



Katedra  
Informatyki i Automatyki  
Politechniki Rzeszowskiej

# Informatyka

C++

Opracował: Maciej Penar

## Spis treści

Komentarze dot. programowania .....	3
(De)Motywator .....	3
Debugging .....	3
Proces budowania .....	4
Istotna uwaga dot. Budowania .....	5
Środowisko dla C/C++ .....	6
Wybór środowiska .....	6
Hello World .....	6
Instalacja binariów, czyli Path .....	7
Wstęp do C++ .....	7
Trochę trudniejsze C++ .....	9
C++: teoria .....	11
Jedno trudne, ale za to wartościowe zadanie .....	13
Programowanie w C++ .....	16
Algorytmika .....	17
Systemy operacyjne: teoria .....	18
Systemy operacyjne: dygresja .....	18
Systemy operacyjne: Fork() .....	18
Systemy operacyjne: #thread .....	21
Schematy blokowe .....	21

## Komentarze dot. programowania

### (DE)MOTYWATOR

Programowanie nie jest trudne.

Programowanie polega na pisaniu tzw. źródeł.

Nie zrażamy się jak coś nie działa.

Programy które są tu do napisania są krótkie. Jeśli rozwiązania nie da się wyrazić w mniej niż 30 liniach (licząc z klamrami, sprawdzeniem warunków brzegowych oraz deklaracjami) to prawie na pewno **przekombinowaliście**. Przy zadaniach napisałem na ile linii kodu *mniej-więcej* są to zadania.

Jeśli Wasze rozwiązanie działa, a linii kodu jest więcej – „That’s OK too”.

### DEBUGGING

Na początku pisanie programów ~~może być~~ będzie problematyczne.

1. Najpierw nie znamy składni języka, co spowoduje że program nie będzie chciał się nam skompilować. To są tzw. *błędy syntaktyczne*. Środowisko (np. Visual Studio) będzie próbował nam pomóc z tego typu błędami.
2. Potem udaje nam się uruchomić program, ale nie działa on tak jak powinien. To są tzw. *błędy logiczne*. Na poprawę złej logiki nie ma sposobu, ale w celu diagnozy mam do dyspozycji tzw. Debugger (pułapkę).

Służą one do zawieszenia działania programu w konkretnym miejscu. Zwyczajowo, w środowiskach graficznych, klikamy na linię kodu gdzie chcemy żeby wywołanie programu się wstrzymało:



Takie miejsca nazywamy „breakpoint”.

Ponadto często wymagana jest konfiguracja programu która zezwala na debugging tj. konfiguracja **Debug**. Gdy uruchomimy program z włączonym Debuggerem i wywołanie programu natrafi na breakpoint, wtedy środowisko:

1. Umożliwi podejrzenie wartości zmiennych
2. **Umożliwi wykonywanie programu krok-po-kroku**

W VS odpowiadają za to ikony:



Górnołotnie opisując te opcje:

- Pierwsza to opcja wejścia do funkcji
- **Druga to przejście do kolejnej linii – to jest istotne**
- Trzecia to wyjście z funkcji

## PROCES BUDOWANIA

### Z tego Was będę gnębił.

Gdy już mamy utworzony projekt oraz nasz program, należy go zbudować. Czyli przekształcić źródła programu w pewien tzw. artefakt (coś co możemy jakoś użyć). Najczęstszymi artefaktami są pliki o rozszerzeniach:

- .exe – pliki wykonywalne, czyli takie które możemy uruchomić
- .dll – pliki bibliotek, czyli takie które możemy użyć ponownie (reużyć) w ramach innego projektu
- .jar – misz-masz .exe/.dll dla języka java
- .war – plik aplikacji webowej w języku java, czyli cała strona webowa w formie pliku

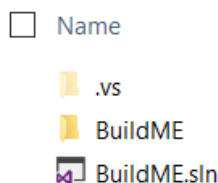
Dla nas istotne są .exe oraz .dll. Rodzaj artefaktu możemy skonfigurować w VS:

1. Klikając PPM-> Właściwości na projekcie
2. Wchodzimy w zakładkę *General*
3. Oglądamy opcję: *Target Extension*

Najczęściej programy możemy zbudować w dwóch konfiguracjach:

- Release – ostateczna wersja programu, często z zastosowaniem agresywnej optymalizacji kodu, może trwać dłużej i często zabrania podpinania Debuggera
- Debug - zbudowanie testowej wersji programu

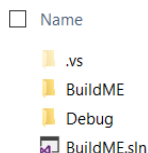
Po utworzeniu projektu jego folder najczęściej wygląda tak:



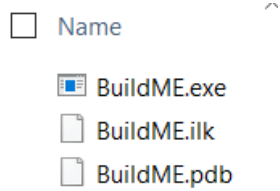
Jeśli kod naszego programu jest poprawny pod względem składni możemy przystąpić do budowania. Mamy trzy opcje:

- Build – czyli możemy zbudować projekt
- Clean – czyli możemy wyczyścić projekt
- Rebuild – czyli możemy wykonać Clean, a potem Build

Gdy wybierzemy Build, w naszym katalogu pojawi się dodatkowy katalog (tu nazwa konfiguracji, czyli Debug):



Jego zawartość to plik .exe – reszta nie jest dla nas w tym momencie istotna:



**Gdy wybierzemy Clean, zawartość katalogu Debug zniknie.**

#### ISTOTNA UWAGA DOT. BUDOWANIA

Niektóre środowiska, np. CodeBlocks **nie** nadpisują artefaktów podczas procesu budowania. To oznacza że modyfikacja programu wypisującego „HelloWorld!” na wypisujący „Hai!” nie będzie widoczna dopóki nie wybierzemy opcji **Clean**.

