Základy programovania 25ZS
Control Flow a Cyklus

Prednášajúci: prof. Gabriel Juhás
Cvičiaci: Milan Mladoniczky

Dátové typy

Kategória	Popis	Dátový typ
Primitívny typ	Primitívny dátový typ je základný typ, ktorý je použitý na reprezentácia jednoduchých hôdnot.	int, char, float, double, void
Derivovaný typ	Dátový typ, ktorý vznikol deriváciou z primitívneho typu ale zložením viacerých primitívnych dátových typov.	pole (array), pointer, funkcia (function)
Používateľov definovaný typ	Dátový typ definovaný programátorom, zvyčajne zložením iných dátových typov.	štruktúra (structure), union, enum

Data Type	Size (bytes)	Range	Format Specifier
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4	1.2E-38 to 3.4E+38	%f
double	8	1.7E-308 to 1.7E+308	%lf
long double	16	3.4E-4932 to 1.1E+4932	%Lf

Porovnávacie operátory

<	menšie než	vráti pravdu ak ľavý operand je menší než pravý
<=	menšie alebo rovné než	vráti pravdu ak ľavý operand je menší alebo rovný ako pravý
>	väčšie než	vráti pravdu ak ľavý operand je väčší než pravý
>=	väčšie alebo rovné než	vráti pravdu ak ľavý operand je väčší alebo rovný ako pravý
==	rovné	vráti pravdu ak ľavý a pravý operand majú rovnakú hodnotu
!=	nerovné	vráti pravdu ak sa hodnota ľavého a pravého operandu nerovnajú

Logické operátory

- **&&** AND / a súčasne vráti pravdu ak oba operandy sú vyhodnotené ako pravda
- **||** OR / alebo vráti pravdu ak aspoň jeden operand je vyhodnotený ako pravda
- **!** NOT / negácia vráti pravdu ak oba operandy sú vyhodnotené ako nepravdivé

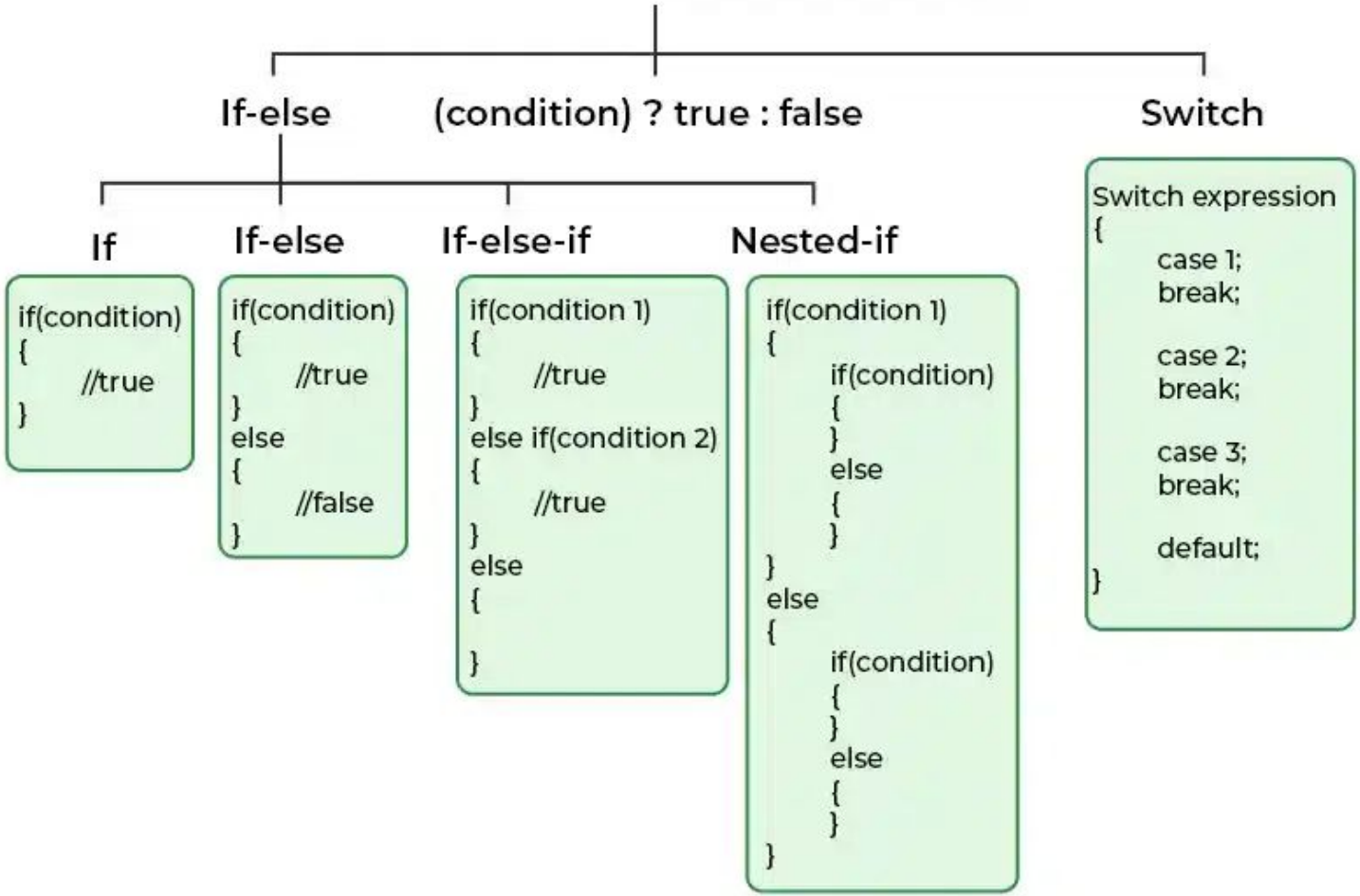
Control Flow

- Podmienené príkazy (známe aj ako rozhodovacie riadiace štruktúry), ako sú **if, if else, switch** atď., sa v programoch v jazyku C používajú na rozhodovacie účely.
- Sú známe aj ako rozhodovacie príkazy a používajú sa na vyhodnotenie jednej alebo viacerých podmienok a na rozhodnutie, či sa súbor príkazov vykoná alebo nie. Tieto rozhodovacie príkazy v programovacích jazykoch rozhodujú o smere toku vykonávania programu (tzv. control flow).

Control Flow

- V reálnom živote nastávajú situácie, keď sa musíme rozhodnúť a na základe týchto rozhodnutí sa rozhodujeme, čo budeme robiť ďalej.
- Podobné situácie nastávajú aj pri programovaní, keď musíme urobiť nejaké rozhodnutie a na základe týchto rozhodnutí vykonáme ďalší blok kódu. Napríklad v jazyku C ak nastane ***x***, potom vykonaj ***y***, inak vykonaj ***z***.
- V jazyku C môže byť aj viac podmienok, napríklad ak nastane ***x***, potom vykonaj ***p***, inak ak nastane podmienka ***y***, vykonaj ***q***, inak vykonaj ***r***.
- Táto podmienka C **if-else** je jedným z mnohých spôsobov importovania viacerých podmienok.

Conditional Statements in C



Podmienka IF

- Podmienka sa vždy musí vyhodnotiť na pravdivostnú hodnotu: TRUE alebo FALSE.
- Telo výrazu sa vykoná iba ak sa podmienka vyhodnotí na TRUE.

```
if (podmienka) {
        // Telo výrazu ak je podmienka pravda
}
```

IF - ELSE

- Prvé sa kontroluje podmienka **if** výrazu.
- Ak je podmienka pravdivá vykoná sa telo **if** výrazu.
- Ak nie je pravdivá vykoná sa telo **else** výrazu.
- Vždy sa vykoná iba jeden z týchto dvoch blokov.

```
if (podmienka) {
        // Telo výrazu ak je podmienka pravda
}
else {
        // Telo výrazu ak podmienka nie je pravda
}
```

IF - ELSE IF - ... - ELSE

- V prípade potreby viacero podmienok s rôznym telom výrazu.
- Vždy sa vykoná iba jeden blok **if** alebo **else if** (telo výrazu), pre ktorý je podmienka pravdivá. Ostatné sú vynechané.
- Ak ani jedna podmienka nie je pravdivá telo výrazu **else** je vykonané.

```
if (podmienka) {
        // Telo výrazu ak táto je podmienka pravda
}
else if (iná podmienka) {
        // Telo výrazu ak táto je podmienka pravda
}
else {
        // Telo výrazu ak podmienka nie je pravda
}
```

