



Programovanie v jazyku C

Prvé programy v C
Vývojové prostredie

[Programovací jazyk C]

- ❏ obľúbený a veľmi rozšírený
- ❏ jazyk profesionálnych programátorov
- ❏ univerzálny štruktúrovaný programovací jazyk strednej úrovne
- ❏ iná paradigma: procedurálne, nie objektové programovanie!
- ❏ vytvorený pôvodne pre systémové programovanie
- ❏ operačné systémy, ovládače, hry, iné programovacie jazyky, ...
- ❏ efektívny, rýchly kód
- ❏ štandardizovaný (ISO/ANSI)
- ❏ podporovaný na mnohých platformách (prenositel'nosť)
- ❏ úsporné vyjadrovanie
- ❏ väčšia sloboda ale i väčšia zodpovednosť

[Programovací jazyk C]

- ✦ Dennis Ritchie, AT&T Bell Labs, 70. roky, Unix

- ✦ štandardy: K & R (1978)
ANSI C (C89)
C99
C11

- ✦ objektovo orientované programovanie:
C++
C#
Objective C



[Prvý program v C]

```
/* pozdrav.c : prvý program v C */      komentár
```

```
#include <stdio.h>      vloženie hlavičkového súboru
```

```
int main(void)      funkcia vracia hodnotu typu int, nemá parametre
```

```
{
```

```
    printf("Ahoj svet!\n");
```

```
    printf("Programujem v C!");
```

```
    return 0;
```

```
}
```

ukončenie programu, funkcia main vráti hodnotu 0

Jazyk C rozlišuje malé a veľké písmená !!!

Ďalší program v C

```
/* priemer.c : aritmeticky priemer dvoch cisel */

#include <stdio.h>

int main(void)
{
    int x, y, sucet;

    printf("zadaj 1. cislo: ");
    scanf("%d", &x);
    printf("zadaj 2. cislo: ");
    scanf("%d", &y);
    sucet = x + y;
    printf("priemer z %d a %d je %f\n", x, y, sucet/2.0);

    return 0;
}
```



[Zapamätajte si !!!

reťazec, v ktorom špecifikujeme formát vstupu
na vstupe budú 2 celé čísla oddelené bielym znakom



```
scanf ( "%d %d", &x, &y );
```

funkcia na čítanie



adresy premenných,
do ktorých sa majú načítané hodnoty uložiť

[Zapamätajte si !!!]

reťazec, v ktorom špecifikujeme formát výstupu
(čo a ako chceme vypísať)



```
printf( "%d plus %d je %d\n", x, y, x + y );
```

funkcia na výpis



hodnoty, ktoré chceme vypísať
(dosadia sa na miesta formátovacích špecifikácií %d)

[Zapamätajte si !!!]

ak chceme používať funkcie zo štandardnej knižnice,
musíme do zdrojového súboru najprv vložiť správne hlavičkové súbory

Napr.:

```
#include <stdio.h>          // scanf, printf
#include <ctype.h>           // tolower, toupper
#include <stdlib.h>          // srand, rand
```

v komentári si môžete poznačiť,
kvôli ktorej funkcii hlavičkový súbor potrebujete