Czasem też środowisko daje opcję **Clean and Build** zamiast **Rebuild**

## Środowisko dla C/C++

### WYBÓR ŚRODOWISKA

Opcje to:

1. Visual Studio Community Edition 2017: <https://www.visualstudio.com/pl/downloads/>
2. Netbeans: <https://netbeans.org/features/cpp/>
3. Eclipse: <https://www.eclipse.org/downloads/eclipse-packages/>
4. Code blocks: <http://www.codeblocks.org/>
5. Cokolwiek innego, np. Visual Code (<https://code.visualstudio.com/>) + wybrany kompilator

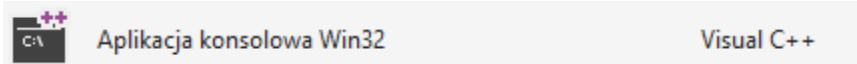
\* W mało prawdopodobnym scenariuszu gdy będziecie chcieli C# to jedyną opcją jest **Visual Studio Community Edition**.

\*Jeśli korzystacie z CodeBlocks – to kompilator (patrz dalej) i debugger (patrz dalej) trzeba zainstalować osobno i w opcjach w CodeBlocks trzeba ustawić ścieżki

### HELLO WORLD

Skompilować i uruchomić Hello World. W Visual Studio wygląda to następująco:

1. Z menu wybieramy: Plik->Nowy->Projekt
2. Szukamy szablonów C++, najlepiej:



3. Tworzymy projekt
4. (W przypadku szablonu Aplikacja konsolowa WIN32): wybieramy domyślne opcje
5. Wprowadzamy kod w pliku [nazwa projektu].cpp:

```
#include "stdafx.h"
#include <iostream>

int main()
{
    std::cout << "Hello World" << std::endl;
    return 0;
}
```

6. Umieszczamy tzw. pułapkę/debugger na linii "return 0"



```

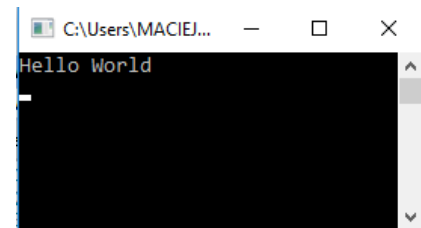
4  #include "stdafx.h"
5  #include <iostream>
6
7
8  int main()
9  {
10     std::cout << "Hello World" << std::endl;
11     return 0;
12 }
13

```

7. Klikamy „Lokalny debugger Windows”:

▶ Lokalny debugger Windows ▾

8. Powinniśmy zobaczyć:



## INSTALACJA BINARIÓW, CZYLI PATH

Częstokroć programy dostarczane są jako pliki wykonywalne np. exe. W środowisku firmy lub w środowisku tzw. produkcyjnym (czyli takim którym reprezentujemy aplikację użytkownikowi końcowemu) często chcemy mieć możliwość wywoływania programów z dowolnego miejsca, najkrótszą możliwą komendą.

Przyczyny takiego chęćstwa są wielorakie:

1. Czasem „Bo tak trzeba” – np. programy wykorzystujące język Java, środowiska budujące / kompilatory
2. Jest nam wygodniej

Najkrótsza możliwa komenda to najczęściej nazwa programu. Żeby móc wywołać program z poziomu linii komend (albo z polecenia ‘uruchom’ na systemie Windows) – **co często jest równoznaczne z instalacją programu dystrybuowanego binarnie** - trzeba wykonać eksport ścieżki (PATH).

Na Windowsie:

1. Kupujemy, pobieramy, ~~pracimy~~, piszemy program który chcemy uruchamiać z linii komend np. *HelloWorld*
2. Wchodzimy w Panel sterowania\Wszystkie elementy Panelu sterowania\System
3. LPM -> Zaawansowane ustawienia systemu
4. Zakładka Zaawansowane -> Zmienne środowiskowe
5. Edytować zmienną **Path**. Dodać ścieżkę do folderu z .exe np. *C:\yaddayadda\Visual Studio 2017\projekt\debug*
6. Uruchomić nowy cmd/powershell’a/*polecenie uruchom*
7. Uruchomić [program].exe... u mnie: HelloWorld.exe

Powinno działać z dowolnego miejsca (katalogu) w systemie.

