Politechnika Wrocławska

Bardzo szybkie podsumowanie: wykład 1

wer. 10 z drobnymi modyfikacjami!

Wojciech Myszka

2023-03-05 21:05:09 +0100

HR EXCELLENCE IN RESEARCH

Uwagi

1. Obowiązuje cały materiał!
2. Tu tylko podsumowanie.

Politechnika Wrocławska

Informatyka I — ARM031007
Wprowadzenie do informatyki — MCM032102

wer. 15 z drobnymi modyfikacjami!

Wojciech Myszka

Katedra Mechaniki, Inżynierii Materiałowej i Biomedycznej

2022-03-03 20:21:16 +0100

HR EXCELLENCE IN RESEARCH

Politechnika Wrocławska

Program zajęć I

1. Wprowadzenie. Algorytm. Schematy blokowe. Idea programowania strukturalnego.
2. Struktura programów w C. Identyfikator, typy danych (typy fundamentalne: całkowite, rzeczywiste, znakowe, logiczny), deklaracja i inicjalizacja zmiennych, definiowanie stałych. Komunikacja poprzez konsolę. Operatory: arytmetyczne, logiczne, inkrementacji, dekrementacji, przypisania. Obliczanie wartości wyrażeń.
3. Struktury sterowania obliczeniami: rozgałęzienia i skoki, pętle pojedyncze i zagnieżdżone. Instrukcje proste i złożone; instrukcje warunkowe, wyrażenia warunkowe, instrukcje iteracyjne.
4. Preprocesor: dyrektywy, makrodefinicje.
5. Funkcje: budowa funkcji, argumenty funkcji, wynik wykonania funkcji, definicje i deklaracje globalne, argumenty funkcji main, rekurencja.

Politechnika Wrocławska

Program zajęć II

6. Tablice (tablice jedno i wielowymiarowe), łańcuchy znaków.
7. Wskaźniki. Pamięć dynamiczna.
8. **Kolokwium.**
9. Struktury danych, unie: deklaracja struktury, definiowanie zmiennej strukturalnej, tablice struktur, wskaźniki a struktury danych.
10. Operacje na plikach: otwieranie, zamykanie plików, czytanie i zapisywanie do plików.
11. Formatowanie w operacjach wejście/wyjście. Binarne wejście/wyjście.
12. Operacje na łańcuchach znaków.
13. Programowanie strukturalne w praktyce: podział programu na moduły, struktury danych, kompilacja.
14. Programy pomocnicze: diff, make, systemy rcs i cvs, debugger. Zarządzanie wersjami. Środowiska zintegrowane.
15. **Kolokwium zaliczeniowe.**

Politechnika Wrocławska

Po co uczyć się programowania? I

Pozwolę sobie przytoczyć tu 12 przewidywań na temat przyszłości programowania [**?**].

1. Procesory graficzne (GPU) będą naszymi następnymi procesorami.
2. Bardzo wiele przyszłego programowania dotyczyć będzie baz danych (*big data*).
3. JavaSript do wszystkiego.
4. Android na każdym urządzeniu.
5. Internet rzeczy — kolejne nowe platformy.
6. Open source na wiele sposóbów.
7. Oparte na platformie WordPress aplikacje webowe.
8. Programy zostaną zastąpione przez „wtyczki” (plug-ins).
9. Niech żyją polecenia wydawane w terminalu!

Politechnika Wrocławska

Po co uczyć się programowania? II

10. Nie liczmy na dalsze upraszczanie językow programowania.
11. Programowaniem zajmować będą się programiści z krajów o najniższym koszcie pracy.
12. W dalszym ciagu szefostwo nie będzie rozumieć o co chodzi z tym programowaniem.

Politechnika Wrocławska

Slajdy

- Robotyka i Automatyzacja Procesów:
  https://myszka.kmim.wm.pwr.edu.pl/dydaktyka/c/rap/
- Mechatronika:
  https://myszka.kmim.wm.pwr.edu.pl/dydaktyka/c/mtr/

Bryk

Na podstawie slajdów powstał Bryk. Celowo nie nazywam go skryptem czy podręcznikiem. Można go znaleźć pod adresem: https://myszka.kmim.wm.pwr.edu.pl/dydaktyka/c/mtr/#bryk. Odsyłacz jest również na stronie wykładu.

Literatura I

- *A GNU Manual*, rozdział Nested Functions.
  Free Software Foundation, Inc., 2015.
- Programowanie w języku C, 2007.
  Wersja elektroniczna dostępna pod adresem:
  http://pl.wikibooks.org/wiki/Programowanie:C.
- David Griffiths, Dawn Griffiths.
  Head First C.
  Head First. O'Reilly, 2011.
  Są szanse, żeby ksiazka byłą dostępna pod tym adresem:
  http://proquestcombo.safaribooksonline.com/9781449335649.
  Można się o nią „dobijać" zgodnie z instrukcją na stronie biblioteki.

