Reference jazyka C - Tahák

Číselné literály

Celá čísla

0b101110	0B101110	dvojkové
02356		osmičkové
5236		desítkové
0x5df	0x5DF	šestnáctvkové

Znaménka

53 / +53	Pozitivní	- 53	negativní

Proměnné

Deklarace = jméno + typ

int i;	deklarace proměnné
char i = 'Z';	deklarace a **inicializace** proměnné na hodnotu 'Z'
float i, j, k;	deklarace více proměnných stejného typu najednou
const int a = 865;	deklarace konstantní proměnné, nelze později změnit hodnotu

Pojmenování proměnných

pocetStudentu20;	Alfanumerické, nesmí být klíčovým slovem jazyka C a začíná písmenem.
20pocetStudentu;	Nesmí začínat číslem.
return;	Nesmí být klíčovým slovem.
pocet studentu;	Není jen alfanumerický, ale i mezera
SIRKA	Konstanty bývají velkými písmeny
vypocetPrumeru()	Proměnné a funkce se mohou také skládat z více slov, rozlišených velikostí písmen

Primitivní datové typy

Celá čísla

typ	počet Bytů	rozsah hodnot
char	1	unsigned nebo signed
unsigned char	1	0 až 2^8-1
signed char	1	-2^7 až 2^7-1
int	2 nebo **4**	unsigned nebo signed
unsigned int	2 nebo **4**	0 až 2^{16}-1 nebo 2^{32}-1
signed int	2 nebo **4**	-2^{15} až 2^{15}-1 nebo -2^{31} až 2^{31}-1
short	2	unsigned nebo signed
unsigned short	2	0 až 2^{16}-1
signed short	2	-2^{15} až 2^{15}-1
long	**4** nebo 8	unsigned nebo signed
unsigned long	**4** nebo 8	0 až 2^{32}-1 nebo 2^{64}-1
signed long	**4** nebo 8	-2^{31} až 2^{31}-1 nebo -2^{63} až 2^{63}-1

Desetinná čísla

float	4	$\pm1{,}2\times10^{-38}$ až $\pm3.4\times10^{38}$
double	**8** nebo 4	$\pm2.3\times10^{-308}$ až $\pm1.7\times10^{308}$

Kvalifikátor

const type	Označí proměnnou, že jde pouze číst

Třídy ukládání

register	Pro rychlý přístup, v RAM či registru
static	Proměnná existuje i po skončení funkce, kde je deklarovaná
extern	Proměnná deklarována v jiném souboru

Přetypování

(typ) x	Vrátí proměnnou x s typem v závorce

Ukazatele

Deklarace

type *x;	Ukazatele mají datový typ, stejně jako normální proměnné
void *v;	Mohou mít i neúplný typ. Nemohou být použity jiné operátory, než je operátor přiřazení, protože je neznámá délka typu.
struct type *y;	Ukazatel na datovou strukturu
type z[];	Jméno pole, řetězce může být použito jako ukazatel na první prvek pole

Přístup k hodnotám

x	Adresa paměti
*x	Hodnota uložená na této adrese
&varName	Adresa paměti obyčejné proměnné varName

Ukazatel je proměnná, která obsahuje místo v paměti.

Práce s ukazateli

#include <stdlib.h>

Alokování

malloc();	Pokud je úspěšné, vrací umístění v paměti, jinak vrací NULL
type *x; x = malloc(sizeof(type));	Paměť pro proměnnou
type *y; y = malloc(sizeof(type) *length);	Paměť pro pole / řetězec
struct type *z; z = malloc(sizeof(struct type));	Paměť pro strukturu

Dealokování

free(ptrName);	Odalokuje paměť alokovanou pro proměnnou ptrName

Realokování

realloc(ptrName, size);	Pokusí se změnit velikost paměti, alokované pro proměnnou ptrName.

Adresy paměti, které vidíte, jsou z virtuální paměti, kterou operační systém přiřadí programu. Nejsou to fyzické adresy. Odkazování se na paměť, která není přiřazena programu, vygeneruje chybu segmentace OS.