## Wstęp do C++

Zaimplementować rozwiązania następujących problemów w C++. Samodzielnie przyjąć założenia (obowiązujące do końca kursu i prawdopodobnie do końca studiów). Oto problemy, zazwyczaj każdy wprowadza nowy element języka:

1. Wczytaj liczbę. Na wyjściu wypisz czy jest parzysta (4 linie) Elementy języka: użyć zmiennych cin/cout, operatorów dzielenia modulo, porównania
2. Wczytaj znak. Na wyjściu wypisz jej wartość kodu ASCII (4 linie) Elementy języka: rzutowanie

3. Wczytaj dwie liczby: Wyświetla sumę, różnicę, dzielenie, mnożenie co tam chcesz. Elementy języka: operatory arytmetyczne
4. Wczytaj znak. Jeśli jest małą literą to nic nie rób, w przeciwnym wypadku wyświetl dużą literę. Elementy języka: instrukcja warunkowa, arytmetyka na znakach
5. Wczytaj liczbę. Jaki jest jej adres? Zmodyfikuj ją za pomocą adresacji pośredniej. Elementy języka: operatory & i \*.
6. Wczytaj liczbę do zmiennej dopóki użytkownik nie wprowadzi -1. (15 linii) Na wyjściu wypisz:
  - a. Sumę
  - b. Ile elementów użytkownik wprowadził
  - c. Średnią
 Elementy języka: pętla
7. Wczytać tablicę liczb / liczby. Znaleźć oraz wypisać na wyjściu: (7 linii)
  - a. Minimalną wartość z tablicy
  - b. Maksymalność wartość z tablicy
 Elementy języka: Twój spryt i satysfakcja
8. Napisać funkcję która zwraca wartość  $ax^2 + bx + c$  (parametry to a,b,c,x). Elementy języka: funkcje
9. Napisać funkcję która zwraca wartości  $ax^2 + bx + c$  dla wybranego zakresu całkowitoliczbowego  $x_1, \dots, x_n$ . Elementy języka: 'o! funkcje dają się wywoływać jedna w drugiej' – **skorzystać z funkcji z poprzedniego zadania.**
10. Napisać funkcję która zwraca wartości  $x^2 + 0x + 0$  dla zakresu  $<-10, 10>$  (1 linijka kodu). Elementy języka: **to nazywamy przeciążaniem funkcji. Skorzystać z funkcji z poprzedniego zadania.**
11. Wczytać ciąg znaków, znak po znaku, oraz wypisać na wyjściu czy dany ciąg jest palindromem. (10 linii). Element języka: Twój spryt i satysfakcja
12. Wczytać ciąg znaków, jako string, oraz wypisać na wyjściu czy dany ciąg jest palindromem. (10 linii). Element języka: klasa *string* (`#include <string>`)
13. Wczytać ciąg znaków oraz wypisać cały wyraz wspak (9 linii).
14. Wczytać ciągi znaków *s1*, *s2*. Określić najdłuższy wspólny prefix tych wyrazów i wypisać na wyjściu (17 linii, dwa podpunkty za jednym zamachem):
  - a. Pozycję do której prefixy się zgadzają
  - b. Prefix
15. Wczytywać liczby z zakresu 0 do 999. Gdy użytkownik wprowadzi -1, wypisać wszystkie wprowadzone liczby w kolejności rosnącej (17 linii). (Założyć że wprowadzę ok: 1073741824 liczb. Tj. **nie** implementować bubble sort, tylko count sort: [link](#)). Element języka: satysfakcja.
16. Wczytać tekst. Zaszyfrować poprzez szyfr Cezara, wypisać zaszyfrowaną postać a potem przeprowadzić deszyfrację (10 linii na problem)... **lubię ten problem:**
  - a. Z kluczem 1 (czyli do każdego znaku dodajemy wartość 1, np. a -> b, b->c, z -> a, itd.).
  - b. Z dowolnym kluczem > 0 (czyli do każdego znaku przesuwamy literę o k pozycji np. k = 5, a->f, b->e)
17. Napisać program który wyliczy wartość wielomianu:  $\sum_{i=0}^n x^i a_i$  (18 linijek)

Przykładowe wyjście:

Podaj wartość x:  
**2**  
 Podaj liczbę czynników:  
**3**  
 Podaj czynnik i = 0  
**5** //w pamięci liczone  $5x^0$   
 Podaj czynnik i = 1  
**2** //w pamięci liczone  $5x^0 + 2x^1$   
 Podaj czynnik i = 2  
**1** //w pamięci liczone  $5x^0 + 2x^1 + 1x^2$



Wartość wielomianu to: 13

18. Wczytać liczbę, wypisać ją binarnie. Element języka: klasa *bitset* (`#include <bitset>`).
19. Wczytać liczbę, pomnożyć przez 2 nie używając operatorów `+`, `-`, `/`, `*`. Obejrzeć wynik na bitsecie.  
Element języka: operator binarny `<<`.
20. Wczytać dwie liczby, sprawdzić na ilu pozycjach bitowych się zgadzają. Element języka: operator binarny `&`.
21. Na wejściu otrzymujesz dwie liczby  $L, R$ , znaleźć jaka jest maksymalna wartość  $A \text{ XOR } B$ . Przy ograniczeniu  $L \leq A \leq B \leq R$ . Ograniczenia:  $L, R \leq 10^3$ . Źródło wiadome.

