Politechnika Białostocka

Wydział Elektryczny

Katedra Elektrotechniki, Energoelektroniki i Elektroenergetyki

Instrukcja do pracowni specjalistycznej

Temat ćwiczenia:

JĘZYK C - FUNKCJE

Ćwiczenie nr INF_D10

Pracownia specjalistyczna z przedmiotu:

Informatyka

Kod: **EDS1B1007**

Opracował:

dr inż. Jarosław Forenc

Białystok 2022

Materiały zostały opracowane w ramach projektu „PB2020 – Zintegrowany Program Rozwoju Politechniki Białostockiej” realizowanego w ramach Działania 3.5 Programu Operacyjnego Wiedza, Edukacja, Rozwój 2014-2020 współfinansowanego ze środków Europejskiego Funduszu Społecznego.

Spis treści

Materiały dydaktyczne przeznaczone dla studentów Wydziału Elektrycznego PB.

1. Opis stanowiska

1.1. Stosowana aparatura

Podczas zajęć wykorzystywany jest komputer klasy PC z systemem operacyjnym Microsoft Windows 10.

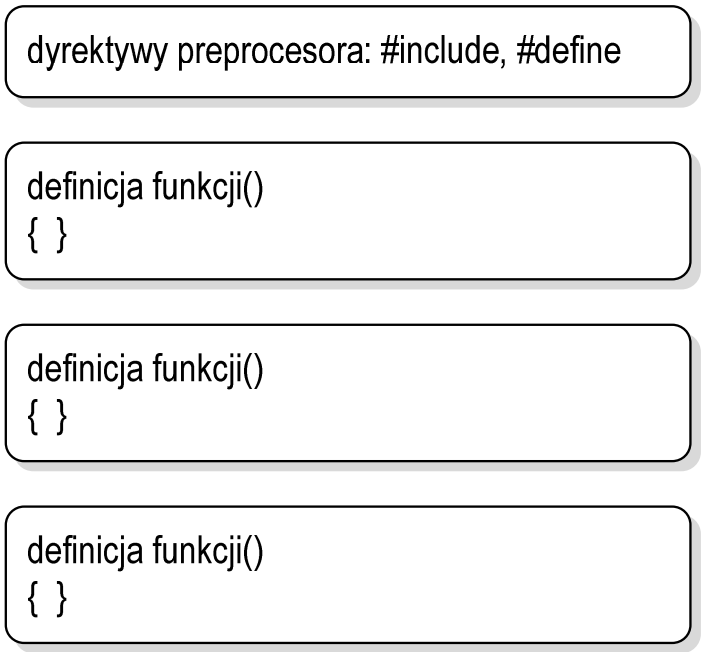
1.2. Oprogramowanie

Na komputerach zainstalowane jest środowisko programistyczne Microsoft Visual Studio 2008 Standard Edition lub nowsze zawierające kompilator Microsoft Visual C++.

2. Wiadomości teoretyczne

2.1. Struktura programu w języku C

W strukturze programu w języku C można wyróżnić dyrektywy preprocesora (np. **#include**, **#define**) oraz definicje funkcji (Rys. 1).

