Prosty Program

Program w C++ wymaga dyrektyw preprocesora `#include` i funkcji `main()`, która może się kończyć instrukcją `return 0;`

```
#include <io str eam>
using namespace std;
int main()
{
    cout << " Witaj Świeci e!";
    return 0;
}
```

Dyrektywa Preprocesora

`#include <na zwa _pl iku>` - wstawia treść pliku nagłówkowego biblioteki

`#include "nazwa _pliku"` - wstawia treść pliku nagłówkowego użytkownika

Przydatne pliki bliblioteki

`iostream`	Obiekty `cin` i `cout`			
`cmath`	Funkcje matematyczne takie jak			
	`pow`	`powf`	`powl`	Potęgowanie
	`sqrt`	`sqrtf`	`srtl`	Pierwiastkowanie
	`sin`	`cos`	`tan`	Trygonometryczne
	`sinh`	`cosh`	`tanh`	Trygonometryczne hiperboliczne
	`asin`	`acos`	`atan`	Trygonometryczne odwrócone
	`floor`	`ceil`	`round`	Zaokrąglanie
	`abs`	`fabs`		Wartość absolutna
	`exp`			Funkcja ekspotencjalna
	Oraz stałe takie jak			
	`M_PI`	Stała PI	`M_E`	Stała Eulera

Przydatne pliki bliblioteki (cont)

`limits`	Właściwości typów numerycznych
`cstdlib`	Funkcje do wartości bezwzględnej oraz generatora liczb pseudolosowych
	`abs` `srand` `rand`

Liczba pseudolosowa

```
int losowa = rand() % max + min
```

Czyli

```
#include <cs tdl ib>

srand (time( nul lptr));  // inicja lizacja ziarna
v1 = rand() % 100;      // 0-99
v2 = rand() % 100 + 1;  // 1-100
v3 = rand() % 30 + 970; // 970-999
```

Komentarze

```
// w jednej linii
/* w wielu
      liniach */
```

Blok instrukcji

Zbiór instrukcji ujętych w instrukcji grupującej jest traktowany jak jedna instrukcja. Taka instrukcja umożliwia deklarowanie danych lokalnych, widocznych tylko w jej obrębie. Stosowana jest głównie w warunkach logicznych i pętlach.

```
{
     instrukcja 1;
     instrukcja 2;
     ...
     instrukcja n;
}
```

Instrukcja warunkowa if / else if / else

`if(war une k_l ogi czny)` `  instru kcja;`	Wykonuje pojedynczą instrukcję jeśli warune-k_logiczny jest spełniony (jeśli warune-k_logiczny zwraca wartość `true`).

`if(war une k_l ogi czny) {`
`  instru kcja1;`
`  ...`
`  instru kcjaN;`
`}`
	Wykonuje blok instrukcji jeśli warune-k_logiczny jest spełniony. W przypadku jednej instrukcji do wykonania nawiasy klamrowe można pominąć.

`if(war une k_l ogi czny) {`
`  instru kcje1;`
`} else {`
`  instru kcje2;`
`}`
	Wykonuje blok instru-kcje1 jeśli warunek_l-ogiczny jest spełniony, a w przeciwnym wypadku (warunek logiczny jest niespe-łniony, czyli ma wartość `false`) wykonuje blok instrukcje2.

`if(war une k_l ogi czny_A) {`
`  instru kcje1;`
`} else if(war une k_l ogi czny_B) {`
`  instru kcje2;`
`} else {`
`  instru kcje3;`
`}`
	Sprawdza warunek_l-ogiczny_A i jeśli jest spełniony to wykonuje instrukcje1. Jeśli nie jest spełniony to sprawdza warunek_l-ogiczny_B, który jeśli jest spełniony to wykonuje instrukcje2. W przeciwnym wypadku (oba warunki nie są spełnione) wykonuje instrukcje3.

Uwagi

Instrukcja warunkowa if / else if / else (cont)

1. C/C++ traktuje 0 jako `false` a każdą inną wartość jako `true`.	**Źle** `if(0) {...}` Nigdy nie wejdzie do środka
2. Nie stawiaj średnika po nawiasach po if ()!	**Źle** `if(!ko niec); {` Średnik po if, czyli pusta instrukcja.
3. Operator przypisania `=` zamiast porównania `==`	**Źle** `if(a=5)` - warunek zawsze spełniony bo po przypisaniu `if(5)` (patrz uwaga 1).

Przykłady

```
int x = 20;
if (x > 5)
    cout << "x jest większe od 5" << endl;
if(x > 10) {
   cout << "x jest większe od 10" << endl;
} else {
    cout << "x jest mniejsze bądź równe 10";
}
if (x < 20)) {
    cout << "x jest mniejsze od 20";
} else if (x == 20) {
    cout << "x jest równe 20";
} else {
    cout << "x jest większe od 20";
}
```

Insstrukcja wyboru switch / case

Pętle

Pętla `for`

``