Wejście	Wyjście
10	7
15	

Czemu:

10 XOR 10 = 0  
10 XOR 11 = 1  
10 XOR 12 = 6  
10 XOR 13 = 7  
....  
14 XOR 15 = 1  
15 XOR 15 = 0

Haczyk w zadaniu: jaka jest złożoność Brute-Force'a i jak się ma to do  $L, R \leq 10^3$ .

22. \* Na wejściu otrzymujesz liczbę  $n > 0$ . Znaleźć ile jest liczb  $x$  spełniających warunek  $n \geq x \geq 0$ , takich że  $x + n = x \text{ XOR } n$ . Ograniczenia:  $n \leq 10^{15}$ . Źródło wiadome.

Wejście	Wyjście
10	4

Czemu:

To są te liczby:  
10 XOR 0 = 10 + 0 = 10  
10 XOR 1 = 10 + 1 = 11  
10 XOR 4 = 10 + 4 = 14  
10 XOR 5 = 10 + 5 = 15

## Trochę trudniejsze C++

W tej sekcji ściągamy rękawiczki dla dzieci. Trzeba być samodzielnym.

1. Zadeklarować kilka zmiennych dowolnego typu używając słowa kluczowego **auto**.
2. Wczytać ciągi znaków dopóki użytkownik nie wpisze STOP. Jako miejsce przechowywania użyć klasy `vector` (`#include <vector>`), a nie tablicy. **Posortować** i wypisać jeden długi ciąg znaków z użyciem **pętli zakresowej**.
3. Napisz funkcję wykonującą potęgowanie liczby 2 (wykładnik to parametr):
  - Przez pętle
  - Przez przesunięcie bitowe
  - Dla chętnych: dodać implementację przez pseudo-przesłonięcie funkcji `pow()` z `<math.h>`. Wykonać test która z trzech metod jest szybsza.

Podpowiedź do 3c

Dołączyć <time.h>.  
Poniższy kod wylicza różnicę czasu:

```
time_t start,end;  
double dif;  
  
time (&start);  
// Do some calculation.  
time (&end);  
dif = difftime (end,start);
```

Link: <https://stackoverflow.com/questions/3557221/how-do-i-measure-time-in-c>

4. Napisz funkcję która oblicza silnie: n!

- Napisz wersję iteracyjną (wykorzystującą pętlę)
- Napisz wersję rekurencyjną (funkcja wywołująca samą siebie). Spróbuj wykorzystać „operator trójargumentowy”
- Wyrób opinię na temat obu wersji (długość kodu / czytelność kodu / wydajność)
- Dla jakiego x następują anomalie na typach int, long, long long

Podpowiedź do 4/4b

Operator trójargumentowy ma następującą składnię:

[warunek] ? [wartość gdy prawda] : [wartość gdy fałsz]

Np. zapis  $x > 0 ? x : -x$  oblicza wartość bezwzględną liczby x

Wzór na silnie:

$$factorial(x) = x! = x * (x - 1) * (x - 2) \dots * 2 * 1 = \prod_{i=1}^x i$$

5. Napisać strukturę *student* która przechowuje następujące dane: identyfikator, imię, nazwisko, wiek, rok studiów, semestr. Dobrać typy danych oraz uzasadnić ich wybór.

6. Kontynuacja: Okazuje się że student może być identyfikowany albo za pomocą:

- Liczby nieujemnej całkowitej
- 10 znaków

Zaproponować unię ([link](#)) reprezentującą identyfikator. Zainicjalizować unię znakami, sprawdzić:

- a) Adres poszczególnych pól
- b) Szerokość unii
- c) Wartość znaków
- d) Wartość liczbowa
- e)

7. Dany jest plik **iris.csv** (dość popularny, jest tu np. [link](#)). Pobrać. Przekierować plik jako dane wejściowe do programu. Zaproponować **strukturę** do przechowywania danych. Napisać program który wyliczy średnią wartość pól „sepal\_length”, „sepal\_width”, „petal\_width”, „petal\_length” w każdej z grup „species”. Wypisać te wartości. Zastanowić się czy trzeba trzymać wszystkie obiekty w pamięci. Wyrzucić nagłówki pliku .csv (można ręcznie).

Podpowiedź do 7

W Visual Studio by przekierować plik jako wejście wczytywane z cin należy:

- a) Skopiować plik do folderu w którym jest projekt
- b) Wejść we właściwości projektu (PPM->Właściwości)
- c) Zakładka Debugowanie
- d) W polu „Argumenty polecenia wpisać < **iris.csv**

Polecenie	\$(TargetPath)
Argumenty polecenia	< iris.csv
Katalog roboczy	\$(ProjectDir)
Dołącz	Nie
Typ debugera	Auto
Środowisko	
Scala Środowisko	Tak
Debugowanie SQL	Nie
Akcelerator domyślny Amp	Akcelerator programowy WARP

8. Kontynuacja 7: jeśli poprzednie zadanie wykonano z wyliczaniem „w locie” – przerobić program tak by buforował rekordy.