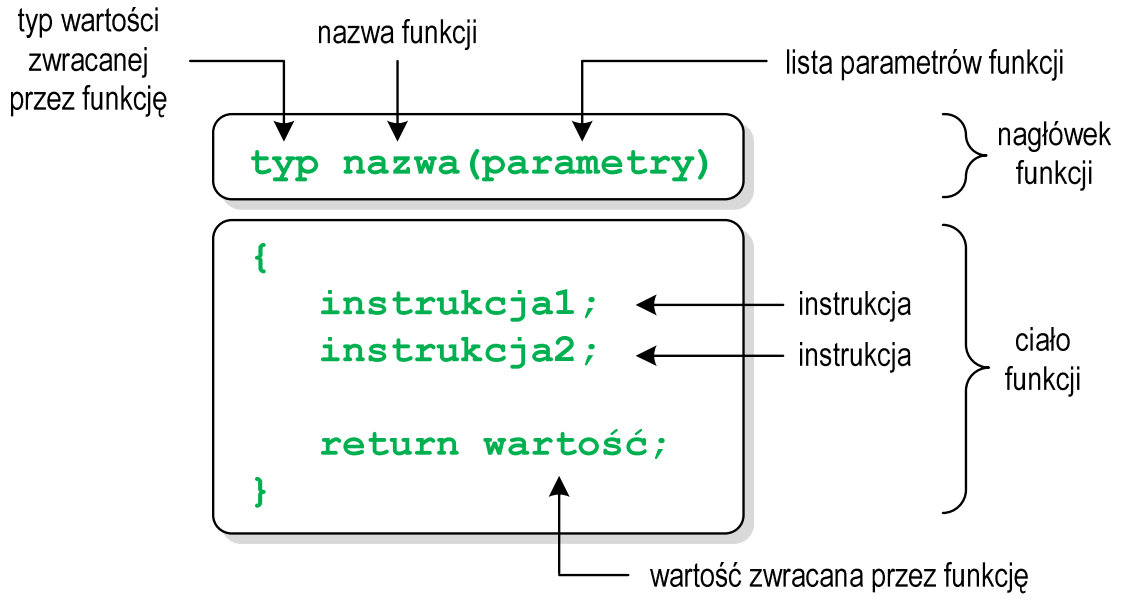


Rys. 1. Struktura programu w języku C

Każdy program musi zawierać przynajmniej jedną definicję funkcji. Funkcja ta powinna nazywać się **main()**. Od niej rozpoczyna się wykonanie całego programu. Oprócz funkcji **main()** w programie mogą występować także inne funkcje zdefiniowane przez użytkownika.

2.2. Ogólna struktura funkcji w języku C

Ogólną strukturę funkcji w języku C przedstawia Rys. 2. Funkcja składa się **nagłówka** oraz **ciała** funkcji, które razem tworzą **definicję funkcji**. Nagłówek zawiera typ wartości zwracanej przez funkcję, nazwę funkcji oraz listę parametrów przekazywanych do niej podczas wywołania (uruchomienia). Ciało funkcji ograniczone jest nawiasami klamrowymi: **{** i **}**. Pomiędzy nawiasami umieszczone są instrukcje. Funkcja kończy swoje działanie po wykonaniu instrukcji zawierającej słowo kluczowe **return**. Po **return** umieszcza się wartość, która zostanie zwrócona do miejsca wywołania funkcji. Typ tej wartości powinien być zgodny z nazwą typu umieszczoną w nagłówku funkcji. Słowo **return** może pojawić się wielokrotnie w ciele funkcji.



Rys. 2. Ogólna struktura funkcji w języku C

Poniżej zamieszczono przykład programu zawierającego dwie funkcje:

- **main()** - główna funkcja programu;
- **suma()** - funkcja obliczająca i zwracająca sumę dwóch liczb rzeczywistych.

Program zawierający funkcję obliczającą sumę dwóch liczb rzeczywistych.

```
#include <stdio.h>

float suma(float a, float b)
{
    float y;
    y = a + b;
    return y;
}

int main(void)
{
    float x1 = 10, x2 = 20, wynik;

    wynik = suma(x1,x2);
    printf("Wynik = %f\n", wynik);

    return 0;
}
```

Wynik uruchomienia programu:

```
Wynik = 30.000000
```

Wykonanie programu rozpoczyna się od funkcji **main()**. Gdy dochodzimy do instrukcji zawierającej funkcję **suma()**, to wywołanie tej funkcji powoduje przekazanie sterowania do jej pierwszej instrukcji. Do funkcji **suma()** przekazywane są dwa argumenty: **x1** i **x2** typu **float**. Pierwszy parametr (**a**) otrzymuje wartość pierwszego argumentu wywołania funkcji (**x1**), natomiast drugi parametr (**b**) - wartość drugiego argumentu wywołania funkcji (**x2**). Parametry funkcji traktowane są tak samo jak zmienne zadeklarowane w tej funkcji i zainicjalizowane wartościami argumentów wywołania. Jeśli funkcja ma kilka parametrów, to dla każdego z nich podaje się typ i nazwę parametru. Parametry oddzielane są od siebie przecinkami.

W ciele funkcji można umieszczać dowolne konstrukcje języka C: deklaracje zmiennych typów prostych i złożonych, instrukcje warunkowe, pętle, itp. W powyższym programie funkcja **suma()** zawiera instrukcję, w której dodawane są do siebie zmienne **a** i **b**, zaś wynik zapisywany jest w zmiennej **y**.