Praktická ukážka –>

Inkrementácia a dekrementácia premennej

- Typickou situáciou v programovaní je tzv. inkrementácia premennej, t.j. zvýšenie jej hodnoty o 1, resp. dekrementácia premennej, t.j. zníženie jej hodnoty o 1.
- V jazyku C sa to vykoná nasledovným spôsobom. Povedzme, že máme premennú i :

Zvýšenie o 1:
- i = i + 1
- i += 1
- i++ // len o inkrement 1

Zníženie o 1:
- i = i - 1
- i -= 1
- i– // len o dekrement 1

For cyklus

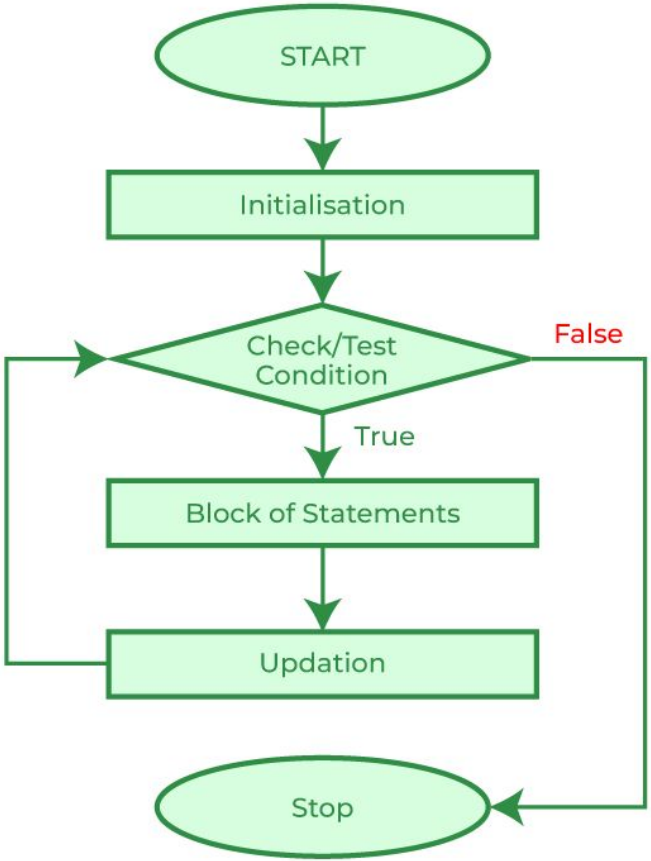
- V mnohých programovacích jazykoch (vrátane C) existujú príkazy, ktoré Vám umožnia zapísať opakovanie nejakej postupnosti príkazov.
- Lepšie povedané, v programovaní / algoritmizácii existuje koncept opakovania postupnosti príkazov.
- Príkazy, pomocou ktorých vieme počítaču prikázať, aby nejakú postupnosť príkazov zopakoval, nazývame cykly alebo aj iterácie.
- Príkaz **for** je základným typom iterácií – ide o tzv. iteráciu s vopred známym počtom opakovaní.
- V jazyku C je predstaviteľom iterácií s vopred známym počtom opakovaní tzv. for-cyklus (angl. for-loop)

For cyklus

Definícia for cyklu pozostáva z:

- inicializácia kontrolnej premennej
- testovacia podmienka
- aktualizácia kontrolnej premennej

```
for (inicializácia; podmienka; aktualizácia) {
        // telo cyklu ... príkazy
        break;  // ukončenie for-cyklu
        continue; // pokračovať ďalšou iteráciou
}
```


START
Initialisation
Check/Test Condition
False
True
Block of Statements
Updation
Stop

Praktická ukážka –>

Prečo používať cykly

- Výhodou cyklov je jednoduchý zápis opakovania postupnosti príkazov.
- Často sa stretnete v praxi so situáciou, že chcete opakovať nejakú činnosť, avšak počet opakovaní nejakým spôsobom závisí od počtu dát, ktoré máte k dispozícii. Napríklad sa stretneme so situáciou, že budeme chcieť pracovať so skupinou čísel, ktorých počet sa môže meniť a opakovanie nejakej činnosti bude závisieť práve od počtu čísel v skupine!
- Cykly sú taktiež užitočné v situácii, že potrebujete – z nejakého dôvodu – pracovať s postupne sa meniacimi číslami (napríklad potrebujete generovať čísla 1, 2, 3, … atď.). Cykly predstavujú veľmi jednoduchý spôsob, ako takéto čísla generovať.
- A ešte veľa ďalších aplikácií...