Pole

Deklarace

type jmeno[int];	Nastavíme délku pole
type jmeno[int] = {x, y, z};	Nastavíme délku pole a inicializujeme prvky pole
type jmeno[int] = {x};	Nastavíme délku pole a inicializujeme všechny prvky na hodnotu x
type jmeno[] = {x, y, z};	Kompilátor nastaví délku pole na základě inicializovaných prvků

Velikost pole nemůže být po jeho deklaraci změněna

Rozměry

jmeno[int]	Jednorozměrné pole
jmeno[int][int]	Dvourozměrné pole

Přístup k hodnotám

jmeno[int]	Hodnota prvku s pořadím int v poli name

*(name + int)	Funguje stejně jako předchozí
Prvky jsou souvisle číslovány od 0 směrem nahoru	
&name[int]	Adresa paměti prvku s indexem int v poli s názvem name
name + int	Funguje stejně jako předchozí
Jednotlivé prvky jsou uloženy v paměti souvisle za sebou	
Měření velikosti	
sizeof(array) / sizeof(arrayType)	Vrací velikost pole (nejistý výsledek)
sizeof(array) / sizeof(array[0])	Vrací velikost pole (jistý výsledek)

Řetězce

'A' character	Jednoduché uvozovky
"AB" string	Dvojité uvozovky
\0	Nulový znak – ukončovací znak
Řetězce jsou pole znaků – typ char	
char name[4] = "Ash";	
Je stejné jako	
char name[4] = {'A', 's', 'h', '\0'};	
int i; for(i = 0; name[i]; i++){}	
\0 se vyhodnotí jako nepravda - false.	
Řetězce musí obsahovat prvek char pro \0	

Escape sekvence

\a	alarm (zvonek/pípnutí)	\b	backspace
\f	posunutí papíru	\n	nová řádka
\r	návrat vozíku	\t	horizontální tabulátor
\v	vertikální tabulátor	\\	zpětné lomítko
\'	jednoduché uvozovky	\"	dvojité uvozovky
\?	*otazník*		
\nnn	jakýkoliv osmičkový ANSI kód znaku		
\xhh	jakýkoliv šestnáctkový ANSI kód znaku		

Definice typů

Definice	
typedef unsigned short uint16;	Zkrácení dlouhého jména typu na uint16.
typedef struct structName { int a, b; } newType;	Vytvoření nového typu newType ze struktury
Deklarace	
uint16 x = 65535;	Deklarace proměnné x typu uint16
newType y = {0, 0};	Struktura y jako proměnná typu newType

Struktury

Definice	
struct strctName { type x; type y; };	Definice typu struktura s názvem strctName, se dvěma členy x a y.
Deklarace	
struct strctName varName;	Proměnná varName deklarovaná jako typ struktury s názvem strctName
struct strctName { type a; type b; } varName;	Zkratka pro definici struktury strctName a deklaraci proměnné varName s typem strctName
struct strctName varName = { a, b };	Deklarace proměnné varName, jako proměnnou typu strctName a inicializace této proměnné
Přístup k hodnotám	
varName.x	Hodnota x, které je členem struktury varName
Bitová pole	
Struct { char a:4, b:4 } x;	Deklaruje x se dvěma členy a a b, oba mají velikost 4 bity.

Uniony

Definice	
union uName{ int x; char y[8]; }	Proměnná uName typu union se dvěma členy x a y. Velikost odpovídá velikosti největšího členu.
Deklarace	
union uN vName;	Proměnná vName jako union typ uN
Přístup k hodnotám	
vName.y[int]	Členové nemohou ukládat hodnoty konkurenčně. Nastavení hodnoty y zničí hodnotu x
Uniony se používají pro ukládání různých datových typů do stejné oblasti paměti	

Výčtový typ

Definice	
enum bool { false, true };	Uživatelsky definovaný datový typ bool se dvěma možnými stavy false nebo true
Deklarace	
enum bool varName;	Proměnná varName datového typu bool
Přiřazení hodnoty	
varName = true;	Proměnné varName můžeme přiřadit pouze hodnotu false nebo true
Vyhodnocení	
If (varName == false)	Testování hodnoty proměnné varName

Funkce

Definice	
typ/void jmenoFunkce([parametry...]){ [return hodnota;] }	
pro jména funkcí platí stejná pravidla jako pro jména proměnných a navíc musí být jedinečné	
typ/void	Návratová hodnota funkce (void pokud není žádná návratová hodnota)
jmenoFunkce	Jméno funkce a závorky pro parametry
parametry...	Typy a jména parametrů (void pokud nic)
{}	Složené závorky ohraničující obsah funkce
return hodnota;	Návratová hodnota, která se vrátí na místo, odkud byla funkce volána. Přeskočí se v případě funkcí, které vrací void. Funkce se opustí okamžitě po příkazu return.

