



## **Wykład: 3**

**Budowa programu**  
**Operacje we/wy**  
**Instrukcje wyboru**



**Kod źródłowy** - program napisany w języku takim jak Pascal lub C++, czyli w języku algorytmicznym - czytelny dla programisty

**Kod wynikowy** - program zapisany jako ciąg rozkazów i danych w kodzie maszynowym procesora (w postaci czytelnej dla komputera), najczęściej w postaci liczb kodu dwójkowego

## Proces tworzenia programu:

- |              |   |                |                                       |
|--------------|---|----------------|---------------------------------------|
| • edytor     | - | ( *.cpp )      | kod źródłowy                          |
| • kompilator | - | ( obj )        | kod wynikowy                          |
| • Linker     | - | ( *.exe )      | kod wynikowy połączony z bibliotekami |
| • debugger   | - | ( step/watch ) | śledzenie działania, usuwanie błędów  |

## Wybrane środowiska programistyczne dla C++

- Dev C++



- Code::Blocks



- MS Visual Studio





## Budowa programu

## Pierwszy program (w języku C)

```
1  #include <conio.h>
2  #include <cstdio>
3
4  using namespace std;
5
6  int main()
7  {
8      printf("To nasz pierwszy program w C\n");
9      getch();
10     return 0;
11 }
```

# Pierwszy program (w języku C++)

```
1  #include <iostream>
2  #include <conio.h>
3
4  using namespace std;
5
6  int main()
7  {
8      cout << "Hello world!" << endl;
9      getch(); //zatrzymuje działanie programu
10     return 0;
11 }
```

# Budowa programu

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      return 0;
8  }
```

Dołączanie plików nagłówkowych bibliotek

Przestrzeń nazw

return...; gdzie za kropki wstawiamy dowolną liczbę - kod wyjścia programu.

Funkcja main() - główna funkcja programu



**Int main( )** – w języku C i C++ nie ma „programu głównego” jest za to funkcja o nazwie main( ) która wykonywana jest zawsze jako pierwsza.

Każdy program musi posiadać funkcję main( )



## Biblioteki standardowe

---

Aby skorzystać z funkcji należy dodać plik nagłówkowy biblioteki standardowej C++ zawierający jej deklaracje.

Używając w tym celu dyrektywy:

**#include** < >

Przykładowo, aby skorzystać z funkcji `cout` należy na początku programu napisać `#include <iostream>`.

*Obecnie, korzystając z bibliotek klasycznego C, zalecane jest używanie nazw bibliotek poprzedzonych literą c. Czyli w powyższym przypadku należało by napisać `#include <cstring>` zamiast `#include <string>`*

## Biblioteki standardowe

Oto pełna lista standardowych plików nagłówkowych C++:

<algorithm>	<cstdint>	<ios>	<ostream>
<bitset>	<cstdlib>	<iosfwd>	<queue>
<cassert>	<cstdio>	<iostream>	<set>
<cctype>	<stdlib>	<istream>	<sstream>
<cerrno>	<cstring>	<iterator>	<stack>
<cfloat>	<ctime>	<limits>	<stdexcept>
<ciso646>	<cwchar>	<list>	<streambuf>
<climits>	<cwctype>	<locale>	<string>
<clocale>	<deque>	<map>	<typeinfo>
<cmath>	<exception>	<memory>	<utility>
<complex>	<fstream>	<new>	<valarray>
<csetjmp>	<functional>	<numeric>	<vector>

## Biblioteki - Operacje wejścia-wyjścia

---

**Operacje wejścia-wyjścia** to podstawowe operacje tzw. komunikacji strumieniowej. Należą do tego głównie operacje na plikach i strumieniach standardowych.

- **iosfwd** i **ios** - z definicjami pierwotnymi
- **streambuf**, **istream**, **ostream** i **iostream**, - podstawowe klasy operujące abstrakcyjnym "io" (plus strumienie standardowe)
- **lomanip** - manipulatory strumieni
- **Fstream** - klasy operacji na plikach
- **Sstream** - klasy operacji na strumieniach tekstowych
- **Cstdio** - operacje "io" zgodne z biblioteką standardową C

## Biblioteki - Operacje na tekstach

---

**Operacje na tekstach** składają się z najróżniejszych operacji na tablicach znaków, implementowanych w różny sposób. Tu wyraźnie rozróżnia się typy tekstowe statyczne, czy też surowe (operujące tablicami surowymi i wskaźnikami) oraz typ string.

- **cctype, cwctype i cwchar**, - funkcje klasyfikujące pojedyncze znaki
- **cstring**, - funkcje do obsługi stringów surowych (tablic znaków)
- **locale, clocale** - obsługa internacjonalizacji



**Wyjątki** oczywiście mogą być dowolnego typu, ale lepiej jest trzymać się pewnej konwencji hierarchizacji typów wyjątków.

- **exception**, - definiuje podstawowe elementy wyjątków: abstrakcyjną klasę `exception` oraz funkcje `set_terminate` i `set_unexpected`.
- **stdexcept**, - definiuje standardowe klasy wyjątków
- **csetjmp**, - definiuje funkcje obsługi sytuacji wyjątkowych w stylu C

Źródło: C++ bez cholesterolu, <http://intercon.pl/~sektor/cbx/>



W językach C i C++ mamy do dyspozycji trzy rodzaje komentarzy:

- komentarz  
jednowierszowy;