## Ultymatywne C++

- Zapytaj użytkownika o imię, nazwisko i wiek. Wyświetlić „Witaj [imię] [nazwisko], masz [wiek] lat”. Zwróć uwagę na walidację wejścia: wiek musi mieć sens, imię/nazwisko nie mogą być puste.
- Zmodyfikuj poprzednie zadanie tak żeby użytkownik był reprezentowany jako **klasa**.
  - Zdefiniuj konstruktor dla użytkownika wymagający imię, nazwisko, wiek.
  - Zdefiniuj konstruktor dla użytkownika wymagający imię i nazwisko. Wiek wtedy to 0.
  - Zdefiniuj metodę „howAreYou”, która wyświetla: „Witam. Jestem [imię] [nazwisko] i mam[wiek] lat”. Jeśli wiek nie ma sensu to pomija „i mam [wiek] lat”.
  - \*Zdefiniuj operator >> dzięki któremu można będzie wczytać studenta przez cin >> ;
  - Zdefiniuj metodę gettingOlder() która postarza użytkownika
  - Zdefiniuj metodę changeName(newName) która zmienia nazwisko użytkownika
- Zdefiniuj klasę Student która jest podtypem użytkownika
  - Zmodyfikuj działanie klasy student, operacji howAreYou -> student odpowiada „Lubie pić piwo i jeść kebaby bo mam [wiek] lat i zero zobowiązań”.
- Zdefiniuj klasę reprezentującą bazę ludzi (obiektów studentów i użytkowników)
  - Napisz metodę zatrudniaj(), która pyta „Czy chcesz dodać użytkownika” jeśli wpisano TAK to pyta o typ, po czym dodaje obiekt do bazy, jeśli wpisano NIE, to przerywa działanie. Nie wiadomo ile będzie takich użytkowników, skorzystać z słowa new.
  - Napisz metodę howAreYouEverybody() – która dla wszystkich osób wywołuje metodę howAreYou
  - Zagwarantuj sprzątanie tej klasy

## C++: teoria

Jak już trochę pokodziliście, to warto zadać sobie kilka pytań odnośnie samego języka.

Podpowiedź do kolejnych zadań (1,2)
W Visual Studio można zmusić pre-processor by zapisał wynik do pliku, by to wykonać należy:
1. Wejść w ustawienia projektu (alt + enter)
2. Właściwości konfiguracji -> C/C++ -> Preprocessor
3. Zaznaczyć <b>Przetwarzaj Wstępnie do Pliku TAK</b>

Odpowiedz na pytania:

- Co robi dyrektywa #include
- Co robi dyrektywa #define:

```
#define MAX(a,b) (a>b ? a:b)
void f(){
```

```
if (MAX(-1, 1) > 0) {...}
}
```

3. Jaki będzie efekt działania kodu:

```
int i = 0;
printf("%d", i);
printf("%d", i++);
printf("%d", ++i);
int s = sizeof(i++);
printf("%d", i);
```

4. Przedstaw program któremu kończy się pamięć:
- Na stercie (Heap)
  - Na stosie (Stack)
5. Na czym polega przepełnienie liczb całkowitych (Integer Overflow). Napisać program który ilustruje problem. Wykorzystać liczby całkowite ze znakiem oraz bez znaku.
6. Wybrać typ zmiennej oraz zadeklarować następujące dane:
- Wiek
  - Stałą** PI
  - Wskaźnik na **stałą** PI
  - Stały** wskaźnik na **stałą** PI
7. Jaka jest różnica pomiędzy strukturą a klasą
8. Jaka jest różnica pomiędzy klasą a obiektem
9. Dana jest następująca klasa:

```
class some_class {};
```

Jakie operatory/metody/moduły zostaną utworzone automatycznie przez kompilator?

10. Jaka jest różnica pomiędzy modyfikatorem dostępu private/protected/public. Jaka jest rola zaprzyjaźniania? Jakie konstrukcje języka mogą być poddane zaprzyjaźnieniu?
11. Dlaczego ten kod działa:

```
class test {
public:
    std::string _holder;
    test(char value[]) : _holder(value) {};
};

int main()
{
    test t = "abba"; // czemu to działa?
    std::cout << t._holder << std::endl;
}
```

12. Wyjaśnić na czym polega dziedziczenie / polimorfizm.
13. Przeanalizować klasę **complex** reprezentującą liczbę zespoloną:

[https://pl.wikipedia.org/wiki/Liczby\\_zespolone](https://pl.wikipedia.org/wiki/Liczby_zespolone)

Struktura **complex** wraz z operacjami modułu, dodawaniem i mnożeniem:

```
struct complex { double real, imaginary; };

double modulus(complex & c) {
    return sqrt(pow(c.real, 2) + pow(c.imaginary, 2));
}
```

```

complex operator+(complex & x, complex & y) {
    return complex{ x.real + y.real, x.imaginary + y.imaginary };
}

complex operator*(complex & x, complex & y) {
    return complex{
        x.real * y.real - x.imaginary * y.imaginary ,
        x.imaginary * y.real + x.real * y.imaginary
    };
}

```

14. Jaka jest różnica (jeśli istnieje) pomiędzy przeciążaniem operatorów wewnątrz klasy, a poza nią? Jeśli istnieje różnica, napisać program ilustrujący ją (teraz to już wiadomo że jest). Patrz przykład:

Przeciążanie poza klasą:
<pre> class some_class { .. };  some_class operator+(some_class&amp; t_first, some_class&amp; t_second) { ... }; </pre>
Przeciążanie wewnątrz klasy:
<pre> class some_class { public:     some_class operator+(some_class&amp; t_second) {...}; }; </pre>