Předání parametrů hodnotou nebo odkazem

void f(typ x); f(y);	Předání proměnné y funkci f jako parametr x (předání hodnotou)
void f(typ *x); f(array);	Předání pole/řetězce funkci f jako parametr x (předání odkazem)
void f(typ *x); f(struktura);	Předání struktury funkci f jako parametr x (předání odkazem)
void f(typ *x); f(&y);	Předání proměnné y funkci f jako parametr x (předání odkazem)

typ f(){
return x;
}
	Proměnná se vrací hodnotou

typ f(){
typ x;
return &x;
}
	Proměnná se vrací odkazem

Předávání odkazem nám umožní uvnitř funkce změnit původní proměnnou.

Rozsah platnosti

```
int f(){
   int i = 0;
}
i++;
```

i je deklarováno uvnitř funkce f(),
i neexistuje mimo tuto funkci f()

Deklarace

typ jmenoFunkce(parametry...);

Takzvaná hlavička funkce se umisťuje před definicí nebo použitím funkce (zpravidla před funkcí main).

typ jmenoFunkce([parametry...])	stejný typ, jméno i parametry jako definovaná funkce
;	středník místo složených závorek

main()

int main(int argc, char *argv[]){ return int; }

i je deklarováno uvnitř funkce f(),
i neexistuje mimo tuto funkci f()

Anatomie

int main	vstupní bod programu
int argc	počet parametrů programu z příkazové řádky
char *argv[]	Parametry z příkazové řádky v poli řetězců První je vždy jméno souboru programu.
return int;	Výstupní hodnota (integer – celé číslo) která se vrátí do operačního systému při opuštění programu

Parametry z příkazové řádky

app dva 3	Tři parametry „app“, „dva“ a „3“
app "dva 3"	Dva parametry „app“ a „dva 3“

main je první funkce, která se volá v okamžiku, spuštění programu

Komentáře

```
// Toto j komentář na jednu řádku
// po výskytu těchto dvou lomítek, už nebude na řádce nic dál zkompilováno
/* Toto je víceřádkový komentář!
   Mezi tímto ohraničením, se nic nezkompiluje. */
```

podmínka (větvení)

if, else if, else

if(a) b;	Provede příkaz b, pokud je a pravda.

if(a){
b;
c;
}
	Provede příkaz b a c, pokud je a pravda.

if(a){
b;
} else {
c;
}
	Provede příkaz b, pokud je a pravda, jinak provede příkaz c.

if(a){
b;
} else if (c){
d;
} else {
e;
}
	Provede příkaz b, pokud je a pravda, jinak pokud je c pravda, tak provede příkaz d, jinak provede příkaz e.

switch, case, break

switch(a){
case b: c;
}
	Provede příkaz c, pokud se proměnná a rovná hodnotě b

switch(a){
default: b;
}
	Provede příkaz b, pokud proměnná a nemá žádnou zmíněnou hodnotu

switch(a){
case b:
case c: d;
}
	Provede příkaz d, pokud se proměnná a rovná hodnotě b nebo hodnotě c

switch(a){
case b: c;
case d: e;
default: f;
}
	Provede příkazy c, e a f, pokud je a rovno hodnotě b.
	Provede příkazy e a f, pokud je a rovno hodnotě e.
	Jinak provede jen příkaz f.

switch(a){
case b: c;
break;
case d: e;
break;
default: f;
}
	Provede příkaz c, pokud je a rovnou hodnotě b.
	Provede příkaz e, pokud je a rovno hodnotě d.
	Jinak provede příkaz f.