Politechnika Wrocławska

Literatura II

David Griffiths, Dawn Griffiths.
C. Rusz głową!
Helion, Gliwice, 2013.
Strona księgarni: http://helion.pl/ksiazki/c-rusz-glowa-david-griffiths-dawn-griffiths,cruszg.htm.

J. Grębosz.
Symfonia C++.
Kallimach, Kraków, 2000.

David Harel, Yishai Feldman.
Rzecz o istocie informatyki: algorytmika.
Klasyka informatyki. Wydawnictwa Naukowo-Techniczne, Warszawa, 2001, 2002, 2008.

Politechnika Wrocławska

Literatura III

George Heineman, Gary Pollice, Stanley Selkow.
Algorytmy. Almanach.
Helion, Gliwice, 2010.

Steve Holmes.
C programming.
http://www.imada.sdu.dk/~svalle/courses/dm14-2005/mirror/c/, 1995.

Ted Jensen.
A tutorial on pointers and arrays in C, Feb. 2000.
Dostępne jako
http://pweb.netcom.com/~tjensen/ptr/pointers.htm.

Literatura IV

- Rob Kendrick.
  Some dark corners of C, 2013.
- B. W. Kernighan, D. M. Ritchie.
  Język ANSI C.
  WNT, Warszawa, 2007.
- Ben Klemens.
  21st Century C.
  O'Reilly Media, 2012.
  http://shop.oreilly.com/product/0636920025108.do.

Politechnika Wrocławska

Literatura V


M. J. Kubiak.
Programuję w językach Turbo Pascal i C/C++: programowanie strukturalne z elementami programowania obiektowego.
Mikom, Warszawa, 2001.


Ciaran O'Riordan.
Learning GNU C.
Ciaran O'Riordan, 2002.


Nick Parlante.
Pointer basics, 1999.


Nick Parlante.
Pointers and memory, 2000.

Politechnika Wrocławska

Literatura VI


Nick Parlante.
Binary trees, 2001.


Nick Parlante.
The great tree-list recursion problem, 2001.


Nick Parlante.
Linked list problems, 2002.


Nick Parlante.
Essential C, 2003.


Nick Parlante, Julie Zelenski.
Unix programming tools, 2001.

Literatura VII


Richard M. Reese.
Understanding and Using C Pointers.
O'Reilly, 2013.


Richard M. Reese.
Wskaźniki w języku C: przewodnik.
Helion, Gliwice, 2014.
Dostęp po zalogowaniu w bazie NASBI.
http://biblioteka.pwr.wroc.pl/NASBI_Naukowa_Akademicka_Sieciowa_Biblioteka_Internetowa,161.dhtml.

Politechnika Wrocławska

Literatura VIII

Adam Sapek.
W głąb języka C.
Helion, Gliwice, lipiec 1993.
Wersja elektroniczna dostępna pod adresem:
`ftp://ftp.helion.pl/online/wglab/1-3.pdf`.

Herbert Schildt.
Leksykon C/C++.
Oficyna Wydawnicza LTP, Warszawa, 2002.