[Preklad na príkazovom riadku

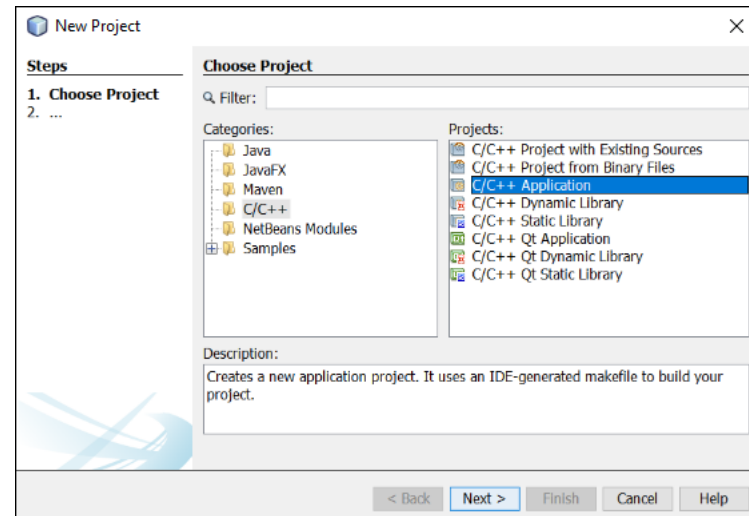
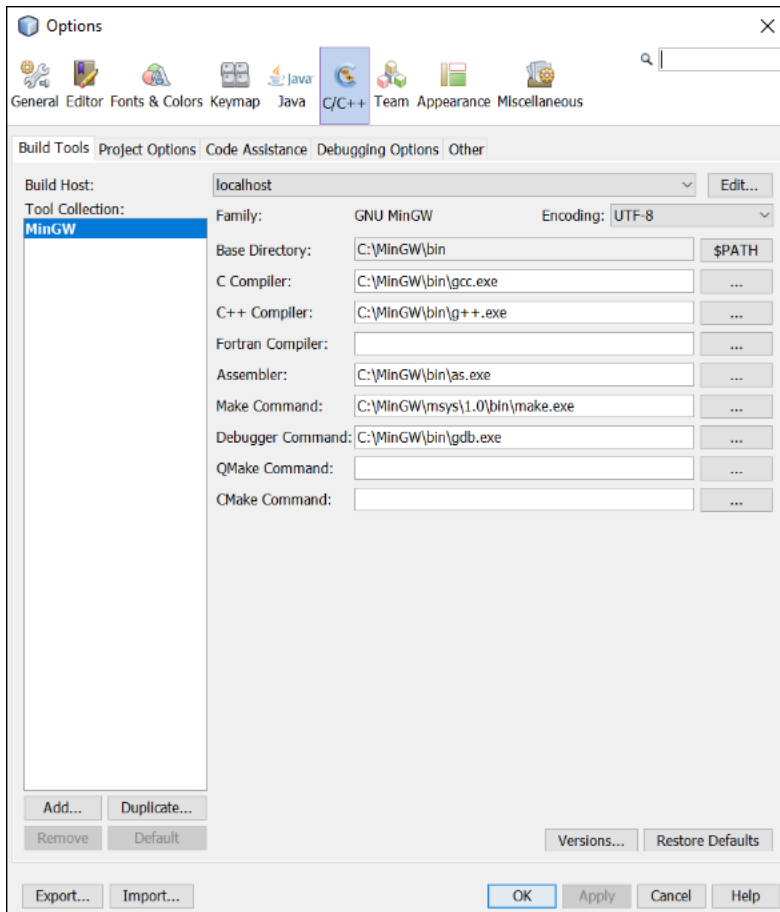
```
gcc prvy.c -o prvy
```

```
gcc prvy.c -Wall -o prvy
```

```
./prvy
```

```
gcc prvy.c -o prvy && ./prvy
```

IDE NetBeans + MinGW



Spracovanie programu

EDITOR

m1.c

m2.c

m1.h

PREPROCESSOR

vloženie hlavičkových súborov, odstránenie komentárov, rozvinutie makier atď.
(predspracovanie zdrojových súborov)

KOMPILÁTOR

m1.o **m2.o** (relatívny, objektový kód)
+ kód z použitých knižníc



LINKER (zostavovací program) → **EXE**

DEBUGGER

ladenie programu

[Riadiace konštrukcie poznáme z Javy]

`if, if-else, switch`
`while, for, do-while`

`continue, break`
`return`



[Operátory tiež

=

==

!=

<

>

<=

>=

+

-

*

/

%

++

--

+=

-=

*=

/=

&

&&

|

||

!

^



Jednoduché údajové typy

celé číslo

`int`

`short int (short)`

`long int (long)`

znak

`char`

reálne číslo

`float`

`double`

`long double`



[Jednoduché údajové typy

char, **short int**, **int** a **long int** môžu byť

unsigned neznamienkový typ

signed znamienkový typ

napr.

unsigned char 0..255

signed char -128..127

int myslí sa

char myslí sa

unsigned int stačí len

signed int

signed char

unsigned

Veľkosti typov, rozsahy hodnôt, sizeof

```
#include <stdio.h>
#include <limits.h>
#include <float.h>
```

hlavičkové súbory obsahujúce definície konštánt

```
int main(void)
{
```

operátor *sizeof* vráti veľkosť príslušného údajového typu

```
    printf("char          %d B\n", sizeof(char));
    printf("short         %d B\n", sizeof(short));
    printf("int            %d B\n", sizeof(int));
    printf("unsigned        %d B\n", sizeof(unsigned));
    printf("long            %d B\n", sizeof(long));
    printf("float           %d B\n", sizeof(float));
    printf("double          %d B\n", sizeof(double));
    printf("long double      %d B\n", sizeof(long double));

    printf("rozsah char      %d .. %d\n", CHAR_MIN, CHAR_MAX);
    printf("rozsah int        %d .. %d\n", INT_MIN, INT_MAX);
    printf("rozsah float      %g .. %g\n", FLT_MIN, FLT_MAX);
    printf("rozsah double     %g .. %g\n", DBL_MIN, DBL_MAX);

    return 0;
}
```


[Formátovacie špecifikácie]

<code>char</code>	<code>%c</code>	
<code>int</code>	<code>%d</code>	<code>%i</code>
<code>unsigned int</code>	<code>%u</code>	
<code>long int</code>	<code>%ld</code>	
<code>float</code>	<code>%f</code>	
<code>long double</code>	<code>%Lf</code>	
<code>double</code>	načítanie	<code>%lf</code>
<code>double</code>	výpis	<code>%f</code>

[Premenné]

```
long cislo;
float x, y, z;
unsigned pocet = 0;

int main(void)
{
    char c;
    double x;
    int i, j = 1;

    ...

    return 0;
}
```

← globálne premenné deklarujeme pred funkciou main

← lokálne premenné deklarujeme vo funkcii main (resp. na začiatku bloku)

premenné môžeme hneď pri deklarácii inicializovať !

