2.Sada otázok – C++ jazykové konštrukcie

1. Aký je rozdiel medzi **operátormi** a **operandami** výrazu?

Ak zapíšeme v matematike napríklad a + b, hovoríme o **výraze** . Ten má dva **operandy** **a** a **b** jeden **operator** **+** .

1. Aké **druhy operátorov** poznáme ? Uveďte príklady ?

Aritmetické (+,-,\*,/), Logické(logický súčin &&, logický súčet [**||**](http://ctp.mkprog.eu/sk/c%2B%2B/logicky_sucet/) alebo negácia !), Bitové (bitový posun doľava/doprava, súčin súčet, atď.), Relačné (<, >, <=,...) a Priradenie (priradenie hodnoty do premennej).

1. Popíšte ako funguje **vetvenie** programu. Čo je to **výraz** ?

Výrazy sú také časti programu ktoré vykonávajú nejakú operáciu a potom vracia výsledok operácie, alebo prestavuje nejakú hodnotu.

Vetvenie – (neviem)

1. Popíšte rozdiel medzi konštrukciami **IF**, **IF-ELSE**, **IF-ELSE IF-...-ELSE**.

**IF** – sa vykoná ak je výraz pravdivý

**IF-ELSE** – IF sa vykoná ak je výraz pravdivý inak ak je nepravdivý tak ELSE

**IF – ELSE – IF** – použijeme ak chceme zreťaziť viac podmienok

1. Aké **logické operátory** poznáš a načo sa využívajú ?

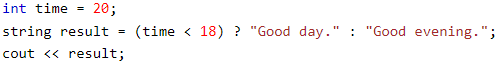
Logické (logický súčin &&, logický súčet [**||**](http://ctp.mkprog.eu/sk/c%2B%2B/logicky_sucet/) alebo negácia !)

* [Logický súčin: **&&**](http://ctp.mkprog.eu/sk/c%2B%2B/logicky_sucin/) Výsledkom je pravda len v tom prípade ak oba vstupy sú splnené, vo všetkých ostatných situáciách výsledkom je nepravda.
* [Logický súčet: **||**](http://ctp.mkprog.eu/sk/c%2B%2B/logicky_sucet/) Výsledok je nepravda keď oba vstupy sú nepravdivé, vo všetkých ostatných prípadoch je výsledok pravda
* [Negácia: **!**](http://ctp.mkprog.eu/sk/c%2B%2B/negacia/) Logický operátor negácia sa používá k určování opačné hodnoty. v našom prípade: pri pravdivé hodnoty výsledok je nepravda, a pri nepravdivom vstupu výsledok je pravda

1. Popíšte ako funguje podmienkový typ konštrukcie **switch**. Načo slúžia kľúčové slovíčka **break** a **default**?

Vetví kód podľa zadaného vstupu. Každej vetve je zodpovedná jedna **case** vetva. Vetva **default** je zodpovedná za všetky ostatné prípady kedy case nestačí. **Break** musí byť v každej vetve pretože ak ho neobsahuje vykonajú sa všetky vetvy za prvou vetvou ktorá vyhovuje.

1. *Čo je to* ***ternarny operáto****r ? Uveďte príklad.*

Obrázok, na ktorom je text

Automaticky generovaný popisExistuje aj skratka if else, ktorá je známa ako **ternárny operátor,** pretože pozostáva z troch operandov. Môže sa použiť na nahradenie viacerých riadkov kódu jedným riadkom.

1. Čo je to blok kódu ? Aké sú **Globálne** a **Lokálne** premenné?

**Blok kódu** je časť kódu medzi zátvorkami {a}. Môže to byť aj celý súbor ako celok.

**Glob. Premenná** – platí v celom programe. Umiestňuje sa na začiatok súboru.

**Lok. Premenná** – je definovaná vo vnútri funkcie alebo nejakého podbloku. Platí iba medzi dvoma { } v rámci ktorých je premenná definovaná.

1. Popíšte a uveďte príklad **Post**-**incrementu** a **Pre-incrementu** ?

Post-....... – je i++ (to znamená že najskôr hodnotu i s niečím porovná alebo ju k niečomu priradí a potom zväčší o 1)

Pre-........ – je ++i (to znamená že najskôr i zväčší o 1 a potom ju s niečím porovná alebo k niečomu priradí)

1. Načo slúži **cyklus**? Aké cykly poznáte?

Slúži na to aby sa nejaká časť kódu opakovala. Poznáme while a for.

1. Popíšte význam kľúčových slov **break** a **continue**.

Break – slúži na okamžité zastavenie cyklu bez ohľadu, na to či je podmienka true alebo false.

Kód za cyklom bude pokračovať normálne ďalej.

Continue – tento príkaz okamžite zastaví aktuálne kolo cyklu a preskočí do ďalšieho kola cyklu.

1. Čo je to **pole**? Načo slúži? **Vysvetlite pojem index poľa.**

Pole je prvá dátová štruktúra preto, lebo uchováva dáta v nejakej štruktúrovanej forme – konkrétne sú dáta uložené za sebou v pamäti. Pomocou poľa môžeme uchovávať viac ako jednu hodnotu. Poradie (index) – každá priehradka má svoje poradie – index – začínajúc od nuly.

1. Ako efektívne **prechádzať poľom** **ak nepoznáme jeho veľkosť** ?

Pre výpočet potrebujeme natívnu funkciu z céčka: sizeof()

sizeof() vracia veľkosť v bitoch zadaného vstupu

1. Čo je to **rozmer** **poľa** ? Aké je to **viacrozmerné** **pole** ? Ako ho **zapisujeme**, **prechádzame**?

Ide vlastne o ukladanie polí do jednotlivých prvkov iného poľa (pole v polí). Najčastejšie sa stretneme s 2D a 3D poliami. Polia o viac rozmeroch sa vyskytujú menej často najmä preto, že ľudská myseľ nevie s nimi tak dobre pracovať.

**Obrázok, na ktorom je text

Automaticky generovaný popis**

1. Popíšte načo slúži **ASCII tabuľka** a ako s ňou môžeme v C++ pracovať.

Dané znaky sú reprezentované ako čísla od 0 di 255. (dnes nepoužiteľné pretože používame viac ako 255 znakov). Znaky sú v tabuľke uložené za sebou v abecedne.

Obrázok, na ktorom je text

Automaticky generovaný popis

1. Popíšte ako **spájame dva stringy dohromady**, ako môžeme **meniť jednotlivé znaky** a ako zistím **dĺžku stringu**.

Spájanie: Dĺžka:

Obrázok, na ktorom je text

Automaticky generovaný popis Obrázok, na ktorom je text

Automaticky generovaný popis

Obrázok, na ktorom je text

Automaticky generovaný popisMenenie: