Temat: Funkcje w języku C++

Funkcja (ang. function) jest fragmentem kodu programu, który możemy wielokrotnie wywoływać z różnych miejsc programu. Każda funkcja przed wykorzystaniem musi być zdefiniowana. Definicja funkcji wygląda następująco:

typ_wyniku nazwa_funkcji(*lista_argumentów*)
{
  treść funkcji
}

typ_wyniku - określa rodzaj informacji, którą zwraca funkcja jako wynik swojej pracy. Jeśli funkcja nie zwraca wyniku (np. interesuje nas tylko wykonanie określonego kodu), to posiada typ void (ang. pusty).

nazwa_funkcji - zbudowana podobnie jak nazwa zmiennej - stosujemy identyczne reguły. Nazwa funkcji umożliwia odwołanie się do jej kodu.

lista_argumentów - zawiera definicje danych przekazywanych do funkcji. Lista zbudowana jest z wpisów:

typ_argumentu1 nazwa_argumentu1, typ_argumentu2 nazwa_argumentu2, ...

typ_argumentu - jeden z typów danych zdefiniowanych w C++ (np. int, double, bool) lub typ zdefiniowany przez użytkownika.

nazwa_argumentu - zbudowana identycznie jak nazwa zmiennej. Pozwala odwoływać się do danego argumentu wewnątrz funkcji.

Przykład:

```
#include <iostream>
using namespace std;
int suma(int x,int y)
{
  return x + y;
}
int main()
{
  cout << suma(2,7) << endl
     << suma(9,-3) << endl;
  return 0;
}
```

Zmienne lokalne funkcji

Każda funkcja może posiadać swój własny zestaw zmiennych, które nazywamy **zmiennymi lokalnymi** (ang. local variables). Czas życia zmiennej lokalnej ogranicza się do funkcji, w której zostały zdefiniowany. W różnych funkcjach mogą istnieć zmienne o takich samych nazwach, jednakże są to różne zmienne i nie wpływają na siebie nawzajem.

Poniższy program demonstruje izolację zmiennych lokalnych. W funkcjach main() i f() tworzymy zmienną lokalną o nazwie a. Zmiennej a przypisujemy w main() wartość 29, po czym wyświetlamy zawartość a. Następnie zostaje wywołana funkcja f(), która tworzy swoją zmienną a, przypisuje jej wartość 5 i wyświetla ją. Po powrocie z funkcji f() do main() zostaje ponownie wyświetlona zmienna a należąca do main(). Zwróć uwagę, że wywołanie f() nie zmieniło wartości tej zmiennej - wciąż jest w niej liczba 29. Funkcja f() operowała na swojej lokalnej zmiennej a, a nie na zmiennej a należącej do main().

Przykład:

```
#include <iostream>
using namespace std;
void f()
{
   int a;
   a = 5;
   cout << a << endl;
}
int main()
{
   int a;
   a = 29;
   cout << a << endl;
   f();
   cout << a << endl;
   return 0;
}
```

Przekazywanie argumentów do funkcji

W języku C++ argumenty możemy przekazywać na dwa sposoby:

Przekazywanie przez wartość

W tym przypadku funkcja otrzymuje wartości jako argumenty. Ten typ przekazywania argumentów do funkcji pozwala stosować przy wywołaniu wyrażenia w miejscu argumentów. Komputer oblicza wartość danego wyrażenia i przekazuje ją funkcji w danym argumencie. Funkcja otrzymaną wartość może dowolnie modyfikować.

Poniższy program zawiera prostą funkcję pisz2(), która wymaga argumentu x typu int. Otrzymany argument funkcja zwiększa o 2, a następnie wyświetla w oknie konsoli. W funkcji main() tworzymy jedną zmienną a, której nadajemy wartość 5 i wyświetlamy w oknie konsoli. Następnie wywołujemy funkcję pisz2() z argumentem a. Po powrocie z funkcji pisz2() wyświetlamy ponownie zawartość a. Zwróć uwagę, że funkcja pisz2() zmodyfikowała jedynie wartość otrzymanego argumentu - zmienna a w funkcji main() pozostała nienaruszona.

Przykład:

```
#include <iostream>
using namespace std;
void pisz2(int x)
{
   x += 2;  // zwiększamy argument o 2
   cout << x << endl;
}
int main()
{
   int a;
   a = 5;
   cout << a << endl;
   pisz2(a);
   cout << a << endl;
   return 0;
}
```

Przekazywanie przez referencję

W tym przypadku przed nazwą argumentu na liście umieszczamy operator &. Funkcja otrzymuje jako argument adres zmiennej, a nie jej wartość. Mając adres może odwołać się do pamięci tej zmiennej i zmienić jej zawartość. Wszelka modyfikacja takiego argumentu wewnątrz funkcji powoduje zmianę skojarzonej z tym argumentem zmiennej.

W poniższym programie zmieniamy w funkcji pisz2() adresowanie przez wartość na adresowanie przez referencję. Teraz zmiana argumentu wewnątrz funkcji pisz2() odbije się automatycznie na zmianie zawartości zmiennej a, którą udostępniliśmy jako argument wywołania pisz2().

Przykład:

```
#include <iostream>
using namespace std;
void pisz2(int & x)
{
   x += 2;  // zwiększamy argument o 2
   cout << x << endl;
}
int main()
{
   int a;
   a = 5;
   cout << a << endl;
   pisz2(a);
   cout << a << endl;
   return 0;
}
```

Ćwiczenia:

1. Napisz program zawierający funkcje NWD, NWW, Dzielenie (wynik całkowity = reszta z dzielenia).
   Napisz dwie wersje:
   a) Przekazywanie zmiennych przez wartość,
   b) Przekazywanie zmiennych przez referencję.

Kod programów prześlij na adres: marek@zstio-elektronika.pl
Temat wiadomości: Klasa……-Imię Nazwisko-funkcje
Termin: 22-03-2020 godz: 12:00