opakování (cykly)

while

```
int x = 0;
while(x < 10) {
   x += 2;
}
```

Cyklus se neprovede, pokud je testovaná podmínka na počátku false – tedy rovna nepravdě

int x = 0;	Deklaruje a inicializuje celočíselnou proměnnou x
while()	Klíčové slovo cyklu a závorky pro umístění podmínky
x < 10	Testovaná podmínka
{}	Složené závorky uzavírající tělo cyklu – to co cyklus vykonává
x += 2;	Obsah cyklu

do while

```
char c = 'A';
do {
   c++;
} while(c != 'Z');
```

Vždy provede tělo cyklu, alespoň jednou.

char c = 'A';	Deklaruje a inicializuje proměnnou c typu char
do	Klíčové slovo cyklu
{}	Složené závorky uzavírající tělo cyklu – to co cyklus vykonává
c++;	Obsah cyklu
while();	Klíčové slovo cyklu a závorky pro umístění podmínky
c != 'Z'	Testovaná podmínka

for

```
int i;
for(i = 0; n[i] != '\0'; i++){
}
```
(podle standardu C89)

nebo

```
for(int i = 0; n[i] != '\0'; i++){
}
```
(podle standardu C99+)

Cyklus s daným počtem opakování

int i;	Deklaruje celočíselnou proměnnou i
for()	Klíčové slovo cyklu
i = 0;	Inicializuje celočíselnou proměnnou i
n[i] !='\0';	Testovaná podmínka
i++	Přičítá jedničku k proměnné i
{}	Složené závorky uzavírající tělo cyklu – to co cyklus vykonává

continue

```
int i=0;
while(i<10){
   i++;
   continue;
   i--;
}
```

Přeskočí zbytek obsahu cyklu a začne znovu od začátku cyklu.

break

```
int i=0;
while(1){
   if(x==10){
      break;
   } i++;
}
```

Přeskočí zbytek obsahu cyklu a vyskočí z něj.

Standardní vstup (klávesnice) a výstup (monitor)

#include <stdio.h>

Znaky

getchar()	Vrací ANSI kód jednoho znaku z bufferu vstupního streamu (proudu) jako celé číslo
putchar(int)	Vytiskne jeden znak zadaného ANSI kódem celým číslem do bufferu výstupního streamu (proudu)

Řetězce

gets(strJmeno)	Přečte řádku ze vstupního streamu (proudu) do proměnné typu string (řetězec)
Alternativa	
fgets(strJmeno, delka, stdin);	Přečte řádku ze vstupního streamu (proudu) do proměnné typu string (řetězec)
puts("string")	Vytiskne řetězec do výstupního streamu - proudu

Formátovaná data

scanf("%d", &x)	Načte hodnotu/hodnoty (typ je definován formátovacím řetězcem) do proměnné/proměnných (typ se musí shodovat) ze vstupního streamu. Přestane číst v okamžiku prvního výskytu prázdného znaku. & prefix není zapotřebí pro pole (včetně řetězců) (není jisté použití)
printf("I love %c %d!", 'C', 99)	Tiskne data (ve formátu definovaném formátovacím řetězcem) jako řetězec do výstupního streamu.
Alternativa	
fgets(strName, length, stdin); sscanf(strName, "%d", &x);	Použijte funkci fgets pro omezení vstupní délky, potom použijte sscanf pro načtení výsledného řetězce na místo funkce scanf. (bezpečné použití)

Buffery streamu musí být spláchnuty, aby reagovali na změny. Znaky ukončující řetězec mohou spláchnout výstup, zatímco znaky označující novou řádku mohou spláchnout vstup. Bezpečné funkce jsou takové, které vás nechají určit délku vstupu. Funkce s nejistým použitím to nedělají a riskují přetečení paměti.

