

Ćwiczenia – lista zadań nr 6

Zadanie 1

Przygotuj krótkie programy ilustrujące odpowiedzi na poniższe pytania:

- a) Czy w języku C/C++ można zdefiniować dwie funkcje o takiej samej nazwie? Jak to się powinno zrobić ? (jeżeli można)
- b) Co się stanie jeżeli w funkcji zwracającej wartość (typu innego niż `void`) zapomnimy napisać polecenia `return` wewnątrz kodu funkcji?
- c) Czym się różni zakończenie funkcji poleceniem
`return(0);`
od
`exit(0);`?
- d) Czym się różnią:
 - definicja funkcji,
 - wywołanie funkcji,
 - prototyp funkcji.

Jakie informacje muszą być w nich zawarte (np. w prototypie funkcji), a które są opcjonalne, tzn. mogą zostać pominięte?

- e) W jaki sposób przekazujemy do/z funkcji argumenty typu tablicowego? Czy tablica może być przekazana do funkcji „przez wartość” (tzn. że funkcja korzysta z kopii tablicy)?
- f) Czy poprawne są następujące deklaracje:
`void funkcja_A(int tablica[10]);`
`void funkcja_B(int tablica[]);`
`void funkcja_C(int [10]);`
`void funkcja_D(int []);`
`void funkcja_E([]);`
- g) Czym się różnią zmienne definiowane wewnątrz ciała funkcji (automatyczne, lokalne) od zmiennych definiowanych poza funkcjami (globalne) ?
- h) Jaki znaczenie mają specyfikatory `static` i `register` występujące przed definicjami zmiennych wewnątrz funkcji?
- i) Jaki znaczenie ma słowo kluczowe `inline` występujące w linii definiującej nagłówek funkcji? Kiedy należy je wykorzystywać, a kiedy jest to niewskazane?
- j) Jakie znaczenie mają argumenty funkcji głównej:
`int main(int argc, char** argv) ?`
Jak można je wykorzystać ?

Zadanie 2

Na portalu w pliku `param.cpp` jest pomocniczy program, który ilustruje różnice w sposobie przekazywania parametrów do funkcji. Przeanalizuj ten program. Zwróć szczególną uwagę na sposób deklaracji parametru w nagłówku funkcji oraz sposób wywołania funkcji. Przeanalizuj komunikaty, które zostaną wypisane na ekranie po uruchomieniu tego programu.

```

#include <stdio.h>
#include <conio.h>

void funkcja_1(int war);    // parametr przekazywany przez WARTOSC
                           // (C/C++)
void funkcja_2(int *wsk);  // parametr przekazywany przez WSKAZNIK
                           // (C/C++)
void funkcja_3(int &ref);  // parametr przekazywany przez REFERENCJE
                           // (tylko C++)

int main()
{   int PAR;

    PAR = 1;
    printf("PRZEKAZYWANIE PARAMETRU PRZEZ WARTOSC: (C/C++)\n");
    printf(" main: PAR = %d  (przed wywołaniem funkcji 1)\n", PAR);
    funkcja_1( PAR );
    printf(" main: PAR = %d  (po wywołaniu funkcji 1)\n", PAR);
    getch(); printf("\n\n");

    PAR = 2;
    printf("PRZEKAZYWANIE PARAMETRU PRZEZ WSKAZNIK: (C/C++)\n");
    printf("czyli przekazywanie przez wartosc adresu parametru\n");
    printf(" main: PAR = %d  (przed wywołaniem funkcji 2)\n", PAR);
    funkcja_2( &PAR );
    printf(" main: PAR = %d  (po wywołaniu funkcji 2)\n", PAR);
    getch(); printf("\n\n");

    PAR = 3;
    printf("PRZEKAZYWANIE PARAMETRU PRZEZ REFERENCJE: (tylko C++)\n");
    printf(" main: PAR = %d  (przed wywołaniem funkcji 3)\n", PAR);
    funkcja_3( PAR );
    printf(" main: PAR = %d  (po wywołaniu funkcji 3)\n", PAR);
    getch();
}

void funkcja_1(int war)
{
    printf("      funkcja_1: war = %d  (przed modyfikacja)\n", war);
    war += 10;
    printf("      funkcja_1: war = %d  (po modyfikacji)\n", war);
}

void funkcja_2(int *wsk)
{
    printf("      funkcja_2: *wsk = %d  (przed modyfikacja)\n", *wsk);
    *wsk += 10;
    printf("      funkcja_2: *wsk = %d  (po modyfikacji)\n", *wsk);
}

void funkcja_3(int &ref)
{
    printf("      funkcja_3: ref = %d  (przed modyfikacja)\n", ref);
    ref += 10;
    printf("      funkcja_3: ref = %d  (po modyfikacji)\n", ref);
}

```

Zadanie 3

Jaki będzie wyniki działania poniższego programu? Wyjaśnij dlaczego.

```
#include <stdio.h>
```

```
void funkc(int * j, int n) //definicja funkcji
{ *j = 1;
  n = *j;
}
```

```
main()
{ int i = 2, k = 3;
  funkc(&i, k); //wywołanie funkcji
  printf("i=%d, k=%d", i, k);
}
```