15. Co to jest singleton. Przedstawić jego implementację.

## Jedno trudne, ale za to wartościowe zadanie

Poniżej przedstawiono kod złej implementacji klasy String (kod znajduje się na repo). Przeanalizować poniższy program oraz wyjście z konsoli. Zidentyfikować błędy oraz poprawić/wskazać jak poprawić. (Język C++, S. Prata, roz. 12, str. 554).

```

#include <iostream>
#include <cstring>
#include <windows.h> // Dla znaków na konsoli Windowsa

typedef char character;
class StringBad {
private:
    character * str;
    int len;
    static int num_strings;
public:
    StringBad(const character * s) {
        len = std::strlen(s);
        str = new character[len + 1];
        std::strcpy(str, s);
        num_strings++;

        std::cout << num_strings << ": \"" << str << "\" - obiekt utworzony.\n";
    }
    StringBad() {
        len = 4;
        str = new character[4];
        std::strcpy(str, "C++");
    }
};

```

```

        num_strings++;

        std::cout << num_strings << ": \"" << str << "\" - obiekt domyślny utwor
zony.\n";
    }
    ~StringBad() {
        std::cout << "\"" << str << "\" - obiekt usunięty,";
        --num_strings;
        std::cout << "są jeszcze " << num_strings << ".\n";
        delete[] str;
    };
    friend std::ostream & operator<<(std::ostream & os, const StringBad & st
) {
        os << st.str;
        return os;
    };
};

int StringBad::num_strings = 0;

void callVal(StringBad val) {
    std::cout << "Obiekt przekazany przez wartość:" << std::endl;
    std::cout << "\"" << val << "\"" << std::endl;
}

void callRef(StringBad & val) {
    std::cout << "Obiekt przekazany przez referencje:" << std::endl;
    std::cout << "\"" << val << "\"" << std::endl;
}

int main()
{
    /*
    Wymagane dla polskich znaków na Windows
    */
    SetConsoleOutputCP(65001);
    setlocale(LC_ALL, "65001");

    using std::cout; using std::endl;
    {
        cout << "Zaczynamy blok" << endl;
        StringBad msg1("Witaj na kursie Informatyka");
        StringBad msg2("Programujemy w C");
        StringBad msg3("Kiedyś będziemy uczyć się sieci");

        cout << "Pierwsza wiadomość: " << msg1 << endl;
        cout << "Druga wiadomość: " << msg2 << endl;
        cout << "Trzecia wiadomość: " << msg3 << endl << endl;

        callRef(msg1);

        cout << "Pierwsza wiadomość raz jeszcze: " << msg1 << endl << endl;

        callVal(msg2);

        cout << "Druga wiadomość raz jeszcze: " << msg2 << endl << endl;

        cout << "Spróbujmy coś innego, inicjalizacja innym obiektem: " << endl;
    }
}

```

```

        StringBad evenWorse = msg3;

        cout << "Trzecia wiadomość z innej zmiennej: " << evenWorse << endl << endl;

        cout << "Ciśniemy dalej: przypisanie do innego obiektem: " << endl;
        StringBad worst;
        cout << "I teraz przypisanie!" << endl;
        worst = msg1;

        cout << "Pierwsza wiadomość z innej zmiennej: " << worst << endl << endl;

        cout << "I wychodzimy z bloku" << endl;
    }
    cout << "Koniec main";
    return 0;
}

```

Wynik z CodeBlocks / Visual Studio:

Zaczynamy blok

- 1: "Witaj na kursie Informatyka" - obiekt utworzony.
- 2: "Programujemy w C" - obiekt utworzony.
- 3: "Kiedyś będziemy uczyć się sieci" - obiekt utworzony.

Pierwsza wiadomość: Witaj na kursie Informatyka

Druga wiadomość: Programujemy w C

Trzecia wiadomość: Kiedyś będziemy uczyć się sieci

Obiekt przekazany przez referencje:

"Witaj na kursie Informatyka"

Pierwsza wiadomość raz jeszcze: Witaj na kursie Informatyka

Obiekt przekazany przez wartość:

"Programujemy w C"

"Programujemy w C" - obiekt usunięty, są jeszcze 2.

Druga wiadomość raz jeszcze:

8

Spróbujmy coś innego, inicjalizacja innym obiektem:

Trzecia wiadomość z innej zmiennej: Kiedyś będziemy uczyć się sieci

Ciśniemy dalej: przypisanie do innego obiektem:

- 3: "C++" - obiekt domyślny utworzony.

I teraz przypisanie!

Pierwsza wiadomość z innej zmiennej: Witaj na kursie Informatyka

I wychodzimy z bloku

"Witaj na kursie Informatyka" - obiekt usunięty, są jeszcze 2.

"Kiedyś będziemy uczyć się sieci" - obiekt usunięty, są jeszcze 1.

"r

8

"będziemy uczyć się sieci" - obiekt usunięty, są jeszcze 0.

**<- Tu Visual wyrzuca wyjątek**

"C++" - obiekt usunięty,są jeszcze -1.