Typy zástupných symbolů (f/printf a f/scanf)

printf("%d%d...", arg1, arg2...);

typ	příklad	popis
%d, %i	-42	desítkové číslo se znaménkem
%u	42	kladné desítkové číslo
%o	52	kladné osmičkové číslo
%x, %X	2a nebo 2A	kladné šestnáctkové číslo
%f, %F	1.21	desítkové desetinné číslo se znaménkem
%e, %E	1.21e+9, 1.21E+9	desítkové desetinné číslo, vědecký zápis
%c	a	znak A
%s	A string.	řetězec znaků
%p		ukazatel
%%	%	znak procenta

formát ukazatele záleží na architektuře a implementaci

Formátování zástupných symbolů (f/printf a f/scanf)

%[Flags][Width][.Precision][Length]Type

Značky

-	Zarovnává vlevo, namísto standardně vpravo
+	Znaménko pro oba druhy čísel – pozitivní i negativní
mezera	Ohraničí tištěné zleva mezerami
0	Ohraničí tištěné zleva nulami

Šířka

celé číslo	Minimální množství znaků, které se vytisknou, když je to potřeba, tak uskuteční ohraničení zleva mezerami nebo nulami. Nebude ořezávat.
*	Šířka specifikována předchozím parametrem v printf.

Přesnost

.celé číslo	Minimální množství číslic, které se vytisknou pro značky %d, %i, %o, %u, %x, %X. Zleva se vyplní nulami. Neořízne se. Přeskočí nulové hodnoty.
	Minimální množství číslic, které se vytisknou po desetinné čárce pro značky %e, %E, %f, %F (defaultně 6)
	Maximální množství znaků, které se vytisknou z %s (řetězce)

.	Pokud není zadáno žádné celé číslo, je to defaultně nula.
.*	Přesnost specifikována předchozím parametrem v printf

Vstup a výstup do a ze souboru

#include <stdio.h>

Otevírání

FILE *fptr = fopen(filename, mode);

FILE *fptr	Deklaruje proměnnou fptr jako ukazatel na typ FILE (uchovává umístění streamu namísto umístění v paměti)
fopen()	Pokud je úspěšný, vrací ukazatel na umístění streamu, jinak vrací nulu
filename	Řetězec, který obsahuje cestu k souboru a jméno tohoto souboru
mode	Řetězec určující mód přístupu k souboru
Módy	
"r" / "rb"	Čte existující textový / binární soubor
"w" / "wb"	Zapíše nový/přepíše existující textový / binární soubor
"a" / "ab"	Zapíše nový/připojí k existujícímu textovému/binárnímu souboru
"r+" / "r+b" / "rb+"	Čte a zapisuje do existujícího textového / binárního souboru
"w+" / "w+b" / "wb+"	Čte a zapisuje do nového / nebo přes existující textový / binární soubor
"a+" / "a+b" / "ab+"	Čte a zapisuje do nového / nebo připojuje k existujícímu textovému / binárnímu souboru

Uzavírání

fclose(fptr);	Spláchne buffery a uzavře stream (proud). Pokud je úspěšný vrací 0, jinak vrací EOF.

Náhodný přístup

ftell(fptr)	Vrací současnou pozici v souboru ve formátu long integeru.
fseek(fptr, offset, origin);	Nastavuje současnou pozici v souboru. Pokud je úspěšná vrací false, jinak vrací true. Offset je typu long integer.
Odkud se počítá offset - origin	
SEEK_SET	Začátek souboru
SEEK_CUR	Současná pozice v souboru
SEEK_END	Konec souboru

Pomůcky

feof(fptr)	Testuje zda jsme na konci souboru
rename(strOldName, strNewName)	Přejmenuje soubor
remove(strName)	Smaže soubor

Znaky

fgetc(fptr)	Vrací přečtený znak nebo EOF, pokud je neúspěšná.
fputc(int c, fptr)	Vrací zapsaný znak nebo EOF, pokud je neúspěšná.

Řetězce

fgets(char *s, int n, fptr)	Přečte n-1 znaků ze souboru fptr do řetězce s. Skončí na EOF a \n. (bezpečná)
fputs(char *s, fptr)	Zapíše řetězec s do souboru fptr. Vrací kladné číslo v případě úspěchu, jinak EOF.

