Lista zadań – Programowanie

# Wprowadzenie

**Funkcja** – podprogram wykonujący pewne zadanie na potrzeby programu głównego. Jest to fragment programu, któremu nadano nazwę i który możemy wykonać przez podanie jego nazwy oraz ewentualnych argumentów (o ile istnieją).

Każda funkcja musi posiadać następujące własności:

* Posiadać nazwę,
* Zwracać wartość lub nie, jeśli tego nie chcemy,
* Mieć dowolną liczbę argumentów.

Funkcja może pobierać argumenty z programu głównego aby wykonać określone zadania.

Jeżeli funkcja zwraca wartość to należy określić jej typ. Sposób wywołania funkcji zależy od jej typu. Każda funkcja, która zwraca wartość, musi być wywołana przy jedno czesnym przypisaniu danej wartości pewnej zmiennej globalnej.

**Funkcja która nie zwraca wartości:**

void wypisz() // brak danych w nawiasach = funkcja nie przyjmuje argumentów

{

//ciało funkcji – kod który zostanie wykonany w momencie wywołania funkcji

}

1. Definicje funkcji umieszczamy przed funkcją główną (**main**) lub w oddzielnym pliku.
2. Zdefiniowane funkcje wywołujemy w funkcji głównej **main**.
3. Aby wywołać funkcje, która nie zwraca żadnych wartości wystarczy w funkcji głównej umieścić jej nazwę i puste nawiasy np. wypisz().

**Funkcja mająca argumenty:**

Wartością zwrotną funkcji jest wartość, która została wyliczona. Następnie jest ona przekazywana do dalszych działań. Typ danej, która ma być zwracana jest wypisywany przed definicją funkcji i jest ona zwracana za pomocą słowa kluczowego **return**.

typ\_zwracanej\_wartosci nazwa\_funkcji(lista\_argumentow)

{

return zwracana\_wartosc;

}

Argumenty funkcji określają, jaki typ mają mieć zmienne, które przekazujemy do funkcji podczas jej wywołania.

Argumenty funkcji oraz zmienne, które są określane w jej ciele mają zasięg **lokalny** – są widoczne tylko przez funkcję.

**Wywoływanie funkcji i przekazywanie argumentów do funkcji**

Aby wywołać funkcję należy napisać jej nazwę w bloku instrukcji głównej – main.

Zmienne, które znajdują się w funkcji głównej i są wykorzystywane w funkcji są nazywane **zmiennymi globalnymi**.

Jeżeli podczas wywoływania funkcji podamy złą liczbę argumentów, wówczas zostanie zwrócony błąd. Dlatego w języku c++ możemy posługiwać się **argumentami domniemanymi czy parametrami domyślnymi/opcjonalnymi**, czyli takimi, które mogą zostać podane w wywoływaniu funkcji lub nie.

void wyświetl\_pi(ploat pi=3.14159)

{

cout<<”Stała pi= „<<pi<<endl;

}

Wszelkie argumenty domniemane trzeba umieszczać na końcu. Nie może dojść do sytuacji, że pierwszy argument będzie domniemany, a pozostałe już nie.

**Prototyp** (deklaracja) funkcji pełni w stosunku do funkcji taką rolę jak deklaracja do zmiennej – informuje jakie typy są używane. W przeciwieństwie do definicji funkcji, prototyp jest zakończony średnikiem. Gdyby nie było średnika, kompilator zinterpretowałby dany wiersz jako nagłówek funkcji i oczekiwałby jej dalszej treści.

void parzysta (int a) ; // prototyp funkcji

int main()

{

int liczba;

cot<<”podaj liczbe”<<endl;

cin>>liczbal;

parzysta(liczba); /// wywołanie funkcji

return 0;

}

void parzysta(int a) // definicja funkcji

{

if(a%2==0)

cout<<”Parzysta”<<endl;

else

cout<<”Nieparzysta”<<endl;

}

W powyższym przykładnie mamy do czynienia z **przekazywaniem argumentów przez wartość**. Polega ona na tym, że w momencie przekazywania zmiennych jako argumentów, funkcja tworzy ich kopie w pamięci. Oznacza to, że wszystkie zmiany jakie zostaną dokonane na wartościach zmiennych nie będą widoczne w miejscu, w którym zostały przekazane do funkcji. Dodatkowo nazwy tych zmiennych mogą być takie same jak nazwy argumentów funkcji.

Można również zastosować **przekazywanie argumentów przez referencje**, dzięki czemu możemy zmniejszyć ilość miejsca jakie w pamięci zajmują dane i wówczas nie będziemy niepotrzebnie tracić mocy obliczeniowej na tworzenie ich kopii. Wystarczy wówczas użyć symbolu & za typem zmiennej a przed jej nazwą, np.: int &zmienna;. Dane przekazane przez referencje nie są kopiowane czyli pracuje się na oryginalnych danych. W praktyce oznacza to, że modyfikacja wartości zmiennej zmodyfikuje nam tę zmienną, która została przekazana do funkcji przez argument.