Powrót z funkcji (do miejsca zaraz po jej wywołaniu) następuje na skutek wykonania instrukcji **return**. Wartość zwracana przez funkcję (czyli wartość zmiennej **y** występującej bezpośrednio po **return**) podstawiana jest pod zmienną **wynik**. Po słowie **return** może występować dowolne wyrażenie. Wyrażenie to często umieszczane jest w nawiasach, ale nie jest to konieczne.

Funkcję **suma()** można zapisać w prostszy sposób, pomijając zmienną **y**.

```
float suma(float a, float b)
{
    return (a+b);
}
```

W definicji funkcji można nadać jej parametrom domyślne wartości.

```
float suma(float a = 15, float b = 25)
{
    return (a+b);
}
```

W takim przypadku funkcję można wywołać z dwoma, jednym lub bez żadnych argumentów. Brakujące argumenty zostaną zastąpione wartościami domyślnymi.

```
wynik = suma(x1,x2);    // 10 + 20
wynik = suma(x1);       // 10 + 25
wynik = suma();         // 15 + 25
```

Nie wszystkie parametry muszą mieć podane domyślne wartości. Wartości muszą być podawane od prawej strony listy parametrów.

```
float suma(float a, float b = 25)
{
    return (a+b);
}
```

Powyższa funkcja może być wywołana z dwoma lub jednym argumentem.

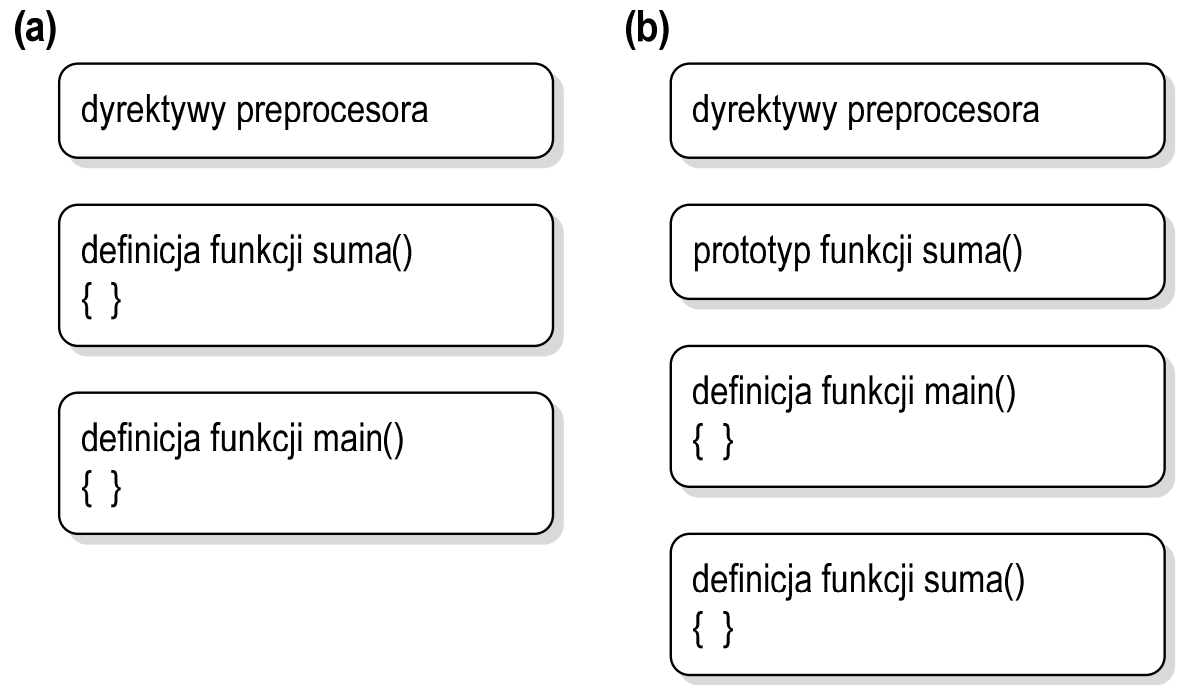
```
wynik = suma(x1,x2);    // 10 + 20
wynik = suma(x1);       // 10 + 25
```

W wywołaniu funkcji jako argumenty mogą występować stałe liczbowe, nazwy zmiennych, wyrażenia arytmetyczne lub wywołania innych funkcji. Wywołanie funkcji może być argumentem innej funkcji.

```
wynik = suma(10,20);
wynik = suma(x1,x2);
wynik = suma(x1*20+4,x1/x2);
wynik = suma(sin(x1),x1+x2);
printf("Wynik = %f\n",suma(x1,x2));
```

2.3. Umieszczanie definicji funkcji w programie

W programie przedstawionym w poprzednim rozdziale definicja funkcji **suma()** była umieszczona przed definicją funkcji **main()** (Rys. 3a).



