Wykład 07 w dniu 12.XI.08r

Przekazywanie argumentów przez wartość w funkcjach C:

Plik: `w07-1.c`
Zmiana wartości argumentu wewnątrz funkcji nie jest „widoczna” zewnątrz jej treści.

```
1  #include <stdio.h>
2
3  void fA(int r)
4  {
5      printf("  fA: %3i\n",r);
6      r = r+100;
7  }
8
9  int main()
10 {
11     int i;
12
13     i = 77;
14     fA(i);
15     printf("main: %3i\n",i);
16     return 0;
17 }
```

działanie:

```
12:01:12 chaos:~/OUT$ gcc -Wall w07-1.c -o a
12:01:36 chaos:~/OUT$ ./a
  fA:  77
main:  77
12:01:39 chaos:~/OUT$
```

Plik: `w07-2.c`
Teraz już „widzimy” z zewnątrz zmianę wartości argumentu funkcji.

```
1  #include <stdio.h>
2
3  void fA(int *r)
4  {
5      printf("  fA: %3i\n",*r);
6      *r = *r+100;
7  }
8
9  int main()
10 {
11     int i;
12
13     i = 77;
14     fA( &i );
15     printf("main: %3i\n",i);
16     return 0;
17 }
```

działanie:

```
12:13:35 chaos:~/OUT$ gcc -Wall w07-2.c -o a
12:13:45 chaos:~/OUT$ ./a
  fA:  77
main: 177
12:13:47 chaos:~/OUT$
```

Plik: w07-3.c
Deklaracja jednowymiarowej tablicy.

```
1  #include <stdio.h>
2
3  const int N=25;
4
5  int main()
6  {
7     double a[N];
8     int i,n;
9
10    n = 3;
11    for(i=0 ; i<n ; i=i+1)
12    {
13       a[i] = -1.00;
14    }
15    for(i=0 ; i<n ; i=i+1)
16    {
17       printf("%7.2lf ",a[i]);
18    }
19    printf("\n");
20    return 0;
21 }
```

działanie:

```
12:28:42 chaos:~/OUT$ gcc -Wall w07-3.c -o a
12:28:51 chaos:~/OUT$ ./a
  -1.00   -1.00   -1.00
12:28:53 chaos:~/OUT$
```

Plik: w07-4.c
Przekazanie jednowymiarowej tablicy jako argumentu funkcji.

```
1  #include <stdio.h>
2
3  const int N=25;
4
5  void nadaj(int n,double *r)
6  {
7     int i;
8
9     for(i=0 ; i<n ; i=i+1)
10    {
11       r[i] = (i+1)*0.1;
12    }
13 }
```

```
14
15 int main()
16 {
17    double a[N];
18    int i,n;
19
20    n = 3;
21    for(i=0 ; i<n ; i=i+1)
22    {
23       a[i] = -1.00;
24    }
25 // nadaj(n , a);
26    nadaj(n , &a[0]);
27    for(i=0 ; i<n ; i=i+1)
28    {
29       printf("%7.2lf ",a[i]);
30    }
31    printf("\n");
32    return 0;
33 }
```

działanie:

```
12:38:11 chaos:~/OUT$ gcc -Wall w07-4.c -o a
12:38:20 chaos:~/OUT$ ./a
   0.10    0.20    0.30
12:38:22 chaos:~/OUT$
```

Plik: w07-5.c
Przekazanie dwuwymiarowej tablicy jako argumentu funkcji.

```
1  #include <stdio.h>
2
3  const int N=25;
4
5  void okresl(int n,int m, double r[][N])
6  {
7     int i,j;
8
9     for(i=0 ; i<n ; i=i+1)
10    {
11       for(j=0 ; j<m ; j=j+1)
12       {
13          r[i][j] = (i+1)*10.0+(j+1)*0.1;
14       }
15    }
16 }
17
18 int main()
19 {
20    double a[N][N];
21    int i,j,n,m;
22
23    n = 2;
24    m = 3;
25    for(i=0 ; i<n ; i=i+1)
26    {
```

```
27          for(j=0 ; j<m ; j=j+1)
28          {
29              a[i][j] = -1.00;
30          }
31      }
32      okresl(n, m, a);
33      for(i=0 ; i<n ; i=i+1)
34      {
35          for(j=0 ; j<m ; j=j+1)
36          {
37              printf("%7.2lf ",a[i][j]);
38          }
39          printf("\n");
40      }
41      printf("\n");
42      return 0;
43  }
```

działanie:

```
12:53:27 chaos:~/OUT$ gcc -Wall w07-5.c -o a
12:53:31 chaos:~/OUT$ ./a
  10.10   10.20   10.30
  20.10   20.20   20.30

12:53:33 chaos:~/OUT$
```

.....

W jaki sposób jeszcze można używać funkcji:

Plik: `w07-6.c`

W języku `C` funkcja może być użyta jako argument przy wywołaniu innej funkcji. Będzie to nam potrzebne np. w algorytmach sortujących. W bibliotece `stdlib.h` jest procedura

```
void qsort(void *base, size_t nmemb, size_t size,
                                        int(*compar)(const void *, const void *));
```

gdzie argument `compar` wskazuje na funkcję określającą relację porządkującą.

```
1   #include <stdio.h>
2
3   int r10(int r)
4   {
5       r = 10*r;
6       return r;
7   }
8
9   int d100(int r)
10  {
11      r = r+100;
12      return r;
13  }
14
15  int fn(int x,int (*op)(int))
16  {
17      x = (*op)(x)+1000;
18      return x;
19  }
20
```

```
21 int main()
22 {
23    int w,x;
24
25    x = -1;
26    w = fn(x , r10);
27    printf("fn z  r10: %5i\n",w);
28    w = fn(x , d100);
29    printf("fn z d100: %5i\n",w);
30    return 0;
31 }
```

działanie:

```
13:25:05 chaos:~/OUT$ gcc -Wall w07-6.c -o a
13:25:15 chaos:~/OUT$ ./a
fn z  r10:   990
fn z d100:  1099
13:25:17 chaos:~/OUT$
```

Plik: w07-7.cc $\Longleftarrow$ to jest C++ uważać czym się kompiluje !

W programowaniu obiektowym w języku C++ metody mają bardzo dużo wspólnego z funkcjami języka C.

```
1  /*  K.T. 19.XI.06r przyklad w C++ ilustrujacy
2           dalsza historie funkcji wlasnych C
3      kompilacja na chaosie:
4      g++ -Wall w07-7.cc -o a
5  */
6  #include <iostream>
7  #include <string>
8
9  using namespace std;
10
11 class CAbc
12 {
13   private:
14     string _dat;
15   public:
16     CAbc()
17     {
18        _dat = "Wartosc: domyslna";
19     }
20     CAbc(string const &dat)
21     {
22        _dat = "Wartosc: " + dat;
23     }
24     ~CAbc()
25     {
26     }
27     void wyswietl()
28     {
29        cout << _dat << endl;
30     }
31 };
32
```

```
33 int main()
34 {
35    CAbc X,Y("Cos wpisalem");
36
37    X.wyswietl();
38    Y.wyswietl();
39    return 0;
40 }
```

działanie:

```
13:39:00 chaos:~/OUT$ g++ -Wall w07-7.cc -o a
13:39:33 chaos:~/OUT$ ./a
Wartosc: domyslna
Wartosc: Cos wpisalem
13:39:36 chaos:~/OUT$
```

.....

Wrocław, dnia 13/11/2008