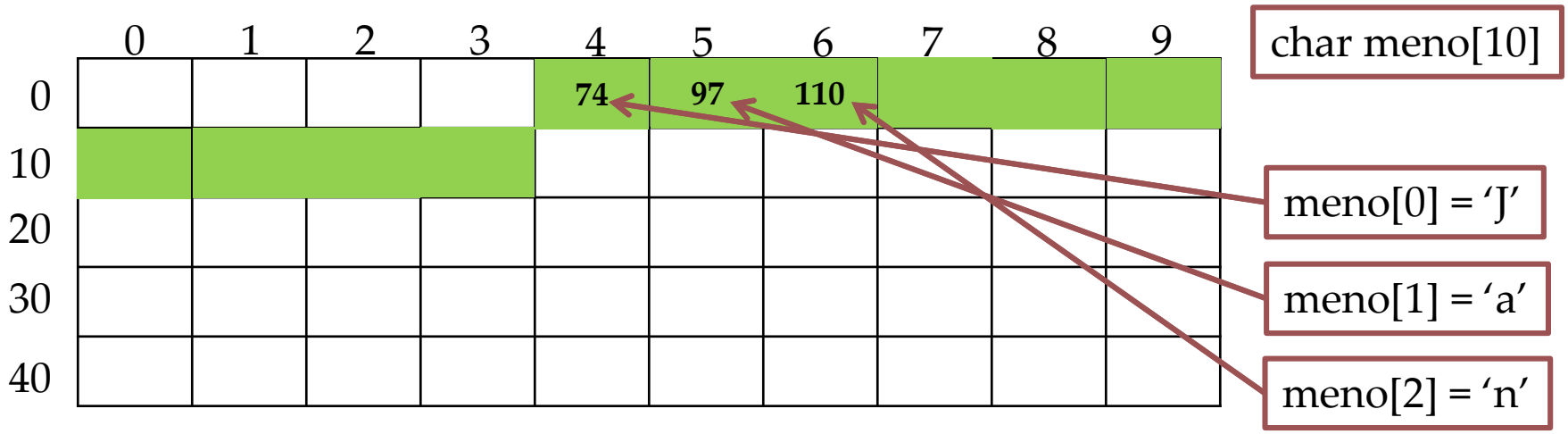
- Písmená **a až z** reprezentujeme nasledujúcimi hodnotami: b je 98, c je 99, ... z je ?
- Ako sú reprezentované veľké písmená? Inými hodnotami: A je 65, B je ? ...
- Ako sú reprezentované číslice? Inými hodnotami: znak nuly je 48, jednotka je 49, ... deviatka je 57

Znaky v pamäti



- Ako si do pamäte uložím meno? (napr. „Bratislava“)
  - Po znakoch: ‘B’, ‘r’, ‘a’, ‘t’, ‘i’, ‘s’, ‘l’, ‘a’, ‘v’, ‘a’
  - Znaky pôjdu v pamäti za sebou

Reťazec znakov v pamäti – napr. Jan



- Ako zistím dĺžku (počet znakov hodnoty) reťazca?
- Pre premennú meno je v pamäti vyhradených 10 znakov, ak je hodnota reťazca „Jan“, tak má len 3 znaky
- Riešenie: za posledným znakom hodnoty reťazca je číslo 0
- Reťazec „Jan“ je teda:

74	97	110	0

- Reťazec „0+2=5“ je:

48	43	50	61	53	0

Znaky v programovaní

- Znaková sada určuje spôsob ako interpretujem číselný kód na znak
- Štandardná znaková sada je ASCII
  - Znak je 7 bitový – teda 128 rôznych hodnôt
    0 a 127 rôznych znakov (pozn. číslica nula je 48)
  - Ôsmy bit sa používal ako kontrolný bit pri prenose
  - Prvých 32 znakov (číselné kódy 0 až 31) sú tzv. kontrolné kódy, nezodpovedajú štandardnému viditeľnému znaku, ale sú metainformácie pre zariadenia ktoré ASCII reťazec interpretujú
  - Väčšina ďalších znakových sád je rozšírenie ASCII
    napr: UTF-8, ISO/IEC 8859 -1, -2, -3, ...

ASCII tabuľka

- Prvých 32 znakov (číselné kódy 0 až 31) sú tzv. kontrolné kódy, nezodpovedajú štandardnému viditeľnému znaku, ale sú metainformácie pre zariadenia ktoré ASCII reťazec interpretujú
- Kód 0 je koniec reťazca
- Kód 3 (^C) je koniec textu
- Kód 4 (^D) je koniec prenosu
- Kód 9 je tabulátor '\t'
- Kód 10 je nový riadok '\n' (LineFeed)
- Kód 13 je návrat na začiatok riadku '\r' (CarriageReturn)

ASCII tabuľka

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	

Reťazec v programovacom jazyku C

- Reťazec je pole znakov (char)
  napr. char meno[5]
- Hodnota reťazca sú znaky až po ukončovací kód 0
- **Výpis reťazca** na obrazovku
  printf a formátovací reťazec %s:

```
char meno[5];
meno[0] = 'J';
meno[1] = 'a';
meno[2] = 'n';
meno[3] = 0;
printf("%s", meno); // vypise Jan
```

- Čo keď v reťazci nie je ukončenie kódom 0?
  - Vypisuje sa pamäť, až kým sa niekde ďalej nenarazí na kód 0
    Napr.: Jan#$TR#RF#E%9^#$@#T#TG#@T%

Reťazec v programovacom jazyku C

- Reťazec je pole znakov (char)
  napr. char meno[5]
- Hodnota reťazca sú znaky až po ukončovací kód 0
- **Načítanie reťazca** zo vstupu
  scanf a formátovací reťazec %s

```
char meno[5];
scanf("%s", meno); // nacita po prvu medzeru
printf("%s", meno); // vypise prve slovo
```

- scanf načíta do adresy (meno) znaky po prvý oddeľovač (medzera, riadok, tabulátor, ...)
- Čo keď je na adrese vyhradené menej pamäte ako načítam, napr. načítam „Bratislava“?