Rys. 3. Kolejność funkcji w programie: (a) - funkcja **suma()** przed funkcją **main()**, (b) funkcja **suma()** za funkcją **main()**

Definicję funkcji **suma()** można umieścić także po definicji funkcji **main()**. Ponieważ zasięg widzialności funkcji rozpoczyna się od miejsca jej deklaracji, należy przed definicją funkcji **main()** podać formalną deklarację czyli **prototyp** funkcji **suma()** (Rys. 3b). Prototyp opisuje to samo co nagłówek funkcji, ale kończy się średnikiem. Dzięki prototypom kompilator sprawdza w wywołaniu funkcji: nazwę funkcji, liczbę i typ argumentów, typ zwracanej wartości.

Program zawierający prototyp funkcji suma().

```
#include <stdio.h>

float suma(float a, float b);

int main(void)
{
    float x1 = 10, x2 = 20, wynik;

    wynik = suma(x1,x2);
    printf("Wynik = %f\n", wynik);

    return 0;
}

float suma(float a, float b)
{
    float y;
    y = a + b;
    return y;
}
```

W prototypie nie musimy podawać nazw parametrów - wystarczą tylko typy:

```
float suma(float, float);
```

Podanie nazw parametrów wpływa na czytelność kodu programu.

W przypadku umieszczenia prototypu funkcji i pominięcia jej definicji błąd wystąpi nie na etapie kompilacji programu, ale na etapie łączenia (linkowania).

2.4. Klasyfikacja funkcji

Funkcja w języku C może zwracać wartość lub jej nie zwracać. Do funkcji mogą być przekazywane argumenty lub też może ich nie być (funkcja bezargumentowa). Z powyższych względów wyróżnia się cztery typy funkcji.

Funkcja nie zwracająca wartości i nie posiadająca argumentów

- w nagłówku funkcji, jako typ zwracanej wartości, podaje się słowo **void**,
- w nagłówku funkcji, w miejscu listy jej parametrów, podaje się słowo **void** (tak zaleca standard języka C) lub nie wpisuje się nic,
- jeśli w ciele funkcji występuje **return**, to nie może znajdować się za nim żadna wartość,
- jeśli w ciele funkcji nie występuje **return**, to sterowanie wraca do punktu wywołania na skutek zakończenia wykonywania wszystkich instrukcji znajdujących się w funkcji,
- definicja funkcji może mieć jedną z poniższych postaci:

```
void nazwa(void)
{
    instrukcje;
    return;
}
```

```
void nazwa()
{
    instrukcje;
    return;
}
```

```
void nazwa(void)
{
    instrukcje;
}
```

```
void nazwa()
{
    instrukcje;
}
```

- w wywołaniu funkcji podaje się jej nazwę i nawiasy **( )**:

```
nazwa();
```

Przykład programu zawierającego funkcję, która nie zwraca wartości i nie ma argumentów wywołania:

```
#include <stdio.h>

void drukuj_linie(void)
{
    printf("-------------------------\n");
}

int main(void)
{
    drukuj_linie();
    printf("  Funkcje nie sa trudne!\n");
    drukuj_linie();

    return 0;
}
```

Wynik uruchomienia programu:

```
-------------------------
  Funkcje nie sa trudne!
-------------------------
```

Funkcja nie zwracająca wartości i posiadająca argumenty

- w nagłówku funkcji, jako typ zwracanej wartości, podaje się słowo **void**,
- jeśli w ciele funkcji występuje **return**, to nie może znajdować się za nim żadna wartość,
- jeśli w ciele funkcji nie występuje **return**, to sterowanie wraca do punktu wywołania na skutek zakończenia wykonywania wszystkich instrukcji znajdujących się w funkcji,
- definicja funkcji może mieć jedną z poniższych postaci:

```
void nazwa(parametry)
{
    instrukcje;
    return;
}
```

```
void nazwa(parametry)
{
    instrukcje;
}
```