---

" - obiekt usunięty,są jeszcze -2.

Koniec main

## Programowanie w C++

Zaprojektować i zaimplementować grę w kółko i krzyżyk na planszy 3x3.

1. Każdy ruch powinien być drukować aktualny stan planszy na konsoli.
2. Program powinien wspierać grę:
  - a. Przeciwko drugiemu graczowi
  - b. Przeciwko AI (niech losuje ruch)
3. Przemyśleć projekt, przeanalizować, zoptymalizować.



## Algorytmika

Opisać rozwiązanie lub zaimplementować w jakimkolwiek języku. W C++ do wykonania zadania wystarczy następujące nagłówki:

```
#include <cmath>
#include <cstdio>
#include <vector>
#include <iostream>
#include <algorithm>
```

Źródło wiadome – jak ktoś był uważny wie skąd.

Michał przygotowuje się do zawodów programistycznych na które składają się  $N$  testów próbnych. Michał wierzy w zababony. Wierzy że może „zachować szczęście”. Każdy test próbny opisuje dwoma liczbami całkowitymi  $L_i$  oraz  $T_i$

- $L_i$  jest to liczba szczęścia powiązana z testem próbnym. Jeśli Michał **podejdzie** do testu próbnego, to jego szczęście opada o wartość  $L_i$ . Jeśli **nie podejdzie** do testu próbnego to odkłada szczęście, czyli zwiększa je o wartość  $L_i$ .
- $T_i$  oznacza ważność testu próbnego. Jeśli jest równa 1 to test jest **ważny**, jeśli jest równy 0 to test **nie jest ważny**

**Dane wejściowe:** Pierwsza linia w pliku to wartość dwie wartości całkowite oddzielone spacją:

$N$  – liczba testów próbnych

$K$  – liczba ważnych konkursów które Michał może przegrać

Potem wystąpi  $N$  linii, gdzie  $i$  – ta linia składa się z dwóch wartości całkowitych oddzielonych spacją:  $L_i$  oraz  $T_i$

### Ograniczenia

- $1 \leq N \leq 100$
- $0 \leq K \leq N$
- $1 \leq L_i \leq 10^4$
- $0 \leq T_i \leq 1$

### Wyjście

Wypisać maksymalną możliwą do uzyskania przez Michała liczbę szczęścia.

### Przykład:

Wejście	Wyjście
6 3 5 1 2 1 1 1 8 1 10 0 5 0	29

Liczba testów próbnych to  $N = 6$ , z czego 4 są ważne. Michał może nie wziąć udziału w trzech testach próbnych  $K = 3$ .

Michał bierze udział tylko w trzecim teście próbnym o liczbie szczęścia  $L_3 = 1$ . Oznacz to że szczęście Michała to:  $5 + 2 - 1 + 8 + 10 + 5 = 29$

## Systemy operacyjne: teoria

1. Wyjaśnij relację pomiędzy: **procesem a wątkiem**
2. Wyjaśnij różnicę pomiędzy: **współbieżnością kooperacyjną a współbieżnością konkurencyjną**
3. Wyjaśnij różnicę (o ile jest) pomiędzy **współbieżnością a równoległością**
4. Co to jest fork-bomba?
5. Co to jest atak typu DoS? Co to jest atak typu DDoS?
6. Do czego służą programy w systemach unixopodobnych:
  - a. ps
  - b. kill
  - c. pstree
  - d. top
7. Co to jest mutex? (patrz pkt 4).

## Systemy operacyjne: dygresja

Dalej trzeba nie ruszymy bez unixowego środowiska:

- Zainstaluj / uzyskaj dostęp do unix'a lub systemu unixo' podobnego wraz z możliwością kompilacji C/C++
- Wejdź na stronę: [https://www.onlinegdb.com/online\\_c++\\_compiler](https://www.onlinegdb.com/online_c++_compiler) (tu semafory nie zadziałają)
- Wejdź na stronę: <https://ideone.com> (a tu zadziałają)

## Systemy operacyjne: Fork()

Ta część wymaga unixów.

1. Co robi poniższy kod?

```
#include <iostream>
#include <unistd.h>

int main() {
    pid_t pid;
    pid = fork();
    if (pid == -1) {
        std::cout << "Exit with failure" << std::endl;
        exit(EXIT_FAILURE);
    }

    if (pid == 0) {
        std::cout << "Child!" << std::endl;
    }
    else {
        std::cout << "Parent!" << std::endl;
    }
    sleep(3);
}
```

```
        return 0;
    }
```

Co robi poniższy kod? Ile komunikatów łącznie się pojawi? Ile procesów w sumie weźmie udział w przetwarzaniu?

```
#include <iostream>
#include <unistd.h>

void shout(int id, bool child) {
    std::cout << "I am process no. " << id << (child ? " and I'm a child" :
    "") << std::endl;
}

void aux(int n) {
    if (n > 0) {
        pid_t pid;
        pid = fork();
        if (pid == -1) {
            std::cout << "Exit with failure" << std::endl;
            exit(EXIT_FAILURE);
        }

        shout(pid, pid == 0);
        aux(n - 1);
    }
}

int main() {
    aux(2);
    sleep(1000);
    return 0;
}
```

2. Na bazie programów 1) oraz 2) napisz program który uruchomi trzy procesy (zdekomponować każdy podpunkt na osobną funkcję):
- W ramach pierwszego nastąpi wypisanie pierwszych 5 wielokrotności liczby 2.
  - W ramach drugiego nastąpi wypisanie 7 razy „...ups”
  - W ramach trzeciego nastąpi wypisanie: „Jestem X kebabem”. Za X podstawić w kolejnych iteracjach (począwszy od iteracji zerowej): „mały”, „średnim”, „dużym”.