```
5  int main()  
6  {  
7      // Komentarz jednowierszowy  
8      return 0;  
9  }
```

- komentarz  
wielowierszowy;

```
5  int main()  
6  {  
7      /* początek komentarza  
8         komentujemy dalej....  
9         i kończymy komentarz */  
10     return 0;  
11 }
```

- komentarz wykonany za pomocą dyrektyw preprocesora.



## Zmienne (wstęp)



Nazwa typu	Zawartość	Przedział wartości	Zajęt. pamięć
<b>char</b>	znak	$-128 \div 127$	1 bajt
<b>int</b>	liczba całkowita	$-32768 \div 32767$	2 bajty
<b>long</b>	liczba całkowita	$-2147\text{mln} \div 2147\text{mln}$	4 bajty
<b>float</b>	liczba rzeczyw.	$10^{-38} \div 10^{38}$ (7cyfr)	4 bajty
<b>double</b>	liczba rzeczyw.	$10^{-308} \div 10^{308}$ (15 cyfr)	8 bajtów



**Modyfikatory typu:**

<b>signed</b>	→	ze znakiem ( $\pm$ ),	<b><u>int</u></b>	<b><u>char</u></b>	—
<b>unsigned</b>	→	bez znaku,	<b>int</b>	<b>char</b>	—
<b>short</b>	→	krótka (mniejsza),	<b>int</b>	—	—
<b>long</b>	→	długa (większa)	<b><u>int</u></b>	—	<b>double</b>

np. **unsigned long int** *dluga\_liczba\_bez\_znaku* ;

**Wartości domyślne:**

<b>long</b>	=	<b>long int</b>
<b>int</b>	=	<b>signed int</b>
<b>char</b>	=	<b>signed char</b>



**Deklaracja zmiennej** - informuje kompilator, że dana nazwa jest znana. Jednak pamięć dla obiektu nie zostaje przydzielona. Do obiektu nie możemy się odwoływać, nie możemy mu przypisywać wartości – obiekt jeszcze nie istnieje.

**extern nazwaTypu nazwaZmiennej;**

Np.: `extern int liczba;`



**Definicja zmiennej** - rezerwuje miejsce w pamięci dla danej zmiennej. Po zdefiniowaniu ze zmiennej możemy korzystać.

**nazwaTypu   nazwaZmiennej;**

Np.:   int liczba;

Każda definicja jest jednocześnie deklaracją (ale nie odwrotnie).



**Inicjalizacja (inicjowanie) zmiennej** - polega na przypisaniu wartości do danej zmiennej w momencie jej deklaracji

**nazwaTypu   nazwaZmiennej = wartość;**

Np.:   `int liczba = 10;`



## Operacje we/wy

## Funkcje printf i scanf (proceduralnie, w C)

---

### Wysłanie informacji na zewnątrz (stand. ekran)

```
printf („lancuch formatujacy”, zmienna_1, zmienna_2);
```

### Pobranie informacji z zewnątrz (stand. klawitura)

```
scanf („prototypy zmiennych”, &zmienna_1, &zmienna_2);
```

## Funkcje printf i scanf (proceduralnie, w C)

---

### Prototypy zmiennych dla funkcji printf i scanf

- `%c` - pojedynczy znak
- `%s` - łańcuch znaków
- `%d` - liczba dziesiętna ze znakiem
- `%f` - liczba zmiennoprzecinkowa (notacja dziesiętna)
- `%e` - liczba zmiennoprzecinkowa (notacja wykładnicza)
- `%g` - liczba zmiennoprzecinkowa (krótszy z formatów `%f` `%e`)
- `%u` - liczba dziesiętna bez znaku
- `%x` - liczba w kodzie szesnastkowym (bez znaku)
- `%o` - liczba w kodzie ósemkowym (bez znaku)
- `l` - przedrostek `l` (long) stosowany przed: `d u x o`

## Funkcje printf i scanf (proceduralnie, w C)

---

### Znaki sterujące wypisywaniem tekstu (nie tylko dla printf)

`\b` - cofanie o 1 znak  
`\f` - nowa strona  
`\n` - nowa linia  
`\t` - tabulator  
`\a` - sygnał dźwiękowy

Jeśli jednak chcemy po prostu wypisać znak...

`\\` - backslash  
`\'` - apostrof  
`\0` - znak o kosie zero  
`\?` - znak zapytania



# Funkcja printf (proceduralnie, w C)