- wywołanie funkcji:

```
nazwa(argumenty);
```

Przykład programu zawierającego funkcję, która nie zwraca wartości i ma argumenty wywołania:

```
#include <stdio.h>

void drukuj_dane(char *imie, char *nazwisko, int wiek)
{
    printf("Imie:           %s\n",imie);
    printf("Nazwisko:       %s\n",nazwisko);
    printf("Wiek:           %d\n",wiek);
    printf("Rok urodzenia:  %d\n\n",2021-wiek);
}

int main(void)
{
    drukuj_dane("Jan","Kowalski",23);
    drukuj_dane("Barbara","Nowak",28);

    return 0;
}
```

Wynik uruchomienia programu:

```
Imie:           Jan
Nazwisko:       Kowalski
Wiek:           23
Rok urodzenia:  1998

Imie:           Barbara
Nazwisko:       Nowak
Wiek:           28
Rok urodzenia:  1993
```

Funkcja zwracająca wartość i nie posiadająca argumentów

- w nagłówku funkcji, w miejscu listy jej parametrów, podaje się słowo **void** (tak zaleca standard języka C) lub nie wpisuje się nic,

- typ wartości zwracanej przez funkcję musi być zgodny z typem wartości występującej po słowie **return**,
- definicja funkcji może mieć jedną z poniższych postaci:

```
typ nazwa(void)
{
    instrukcje;
    return wartość;
}
```

```
typ nazwa()
{
    instrukcje;
    return wartość;
}
```

- w wywołaniu funkcji zwracana wartość podstawiana jest pod zmienną:

```
typ zmienna;
zmienna = nazwa();
```

- poprawne jest także wywołanie funkcji bez podstawiania zwracanej wartości pod zmienną:

```
nazwa();
```

Przykład programu zawierającego funkcję, która zwraca wartość i nie ma argumentów wywołania:

```
#include <stdio.h>

int liczba_sekund_rok(void)
{
    return (365 * 24 * 60 * 60);
}

int main(void)
{
    int wynik = liczba_sekund_rok();
    printf("W roku jest: %d sekund\n",wynik);

    return 0;
}
```

Wynik uruchomienia programu:

```
W roku jest: 31536000 sekund
```

Funkcja zwracająca wartość i posiadająca argumenty

- najbardziej popularny typ funkcji,
- typ wartości zwracanej przez funkcję musi być zgodny z typem wartości występującej po słowie **return**,
- definicja funkcji powinna mieć postać:

```
typ nazwa(parametry)
{
    instrukcje;
    return wartość;
}
```

- w wywołaniu funkcji zwracana wartość podstawiana jest pod zmienną:

```
typ zmienna;
zmienna = nazwa(argumenty);
```

- poprawne jest także wywołanie funkcji bez podstawiania zwracanej wartości pod zmienną:

```
nazwa(argumenty);
```

Przykład programu zawierającego funkcję, która zwraca wartość i ma argumenty wywołania:

```
#include <stdio.h>

float prad(float nap, float moc)
{
    return (moc/nap);
}
```

```
int main(void)
{
    float U, P, I;

    printf("Podaj U [V]:  ");
    scanf("%f",&U);

    printf("Podaj P [W]:  ");
    scanf("%f",&P);

    I = prad(U,P);
    printf("Prad [A]:     %g\n",I);

    return 0;
}
```

Wynik uruchomienia programu:

```
Podaj U [V]:  230
Podaj P [W]:  1600
Prad [A]:     6.95652
```

3. Przebieg ćwiczenia

Na pracowni specjalistycznej należy wykonać wybrane zadania wskazane przez prowadzącego zajęcia. W różnych grupach mogą być wykonywane różne zadania.

1. Napisz program zawierający funkcję wyświetlającą na ekranie wizytówkę o poniższej postaci (pamiętaj o ramce z gwiazdek). Wywołaj napisaną funkcję.

```
**********************************
*          Jan Kowalski          *
*  e-mail: j.kowalski@gmail.com  *
*        tel. 123-456-789        *
**********************************
```

2. Energię elektryczną **W** pobraną w czasie **t** przez odbiornik o mocy **P** określa wzór:

$$W = P \cdot t \tag{1}$$

Napisz funkcję obliczającą i zwracającą zużycie energii elektrycznej (w **kWh**) pobranej przez odbiornik o mocy **P** w czasie **t**. W funkcji **main()** wczytaj z klawiatury wartości **P** i **t**, wywołaj napisaną funkcję, a następnie wyświetl wartość przez nią zwróconą.