3. Przeanalizuj poniższy kod (dzięki uprzejmości dr Sławomira Samoleja: [http://ssamolej.kia.prz.edu.pl/dydaktyka/inf\\_1EE\\_ZI/Procesy\\_wspolbieznosc1.zip](http://ssamolej.kia.prz.edu.pl/dydaktyka/inf_1EE_ZI/Procesy_wspolbieznosc1.zip)):

```
#include <iostream>
#include <unistd.h>
#include <sys/file.h>
#include <semaphore.h>

int main()
{
    int a = 0;
    pid_t pid;
    sem_t * sem = sem_open("my_semaphore", O_CREAT);

    sem_init(sem, 1, 0);
    pid = fork();

    if (pid == -1)
    {
        exit(EXIT_FAILURE);
    }

    if (pid == 0)
    {
        while (1)
        {
            sem_wait(sem);
            std::cout << "Synchronizing" << std::endl;
        }
    }
    else
    {
        for (int i = 0; i < 5; i++)
        {
            std::cout
                << "Notifying the semaphore "
                << i
                << std::endl;
            sem_post(sem);
            sleep(1);
        }
    }
    exit(EXIT_SUCCESS);
}
```

## Systemy operacyjne: #thread

1. Przeanalizuj ten fragment (powinien zadziałać na każdym systemie operacyjnym).

```
#include <iostream>
#include <mutex>
#include <thread>
#include <chrono>

std::mutex mtx;

void f(int iterations) {
    for (int i = 0; i < iterations; ++i) {
        mtx.lock();
        std::cout << i << std::endl;
        mtx.unlock();
        std::this_thread::sleep_for(std::chrono::seconds(1));
    }
}

int main(){
    const int THREADS = 7;
    std::thread temp[THREADS];
    for (int i = 0; i < THREADS; ++i) {
        temp[i] = std::thread(f, 10); //Przekazanie funkcji i parametru
    }
    std::this_thread::sleep_for(std::chrono::seconds(60));
    return 0;
}
```

2. Powtórz zadanie z Kebabami. Ale zamiast procesów będzie chodzić o wątki.
3. Zapoznać się z problemem stołujących filozofów. Znaleźć twist fabularny w zadaniu. Kilka podpowiedzi:
  - a. Jak umiecie – to poprawcie kod tak żeby wszyscy się najedli.
    - i. W main() – 1 linijka
    - ii. Pozostały kod: ok. 12-14 linijek copy-paste
  - b. W kodzie nie powinno być zbędnych słów kluczowych

Link: <https://github.com/mpenarprz/InformatykaA1/blob/master/kod/philosophers.cpp>

## Schematy blokowe

Narysuj schemat blokowe następujących problemów:

1. Wczytaj liczbę do zmiennej  $k$ . Na wyjściu wypisz liczbę przeciwną.
2. Wczytaj liczbę do zmiennej  $k$ . Na wyjściu wypisz czy jest parzysta
3. Wczytaj liczby do zmiennej  $k$  dopóki użytkownik nie wprowadzi -1. Na wyjściu wypisz:
  - a. Sumę
  - b. Ile elementów użytkownik wprowadził
  - c. Średnią
4. Wczytaj liczbę do zmiennej  $k$ . Na wyjściu wypisz z ilu cyfr się składa.
5. Wczytaj liczby do zmiennych  $k, m$ . Na wyjściu wypisz resztę z dzielenia  $k$  przez  $m$ .

Komentarz: założmy że nie mamy operacji modulo % (link: <https://pl.wikipedia.org/wiki/Modulo>)

6. Wczytać tablicę liczb do zmiennej  $t$ . Znaleźć oraz wypisać na wyjściu:
  - a. Minimalną wartość
  - b. Maksymalność wartość
7. Załóżmy że użytkownik wczytuje ciąg znaków do zmiennej  $s$  o długości  $n$  oraz mamy zdefiniowaną operację  $s[i]$  zwracającą  $i$ -ty znak.  
Np. dla  $s = \text{"Informatyka"}$  operacja  $s[0] \rightarrow \text{"I"}$ ,  $s[1] \rightarrow \text{"n"}$ ,  $s[2] \rightarrow \text{"f"}$ , itd.  
Wczytać ciąg znaków oraz wypisać na wyjściu czy dany ciąg jest palindromem.
8. Dla modelu z zadania 7. Wczytać ciąg znaków  $s$  oraz wypisać cały wyraz wspak.
9. Dla modelu z zadania 7. Wczytać ciągi znaków  $s1$ ,  $s2$ . Określić najdłuższy wspólny prefix tych wyrazów i wypisać na wyjściu (liczbę, nie prefix).