**Przeładowanie funkcji** polega na tym, że w danym programie występuje więcej niż jedna funkcja o tej samej nazwie. To, która z nich zostanie wywołana zależy od argumentów, z którymi zostanie ona wywołana. Np. pole koła potrzebuje tylko jednego argumentu, natomiast pole trójkąta już dwóch.

float pole(float promien);

float pole(float dl, float szer);

# Zadania

**Korzystając z funkcji**

1. Napisz program wyświetlający Twoje imię i nazwisko.
2. Napisz program, który prosi o podanie odległości w milach morskich i zamienia ją na metry (jedna mila morska to 1852 metry).
3. Napisz program, który prosi użytkownika o podanie wieku, a następnie wyświetla wiek użytkownika wyrażony w miesiącach.
4. Napisz program, który pobiera od użytkownika temperaturę w stopniach Celsjusza i zwraca odpowiednią jej temperaturę w skali Farenheita. Zmianę temperatury przeprowadź zgodnie ze wzorem: Farenheit = 1,8 \* stopnie Celsjusza + 32,00.
5. Napisz program sprawdzający czy z trzech liczb podanych przez użytkownika można zbudować trójkąt prostokątny. Funkcja ma być typu void i mieć trzy argumenty typu float.
6. Napisz program zawierający cztery funkcje wykonujące cztery podstawowe działania arytmetyczne. Każda z funkcji ma zwracać wartość typu rzeczywistego i pobierać dwa argumenty typu float.
7. Napisz program, który pobierze z klawiatury trzy wartości, przypisze je pewnym zmiennym, które przekaże jako argumenty pewnej funkcji. Jeśli wartość pierwszej zmiennej pomnożonej przez wartość drugiej zmiennej jest większa od 100-krotności trzeciej zmiennej, to wynikiem funkcji będzie podwojona wartość pierwszej zmiennej, w przeciwnym razie wynikiem funkcji będzie wartość średnia arytmetyczna trzech zmiennych.
8. Napisz program, który prosi użytkownika o podawanie dwóch liczb tak długo, aż jedna z nich będzie zerem. Dla każdej pary liczb program ma wyliczyć ich średnią harmoniczną. Średnia harmoniczna liczb to odwrotność średniej ich odwrotności, wyliczana następująco: średnia harmoniczna = 2.0\*x\*y/(x+y).
9. Napisz funkcję, która stwierdza, czy zadana jako parametr liczba całkowita jest kwadratem pewnej liczby całkowitej. Liczby będące kwadratami liczb całkowitych to 1, 4, 9, 16 itd. Wartością funkcji ma być jeden, jeśli liczba spełnia warunek oraz zero w przeciwnym wypadku.
10. Napisz funkcję, która znajduje w tablicy element maksymalny. Parametrami funkcji mają być: tablica tab oraz liczba elementów tablicy – n. Wartością funkcji ma być element maksymalny.
11. Napisz funkcję, która ma trzy parametry formalne a, b, c będące liczbami całkowitymi. Wartością funkcji jest jeden, jeśli zadane liczby są liczbami pitagorejskimi oraz zero w przeciwnym wypadku. Liczby pitagorejskie spełniają warunek: a\*a+b\*b=c\*c.
12. Napisać funkcję o nazwie wybraneLiczby, która pobiera dwie liczby całkowite dodatnie reprezentujące końce przedziału liczbowego i zwraca ilość liczb całkowitych z tego przedziału, które są podzielne przez 3 i 4.
13. Napisać dwie funkcje o nazwach pole, obwod, które pobierają dwie liczby rzeczywiste reprezentujące długości boków prostokąta i zwracają odpowiednio jego pole oraz obwód. Przetestować te funkcje w przykładowym programie.
14. Napisać funkcję o nazwie potega, która oblicza n-tą potęgę liczby 2. Funkcja pobiera wykładnik n i zwraca potęgę liczby 2 o wykładniku n. Przetestować tę funkcję w przykładowym programie.
15. Napisz funkcję, która zwraca największą wartość przechowywaną w tablicy jednowymiarowej. Przetestuj ją w prostym programie (użyj tablicy alokowanej dynamicznie, wypełnionej liczbami losowymi z przedziału <1, 100>).
16. Napisać funkcję pole, która pobiera trzy liczby rzeczywiste a, b, c reprezentujące długości boków trójkąta i zwraca jego pole obliczone ze wzoru Herona: sqrt(p\*(p-a)\*(p-b)\*(p-c)), gdzie p jest połową obwodu trójkąta. Za pomocą tej funkcji obliczyć pole przykładowego trójkąta.