[Konstanty]

Zápis celých čísel

desítkový: 12, 0, -44
osmičkový: 065, 015, 0, 01
šestnáctkový: 0X4A, 0xcd, 0x0, 0x1

12345L

číslo bude typu **long int**

884U

číslo bude typu **unsigned int**

Zápis desatinných čísel

3.1415 1.0 7. .38 5e6 2E-13

3.14f

číslo bude typu **float**

12e4L

číslo bude typu **long double**



[Konštanty

Znaky v apostrofoch

`'a'` `'x'` `' '` `'\013'` `'\x0A'`

`\n` `\t` `\b` `\a` `\\` `\"`

*escape (únikové) sekvencie
začínajú lomítkom*

Reťazce v úvodzovkách

`"Toto je retazec"`

`"\nToto \tje na 100%%"`

`\"iny\" retazec\n"`

Konštanty

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

```
#define PI 3.1415f
```

```
#define MAXPP 100
```

```
int main(void)
```

```
{
```

```
    int a[100MAXPP];
```

```
    for (int i = 0; i < 100MAXPP; i++) a[i] = 0;
```

```
    scanf("%f", &polomer);
```

```
    printf("%6.2f", 2*PI*r);
```

^{3.1415f}

```
    return 0;
```

```
}
```

#define je direktíva preprocesora

← takto definujeme tzv. *symbolické konštanty*
(makrá bez parametrov)

konštantá ako „*read only*“ *inicializovaná premenná*:

```
const int N = 10;
```

v C nemožno takúto konštantu použiť v definícii poľa, v C++ už áno!



[Logické výrazy (podmienky)]

v C **neexistuje typ *boolean* !**

logické hodnoty sú reprezentované celočíselnými

ľubovoľná **nenulová hodnota** (najčastejšie 1)

pravda

nulová hodnota

nepravda

ANSI C nepozná **true/false**, C99 už umožňuje používať tieto identifikátory (pre kompilátor je to ale aj tak 1 a 0).

Príklad – logický výraz

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

```
int main(void)
```

```
{
```

```
    int x = 0;
```

```
    int y = 0;
```

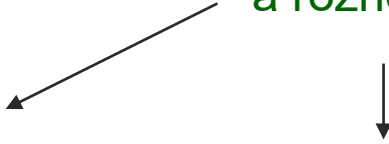
```
    int vyraz = !((x==0) || (y==0));
```

```
    printf("%d\n", vyraz);
```

```
    return 0;
```

```
}
```

vyskúšajte rôzne hodnoty x, y
a rôzne logické výrazy





[Bude podmienka splnená??]

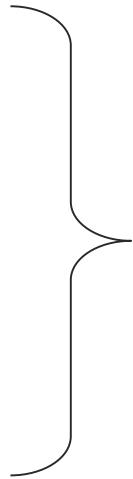
```
if (i==7)
```

Výsledkom výrazu je 1, ak má premenná i hodnotu 7 resp. 0, ak má i inú hodnotu.

```
if (i=7)
```

```
if (7)
```

```
if (i)
```

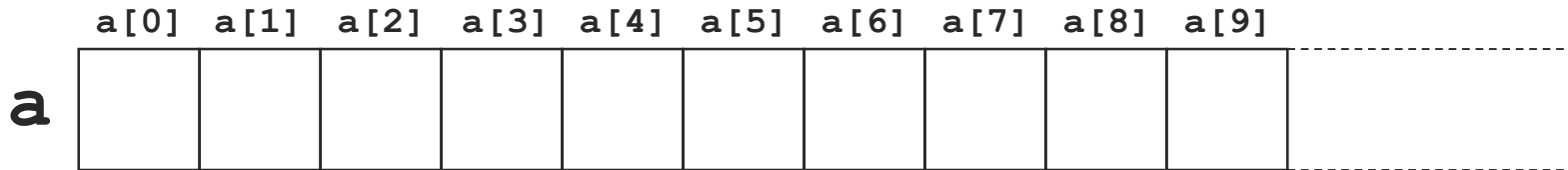


Vo všetkých prípadoch je výsledok výrazu celé číslo 7, čiže nenulová hodnota!



[Jednorozmerné pole]

typ prvku poľa → **int** meno poľa → **a** počet prvkov poľa → **[10]** ;



prvý prvok poľa má vždy index 0

ak má pole n prvkov, posledný prvok má index $n - 1$

Prekročenie rozsahu poľa sa nekontroluje !

napr. príkazom `a[10] = 321;` prepíšeme pamäť, ktorá „nie je naša“ 25



[Jednorozmerné pole - inicializácia]

```
int a[] = {12, 0, 33, 71, 62};
```

pole 5 prvkov typu int

rozmer poľa si prekladač doplní automaticky podľa počtu prvkov v inicializácii

```
double b[] = {1.82, 4.18, 33.2};
```

pole 3 prvkov typu double

```
int c[15] = {1, 2, 3};
```

pole 15 prvkov, prvky s indexami 3, 4, .., 14 sa nastavlia na hodnotu 0