Formátovaná data

fscanf(fptr, format, [...])	Stejný jako scanf, má navíc parametr ukazatele na soubor (Funkce s nejistým použitím)
fprintf(fptr, format, [...])	Stejný jako printf, má navíc parametr ukazatele na soubor
Alternativa	
fgets(strName, length, fptr); sscanf(strName, "%d", &x);	Používá fgets, aby se omezila délka vstupu a pak používá sscanf pro přečtení výsledného řetězce namísto scanf. (bezpečné)

Binární data

fread(void *ptr, sizeof(element), number, fptr)	Přečte počet „number“ elementů ze souboru fptr do pole *ptr. (bezpečné)
fwrite(void *ptr, sizeof(element), number, fptr)	Zapíše počet „number“ elementů do souboru fptr z pole *ptr.

Bezpečné funkce jsou takové, které vás nechají určit délku vstupu. Funkce s nejistým použitím to nedělají a riskují přetečení paměti.

Knihovna pro práci s řetězci

#include <string.h>

strlen(a)	Vrací množství znaků v řetězci a jako celé číslo. Vynechává nulový znak \0. (není jisté použití)
strcpy(a, b)	Kopíruje řetězce. Kopíruje řetězec b přes řetězec a, až do a včetně nulového znaku \0. (není jisté použití)
strcat(a, b)	Spojuje řetězce. Kopíruje řetězec b přes řetězec a, až do a včetně nulového znaku \0. Začíná na pozici nulového znaku \0 v řetězci a. (není jisté použití)
strcmp(a, b)	Porovnává řetězce. Vrací nepravdu, pokud se řetězec a rovná řetězci b, jinak vrací pravdu. Ignoruje znaky po nulovém znaku \0. (není jisté použití)
strstr(a, b)	Hledá řetězec b uvnitř řetězce a. Pokud je úspěšná vrací ukazatel, jinak vrací NULL. (není jisté použití)
Alternativy	
strncpy(a, b, n)	Kopíruje řetězce. Kopíruje n znaků z řetězce b přes řetězec a, až do a včetně nulového znaku \0. (bezpečné použití)
strncat(a, b, n)	Spojuje řetězce. Kopíruje n znaků z řetězce b přes řetězec a až do a včetně nulového znaku \0, začíná na pozici nulového znaku \0 v řetězci a. (bezpečné použití)
strncmp(a, b, n)	Porovná prvních n znaků dvou řetězců. Vrací nepravdu, pokud se řetězec a rovná řetězci b, jinak vrací pravdu. Ignoruje znaky po nulovém znaku \0. (bezpečné použití)

Bezpečné funkce jsou takové, které vás nechají určit délku vstupu. Funkce s nejistým použitím to nedělají a riskují přetečení paměti.

Knihovna pro práci se znaky

#include <ctype.h>

tolower(char)	Změní znak na malé písmeno.
toupper(char)	Změní znak na velké písmeno.
isalpha(char)	Vrací pravdu, pokud je znak písmenem abecedy, jinak vrací nepravdu.
islower(char)	Vrací pravdu, pokud je znak malým písmenem z abecedy, jinak vrací nepravdu.
isupper(char)	Vrací pravdu, pokud je znak velkým písmenem z abecedy, jinak vrací nepravdu.
isnumber(char)	Vrací pravdu, pokud je znak číslo od 0 do 9, jinak vrací nepravdu.
isblank	Vrací pravdu, pokud je znak prázdným znakem (' ', '\t', '\n'), jinak vrací nepravdu.

Knihovna pro práci s časem

#include <time.h>

Typy proměnných	
time_t	Uchová čas z kalendáře.
struct tm *x;	Uloží datum a čas.
Členové struktury tm	
int tm_sec	Sekundy od 0 do 59.
int tm_min	Minuty od 0 do 59.
int tm_hour	Hodiny od 0 do 23.
int tm_mday	Den v měsíci od 1 do 31.
int tm_mon	Měsíc od 0 do 11.
int tm_year	Roky od 1900
int tm_wday	Den v týdnu, 0 až 6
int tm_yday	Den v roce, 0 až 365
int tm_yday	Letní čas.
Funkce	
time(NULL)	Vrací Unixový čas (počet sekund od 1. 1. 1970).
time(&time_t);	Uloží aktuální čas v proměnné time_t.
ctime(&time_t)	Vrací proměnnou time_t jako řetězec.
x = localtime(&time_t);	Rozloží proměnnout time_t na členy struktury tm