```
1  #include<stdio.h>
2  int main()
3  {
4      char znak='a';
5      float liczba = 1/3.0;
6      printf("znak = %c\nznak(dziesietnie) = %d\nznak (szesnastkowo) :
7      |      = %x\nznak (osemkowo) = %o\n", znak, znak, znak, znak);
8      printf("liczba = %f\n", liczba);
9      printf("liczba = %.1f\n", liczba);
10     printf("liczba = %10.2f\n", liczba);
11     printf("liczba = %e\n", liczba);
12     printf("liczba = %d\n", liczba);
13     return 0;
```

```
znak = a
znak(dziesietnie) = 97
znak (szesnastkowo) = 61
znak (osemkowo) = 141
liczba = 0.333333
liczba = 0.3
liczba =      0.33
liczba = 3.333333e-001
liczba = 1610612736
```

## Funkcja scanf (proceduralnie, w C)

Program wczytuje i wyświetla wartość podanej liczby całkowitej.

```
1  #include <stdio.h>
2  int main()
3  {
4      int x;
5      printf("Podaj liczbe: ");
6      scanf("%d", &x);
7      printf("Podales liczbe %d \n", x);
8      return 0;
9  }
```

```
Podaj liczbe: 8
Podales liczbe 8
```

# Funkcje printf i scanf (proceduralnie, w C)

```
1  #include <stdio.h>
2  #include <math.h>
3
4  int main()
5  {
6      double a, b, c, delta, x1, x2;
7      printf("a = "); scanf("%lf", &a);
8      printf("b = "); scanf("%lf", &b);
9      printf("c = "); scanf("%lf", &c);
10     if ((delta = b*b-4*a*c) >= 0)
11     {
12         x1 = (-b-sqrt(delta))/(2*a);
13         x2 = (-b+sqrt(delta))/(2*a);
14         printf("x1 = %lf\nx2 = %lf\n", x1, x2);
15     }
16     else
17         printf("Brak rozwiazan rzeczywistych\n");
18     return 0;
19 }
```

## Klasy cout i cin (obiektowo w C++)

---

Strumień – to najprościej mówiąc jest to ciąg bajtów o nieokreślonej długości.

Wyróżniamy trzy rodzaje strumieni:

1. **Strumień konsoli – wczytanie z klawiatury i wypisanie na ekran**
2. Strumień plikowe
3. Strumień napisów

Do obsługi strumieni służą obiekty **cin** oraz **cout**

Domyślnym strumieniem jest strumień konsoli, którym będziemy posługiwać się w tym wykładzie.

## Klasy cout i cin (obiektoowo w C++)

---

Wyprowadzenie wartości do strumienia wyjściowego (stdout)

```
cout << „tekst”;
```

```
cout << zmienna;
```

Wczytanie ze strumienia wejściowego (stdin)

```
cin >> zmienna;
```

Prototypy cin i cout znajdują się w bibliotece iostream.h

```
#include <iostream>
```

# Klasy cout i cin (obiektoowo w C++)

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hej tam.\n";
8      cout << "To jest 5: " << 5 << "\n";
9      cout << "Manipulator endl ";
10     cout << "wypisuje nowa linie na ekranie.";
11     cout << endl;
12     cout << "To jest bardzo duza liczba:\t" << 70000;
13     cout << endl;
14     cout << "To jest suma 8 i 5:\t";
15     cout << 8+5 << endl;
16     cout << "To jest ulamek:\t\t";
17     cout << (float) 5/8 << endl;
18     cout << "I bardzo, bardzo duza liczba:\t";
19     cout << (double) 7000 * 7000 << endl;
20     return 0;
21 }
```

Hej tam.

To jest 5: 5

Manipulator endl wypisuje nowa linie na ekranie.

To jest bardzo duza liczba: 70000

To jest suma 8 i 5: 13

To jest ulamek: 0.625

I bardzo, bardzo duza liczba: 4.9e+007



## Instrukcje sterujące



W języku C++ nie ma osobnych zmiennych przechowujących dane typu prawda-Fałsz.

Tę rolę pełnić może każda zmienna, wyrażenie lub funkcja , która przyjmuje (lub zwraca) wartość zero lub różną od zera.