Reťazec v programovacom jazyku C

- Čo je na adrese vyhradené menej pamäte ako načítam, napr. načítam „Bratislava“?
  - Pri načítavaní sa začne prepisovať pamäť, a poškodia sa tým hodnoty iných premenných
  - Môže sa prerušiť vykonávanie programu
- Pre načítanie desaťznakovej hodnoty reťazca si musím v pamäti vyhradiť koľko znakov?
  - jedenásť ... dĺžka reťazca a kontrolný ukončovací kód 0

Operácie s reťazcami

- Už vieme reťazec znakov v programe uchovať:
  - **pole znakov** – napr. **char buf[100]**
  - Výpis na samostatný riadok: **printf("%s\n", buf);**
  - Znaky reťazca musia byť nasledované kódom 0, ktorý ukončuje hodnotu reťazca!
  - Načítanie: **scanf("%s", buf);**
  - Pre reťazcoch si musíme dávať veľký pozor, koľko pamäte máme pre reťazec vyhradené!
- Ako s reťazcom znakov pracovať?
  - Inými slovami:
    aké operácie môžeme s reťazcom vykonávať?

Operácie s reťazcami

- Načítanie: **scanf("%s", retazec);**
- Výpis: **printf("%s", retazec);**
- Ďalšie spravíme spoločne:
  - **zistiť dĺžku reťazca**
  - **zapíš do reťazca znak viac krát**
  - **nájdi znak (prvý výskyt znaku v reťazci)**
  - **pripoj reťazec k inému reťazcu**
  - **...**

Operácie s reťazcami

- **Zistiť dĺžku reťazca:**

```
// zistit dlzku retazca s (je spravne ukonceny retazec)
int dlzka(char *s)
{
    // postup: najst nulovy znak (znak s kodom 0, nie 48 :)
    int i;
    for (i = 0; ; i++) // ide do nekonecna
    {
        // i = 0, 1, 2, ... INFINITY (nekonecno)
        if (s[i] == 0)
        {
            // i = 5 : 'A' 'A' 'A' 'A' 'A' 0
            //          0   1   2   3   4  5
            // koniec retazca
            return i;
        }
    }
    // sem sa program vacsinou nedostane :)
}
```

Operácie s reťazcami

- **Zapíš do reťazca znak viac krát:**

```
// do retazca str (ktory ma vyhradenych aspon pocet+1 bajtov)
// zapisat pocet krat znak z
void opakuj_znak(char *str, char z, int pocet)
{
    int i;
    for (i = 0; i <= pocet-1; i++)
    {
        // i = 0, 1, ... pocet-1
        str[i] = z;
    }
    str[pocet] = 0;
}
```

Operácie s reťazcami

- **Nájdi znak (prvý výskyt znaku v reťazci):**

```
// ak tam znak nie je, vrati nejaky specialny kod... aky? <0
int najdi_znak(char *str, char z)
{
    int i;
    for (i = 0; str[i] != 0; i++)
    {
        if (str[i] == z)
            return i;
    }
    return -47;
}
```

Operácie s reťazcami

- **Pripoj reťazec k inému reťazcu:**

```
// vysledok = vysledok+oddelovac+s
void pripoj_retazec(char *vysledok, char *s, char oddelovac)
{
    // zistim dlzky oboch retazcov
    int d1 = dlzka(vysledok);
    int i, d2 = dlzka(s);

    // vysledok[0..d1-1] zostane zachovane
    // vysledok[d1] uz nebude 0, ale znak oddelovac
    vysledok[d1+0] = oddelovac;
    // printf("znak %d bude %c\n", d1+0, oddelovac);
    // vysledok[d1+1] bude s[0]
    // ..
    // vysledok[d1+d2+1] bude s[d2]; .. kolko je s[d2]? == 0
    for (i = 0; i <= d2; i++)
    {
        vysledok[d1+i+1] = s[i];
        //printf("znak %d bude %c\n", d1+i+1, s[i]);
    }
}
```

Operácie s reťazcami v jazyku C

- V programovacom jazyku C sú viaceré tieto operácie implementované v knižnici **string.h**
- Dĺžka reťazca: **strlen**
- Prekopírovať reťazec do iného reťazca: **strcpy, strncpy**
- Pripojiť reťazec k inému reťazcu: **strcat**
- Nájsť prvý výskyt znaku v reťazci: **strchr**
- Nájsť posledný výskyt znaku v reťazci: **strrchr**
- Porovnanie reťazcov: **strcmp**
- Vyhľadanie prvého výskytu podreťazca v reťazci: **strstr**
- Previesť reťazec na číslo: **sscanf, atoi, atol, atof**
- Previesť číslo na reťazec: **sprintf**
- Načítať reťazec (celý riadok) zo vstupu: **fgets, gets**

Úloha: Porovnanie dvoch reťazcov

- Dané sú dva reťazce r1 a r2, zisti, ktorý je v abecede skôr? Namiesto abecedy, použijeme ASCII kódy.
- Program si napíšte ako cvičenie ...

Načítaj celý riadok (aj medzery) do reťazca

- Použijeme funkciu **fgets**
- Program si napíšte ako cvičenie ...

Úloha: Načítaj binárne číslo na vstupe do intu

- Na vstupe je binárne číslo zapísané ako reťazec znakov 0 a 1, načítaj ho do premennej typu int.
- Program si napíšte ako cvičenie ...