3. Napisz funkcję zamieniającą odległość podaną w **kilometrach** na **mile lądowe** i funkcję zamieniającą odległość podaną w **kilometrach** na **mile morskie**. W funkcji **main()** wczytaj z klawiatury odległość w kilometrach, wywołaj napisane funkcje i wyświetl wartości przez nie zwrócone.

   Uwaga: 1 mila lądowa = 1609,344 metrów, 1 mila morska = 1851,852 metrów.

4. Napisz funkcję obliczającą i zwracającą rezystancję **R** jednorodnego przewodnika o przekroju poprzecznym **S** i długości **l** wykonanego z materiału o rezystywności ρ. Stosując funkcję oblicz rezystancję **R** przewodnika o długości **l = 100 m** i przekroju **S = 2,5 mm²** w przypadku, gdy jest on wykonany z miedzi, aluminium, srebra lub złota (wywołaj 4-krotnie tę samą funkcję, ale z różnymi argumentami).

   Tabela 1. Rezystywność wybranych materiałów w temperaturze 20 ºC

Materiał	**Rezystywność [Ω·m]**
miedź	$1{,}72 \cdot 10^{-8}$
aluminium	$2{,}82 \cdot 10^{-8}$
srebro	$1{,}59 \cdot 10^{-8}$
złoto	$2{,}44 \cdot 10^{-8}$

5. Napisz program zawierający funkcję obliczającą i zwracającą częstotliwość rezonansową **fr** układu o rezystancji **R**, indukcyjności **L** i pojemności **C** wprowadzonych z klawiatury w funkcji **main()**.

Przykładowe uruchomienie programu	Wzór

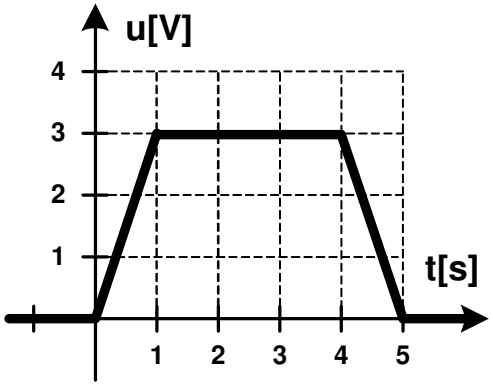
`Indukcyjnosc L [H]:  0.04`
`Pojemnosc C [F]:  2.0e-6`
`---`
`Czestotliwosc fr [Hz]:  562.697693`
	$f_r = \frac{1}{2\pi\sqrt{LC}}$ (2)

Rezystancja R [Om]: 5000 Indukcyjnosc L [H]: 0.02 Pojemnosc C [F]: 4.0e-5 ---------- Czestotliwosc fr [Hz]: 177.942413	$f_r = \frac{1}{2\pi\sqrt{LC - \left(\frac{L}{R}\right)^2}}$ (3)
Rezystancja R [Om]: 500 Indukcyjnosc L [H]: 0.03 Pojemnosc C [F]: 6.0e-5 ---------- Czestotliwosc fr [Hz]: 118.508408	$f_r = \frac{1}{2\pi}\sqrt{\frac{1}{LC} - \frac{1}{(RC)^2}}$ (4)
Rezystancja R [Om]: 10 Indukcyjnosc L [H]: 1 Pojemnosc C [F]: 1.0e-6 ---------- Czestotliwosc fr [Hz]: 159.146988	$f_r = \frac{1}{2\pi}\sqrt{\frac{1}{LC} - \left(\frac{R}{L}\right)^2}$ (5)
Rezystancja R [Om]: 100 Indukcyjnosc L [H]: 0.05 Pojemnosc C [F]: 5.0e-3 ---------- Czestotliwosc fr [Hz]: 10.060807	$f_r = \frac{1}{2\pi\sqrt{LC}}\sqrt{1 - \frac{L}{R^2C}}$ (6)
Rezystancja R [Om]: 10 Indukcyjnosc L [H]: 0.1 Pojemnosc C [F]: 1.0e-6 ---------- Czestotliwosc fr [Hz]: 503.54397	$f_r = \frac{1}{2\pi\sqrt{LC - (RC)^2}}$ (7)