Direktivy preprocesoru

#include <inbuilt.h>	Nahradí řádku obsahem standardního hlavičkového souboru jazyka C
#include "./custom.h"	Nahradí řádku obsahem zvoleného hlavičkového souboru jazyka C, musí se uvést i cesta k danému souboru
#define NAME value	Nahradí v kódu všechny výskyty názvu NAME hodnotou value

Unární operátory

V sestupném pořadí, podle vyhodnocovací priority

+A	Sečte 0 a A (0 + A)
-A	Rozdíl 0 a A (0 - A)
!A	Logická negace proměnné A
~A	Bitová negace proměnné A
++A	Přičte jedničku k A, A = A + 1
--A	Odečte jedničku od A, A = A - 1
A++	Vrátí A, a pak k němu přičte 1, A = A + 1
A--	Vrátí A, a pak od něj odečte 1, A = A - 1
(type)A	Přetypování A na typ type
&A;	Umístění v paměti pro proměnnou A
sizeof(A)	Velikost paměti, kterou zabírá proměnná A v paměti v Bytech.

Binární operátory

V sestupném pořadí, podle vyhodnocovací priority

A * B;	Součin A a B, A x B
A / B;	Dělení čitatele jmenovatelem, jmenovatel se nesmí rovnat nule
A % B;	Zbytek po dělení čitatele jmenovatelem
A + B;	Součet A a B
A - B;	Rozdíl A a B
A << B;	Bitový posun vlevo – posune A o počet bitů B
A >> B;	Bitový posun vpravo – posune A o počet bitů B
A < B;	Menší než. Je pravda, pokud je A menší než B, jinak je nepravda
A <= B;	Menší nebo rovno. Je pravda, pokud je A menší nebo rovné B, jinak je nepravda
A > B;	Větší než. Je pravda, pokud je A větší než B, jinak je nepravda
A >= B;	Větší nebo rovno. Je pravda, pokud je A větší nebo rovné B, jinak je nepravda
A == B;	Rovnost. Je pravda, pokud se A rovná B, jinak je nepravda.
A != B;	Nerovnost. Je pravda, pokud se A nerovná B, jinak je nepravda.
A & B;	Bitový součin AND mezi A a B
A ^ B;	Bitový exkluzivní součet XOR mezi A a B
A | B;	Bitový součet OR mezi A a B
A && B;	Logický součin AND. Je pravdivý, pokud jsou oba A i B nenulové.
A || B;	Logický součet OR. Je pravdivý, pokud je buď A nebo B nenulové.

Ternární operátory a operátory přiřazení

V sestupném pořadí, podle vyhodnocovací priority

X ? A : B;	Provede A pokud je X pravdou, jinak provede B (if(x){ a; } else { b; })
X = A;	Přiřadí hodnotu A do proměnné X
A *= B;	Přiřadí součin A a B do A (a = a × b)
A /= B;	Přiřadí výsledek po dělení A / B do A (a = a ÷b)
A %= B;	Přiřadí zbytek po dělení A % B do A (a = a mod b)
A += B;	Přiřadí součet A a B do A (a = a + b)
A -= B;	Přiřadí rozdíl A a B do A (a = a - b)
A <<= B;	Přiřadí bitový posun A vlevo o B pozic do A
A >>= B;	Přiřadí bitový posun A vpravo o B pozic do A
A &= B;	Přiřadí bitový součin A a B do A
A ^= B;	Přiřadí exkluzivní bitový součet A a B do A
A |= B;	Přiřadí bitový součet A a B do A

Rezervovaná slova jazyka C

_Alignas	break	float	signed
_Alignof	case	for	sizeof
_Atomic	char	goto	static
_Bool	const	if	struct
_Complex	continue	inline	switch
_Generic	default	int	typedef
_Imaginary	do	long	union
_Noreturn	double	register	unsigned
_Static_assert	else	restrict	void
_Thread_local	enum	return	volatile
auto	extern	short	while

Přeloženo a upraveno z C Reference Cheat Sheet by Ashlyn Black