Wartość zero -

FAŁSZ

Wartość inna niż zero -

PRAWDA



## Instrukcja warunkowa if

```
if (wyrażenie) instrukcja;
```

```
if (wyrażenie) instrukcja_1;  
else instrukcja_2;
```

```
if (wyrażenie)  
{  
    instrukcja_1;  
    instrukcja_2;  
}  
else instrukcja_3;
```

# Instrukcja warunkowa if

---

Przykład:

```
cin >> i;  
if (i!=0) cout << „i rozne od zera”;  
else cout << „i rowne zero”;
```

Można i tak:

```
cin >> i;  
if (i) cout << „i rozne od zera”;  
else cout << „i rowne zero”;
```

# Instrukcja warunkowa if - przykład

## Równanie kwadratowe

```
2  #include <stdio.h>
3  #include <conio.h>
4  #include <math.h>
5
6  using namespace std;
7
8  int main()
9  {
10     double a, b, c, delta, x1, x2;
11     printf("a = "); scanf("%lf", &a);
12     printf("b = "); scanf("%lf", &b);
13     printf("c = "); scanf("%lf", &c);
14     if ((delta = b*b-4*a*c) >= 0)
15     {
16         x1 = (-b-sqrt(delta))/(2*a);
17         x2 = (-b+sqrt(delta))/(2*a);
18         printf("x1 = %lf\nx2 = %lf\n", x1, x2);
19     }
20     else
21         printf("Brak rozwiazań rzeczywistych\n");
22     getch();
23     return 0;
24 }
```

# Instrukcja warunkowa if - przykład

Równanie kwadratowe v. 2  
– z użyciem cin i cout

```
1  #include <iostream>
2  #include <conio.h>
3  #include <math.h>
4  using namespace std;
5
6  int main()
7  {
8      double a, b, c, delta, x1, x2;
9      cout << "a = ";    cin >> a;
10     cout << "b = ";    cin >> b;
11     cout << "c = ";    cin >> c;
12     if ((delta = b*b-4*a*c) >= 0)
13     {
14         x1 = (-b-sqrt(delta))/(2*a);
15         x2 = (-b+sqrt(delta))/(2*a);
16         cout << "x1 = " << x1 << endl << "x2 = " << x2 << endl;
17     }
18     else
19         cout << "Brak rozwiazań rzeczywistych" << endl;
20     getch();
21     return 0;
22 }
```

# Instrukcja wyboru wielokrotnego switch

```
switch (zmienna)
{
    case wartosc_1: instrukcja_1; break;
    case wartosc_2: instrukcja_2; break;
    case wartosc_3: instrukcja_3; break;
    default: instrukcja_defaltowa;
}
```

# Instrukcja wyboru wielokrotnego switch

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int d;
6      cout << "Podaj nr dzien tygodnia:\t";
7      cin >> d;
8      switch (d)
9      {
10         case 1: cout<< endl << "Niedziela";      break;
11         case 2: cout<< endl << "Poniedzialek";    break;
12         case 3: cout<< endl << "Wtorek";          break;
13         case 4: cout<< endl << "Sroda";           break;
14         case 5: cout<< endl << "Czwartek";        break;
15         case 6: cout<< endl << "Piatek";          break;
16         case 7: cout<< endl << "Sobota";          break;
17         default: cout <<endl
18                 << "to juz nie w tym tygodniu";
19     }
20     return 0;
21 }
```



```
for ( instrukcja_ini ; wyrażenie_warunkowe ; instrukcja_krok )  
    tresc_petli ;
```

- **instrukcja\_ini** - instrukcja wykonywana zanim pętla zostanie poraz pierwszy uruchomiona
- **wyrażenie\_warunkowe** – wyrażenie obliczane przed każdym obiegiem pętli. Jeżeli jest ono różne od zera, to pętla będzie dalej wykonywana
- **instrukcja\_krok** – instrukcja wykonywana po zakończeniu każdego obiegu pętli

## Pętla for - przykład

```
1  #include <stdio.h>
2  #include <conio.h>
3  #include <cmath>
4
5  int main()
6  {
7      int n, k;
8      double s = 0;
9      printf("n = ");
10     scanf("%d", &n);
11     for (k=2; k<=n; k*=2)
12         s += 1.0/k;
13     printf("Suma = %lf\n", s);
14     getch();
15     return 0;
16 }
17
```

$$x_n = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots + \frac{1}{2^n}$$



## Literatura:

---

W prezentacji wykorzystano przykłady i fragmenty:

- Grębosz J.: **Symfonia C++**, Programowanie w języku C++ orientowane obiektowo, Wydawnictwo Edition 2000.
- Jakubczyk K.: *Turbo Pascal i Borland C++ Przykłady*, Helion.

Warto zajrzeć także do:

- Sokół R.: **Microsoft Visual Studio 2012 Programowanie w Ci C++**, Helion.
- Kernighan B.W., Ritchie D. M.: **język ANSI C**, Wydawnictwo Naukowo Techniczne.

Dla bardziej zaawansowanych:

- Grębosz J.: **Pasja C++**, Wydawnictwo Edition 2000.
- Meyers S.: **język C++ bardziej efektywnie**, Wydawnictwo Naukowo Techniczne