6. Suma poniższego szeregu liczbowego wynosi **1/4**. Napisz program zawierający funkcję obliczającą i zwracającą sumę **n**-wyrazów tego szeregu. Następnie wykorzystując powyższą funkcję oblicz i wyświetl różnice pomiędzy sumą dokładną (**1/4**), a sumą **n = 10**, **n = 100** i **n = 1000** wyrazów tego szeregu.

$$\frac{1}{1\cdot 2\cdot 3} + \frac{1}{2\cdot 3\cdot 4} + \cdots + \frac{1}{n(n+1)(n+2)} \qquad (8)$$

7. Rys. 4 przedstawia przebieg impulsu trapezowego. Napisz funkcję, która na podstawie przekazanego do niej czasu **t** oblicza i zwraca odpowiadającą mu wartość napięcia **u**. Następnie wykorzystując powyższą funkcję oblicz i wyświetl wartości napięcia dla czasu **t** zmieniającego się od **0** do **6** sekund z krokiem **0,25** sekundy (zastosuj pętlę **for**). Wyświetl wyniki w dwóch kolumnach (**czas**, **napięcie**).



Rys. 4. Przebieg impulsu trapezowego

4. Literatura

[1] Prata S.: Język C. Szkoła programowania. Wydanie VI. Helion, Gliwice, 2016.

[2] Kernighan B.W., Ritchie D.M.: Język ANSI C. Programowanie. Wydanie II. Helion, Gliwice, 2010.

[3] Deitel P.J., Deitel H.: Język C. Solidna wiedza w praktyce. Wydanie VIII. Helion, Gliwice, 2020.

[4] Kochan S.G.: Język C. Kompendium wiedzy. Wydanie IV. Helion, Gliwice, 2015.

[5] King K.N.: Język C. Nowoczesne programowanie. Wydanie II. Helion, Gliwice, 2011.

[6] http://www.cplusplus.com/reference/clibrary - C library - C++ Reference

[7] https://cpp0x.pl/dokumentacja/standard-C/1 - Standard C

5. Pytania kontrolne

1. Opisz ogólną strukturę definicji funkcji w języku C.
2. Omów sposób wykonania programu składającego się z więcej niż jednej definicji funkcji.
3. Wyjaśnij czym różni się deklaracja od definicji funkcji?

4. Omów klasyfikację funkcji ze względu na liczbę parametrów i zwracaną wartość.

6. Wymagania BHP

Warunkiem przystąpienia do praktycznej realizacji ćwiczenia jest zapoznanie się z instrukcją BHP i instrukcją przeciw pożarową oraz przestrzeganie zasad w nich zawartych.

W trakcie zajęć laboratoryjnych należy przestrzegać następujących zasad.

- Sprawdzić, czy urządzenia dostępne na stanowisku laboratoryjnym są w stanie kompletnym, nie wskazującym na fizyczne uszkodzenie.
- Jeżeli istnieje taka możliwość, należy dostosować warunki stanowiska do własnych potrzeb, ze względu na ergonomię. Monitor komputera ustawić w sposób zapewniający stałą i wygodną obserwację dla wszystkich członków zespołu.
- Sprawdzić prawidłowość połączeń urządzeń.
- Załączenie komputera może nastąpić po wyrażeniu zgody przez prowadzącego.
- W trakcie pracy z komputerem zabronione jest spożywanie posiłków i picie napojów.
- W przypadku zakończenia pracy należy zakończyć sesję przez wydanie polecenia wylogowania. Zamknięcie systemu operacyjnego może się odbywać tylko na wyraźne polecenie prowadzącego.
- Zabronione jest dokonywanie jakichkolwiek przełączeń oraz wymiana elementów składowych stanowiska.
- Zabroniona jest zmiana konfiguracji komputera, w tym systemu operacyjnego i programów użytkowych, która nie wynika z programu zajęć i nie jest wykonywana w porozumieniu z prowadzącym zajęcia.
- W przypadku zaniku napięcia zasilającego należy niezwłocznie wyłączyć wszystkie urządzenia.

- Stwierdzone wszelkie braki w wyposażeniu stanowiska oraz nieprawidłowości w funkcjonowaniu sprzętu należy przekazywać prowadzącemu zajęcia.
- Zabrania się samodzielnego włączania, manipulowania i korzystania z urządzeń nie należących do danego ćwiczenia.
- W przypadku wystąpienia porażenia prądem elektrycznym należy niezwłocznie wyłączyć zasilanie stanowiska. Przed odłączeniem napięcia nie dotykać porażonego.