C. Sexton.
Język C to proste.
Wyd. RM, Warszawa, 2001.

Literatura IX

C. Sexton.
Programowanie w C++ — to proste.
RM, Warszawa, 2001.

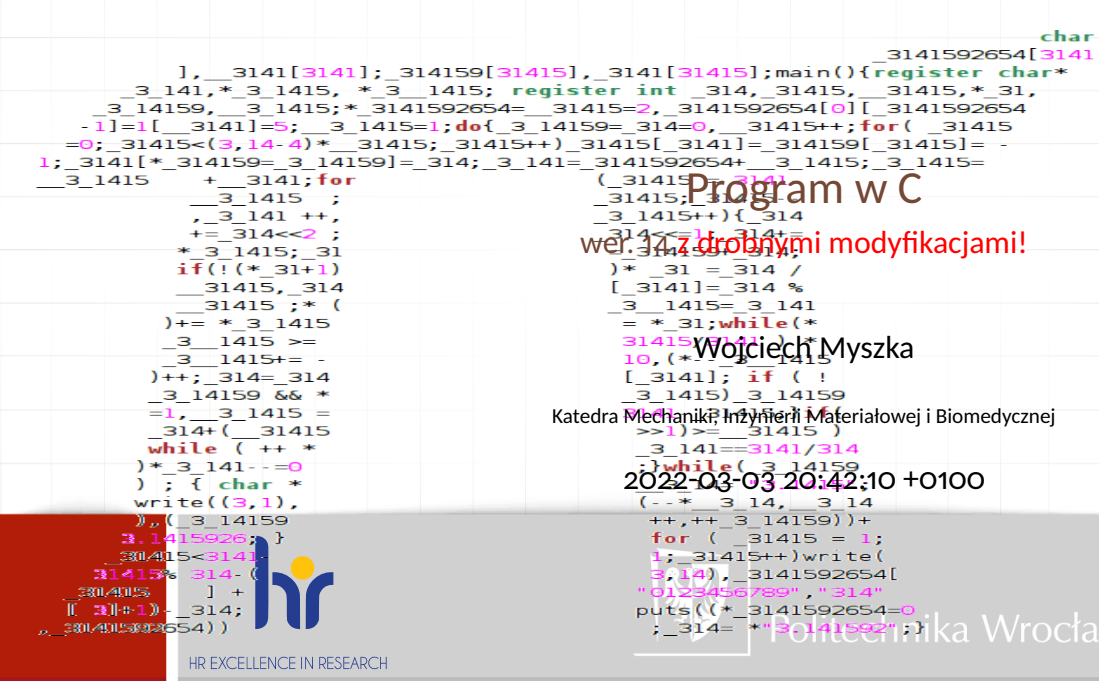
Piotr Stańczyk.
Algorytmika praktyczna: Nie tylko dla mistrzów.
Wydawnictwo Naukowe PWN, Warszawa, 2009.

K. Stec.
Wybrane elementy języka C.
Wyd. Pol. Śląskiej, Gliwice, 2001.

Clovis L Tondo, Scott E Gimpel, Danuta Kruszewska.
Język ANSI C: Ćwiczenia I Rozwiązania.
Wydawnictwa Naukowo-Techniczne, Warszawa, wydanie wyd. 2, 2004.

Literatura X

- Peter Wayner.
  12 predictions for the future of programming, Luty 2014.
- N. Wirth.
  Algorytmy + struktury danych = programy.
  WNT, Warszawa, 2001.
- Piotr Wróblewski.
  Algorytmy: struktury danych i techniki programowania.
  Helion, Gliwice, 2010.



Program w C

wer. 14 z drobnymi modyfikacjami!

Wojciech Myszka

Katedra Mechaniki Inżynierii Materiałowej i Biomedycznej

2022-03-03 20:42:10 +0100

Tak wygląda program w języku C

```c
/* Hello World in C, Ansi-style */
#include <stdio.h>
int main(void)
{
        puts("Hello World!");
}
```

A tak (ten sam) w Pascalu

```
program test;
{rozne deklaracje}
begin
        writeln('to jest program');
end.
```

Program składa się z:

1. Pewnej **struktury**;
2. **Poleceń** wykonywanych przez procesor;
3. Obiektów (zwanych **zmiennymi**) służących do przechowywania danych, wyników i wartości pośrednich uzyskanych podczas obliczeń;
4. **Stałych** używanych podczas obliczeń i do inicjowania wartości zmiennych;

Program

```c
/*
  Hello World in C, Ansi-style
*/
#include <stdio.h>
int main(void)
{
      int z;
      z = z + 1;
      puts("Hello World!"); // druk
}
```

Program

- komentarz

```
/*
  Hello World in C, Ansi-style
*/
#include <stdio.h>
int main(void)
{
      int z;
      z = z + 1;
      puts("Hello World!"); // druk
}
```

Program

- struktura

```c
/*
  Hello World in C, Ansi-style
*/
#include <stdio.h>
int main(void)
{
      int z;
      z = z + 1;
      puts("Hello World!"); // druk
}
```

Program

- struktura
  - deklaracje

```
/*
  Hello World in C, Ansi-style
*/
#include <stdio.h>
int main(void)
{
      int z;
      z = z + 1;
      puts("Hello␣World!"); // druk
}
```

Program

- polecenia

```c
/*
  Hello World in C, Ansi-style
*/
#include <stdio.h>
int main(void)
{
      int z;
      z = z + 1;
      puts("Hello World!"); // druk
}
```

Program

- identyfikatory (zmienne?)

```c
/*
  Hello World in C, Ansi-style
*/
#include <stdio.h>
int main(void)
{
      int z;
      z = z + 1;
      puts("Hello World!"); // druk
}
```

Program

- stałe

```
/*
  Hello World in C, Ansi-style
*/
#include <stdio.h>
int main(void)
{
      int z;
      z = z + 1;
      puts("Hello World!"); // druk
}
```

Program

- zmienne

```c
/*
  Hello World in C, Ansi-style
*/
#include <stdio.h>
int main(void)
{
      int z;
      z = z + 1;
      puts("Hello World!"); // druk
}
```

Program

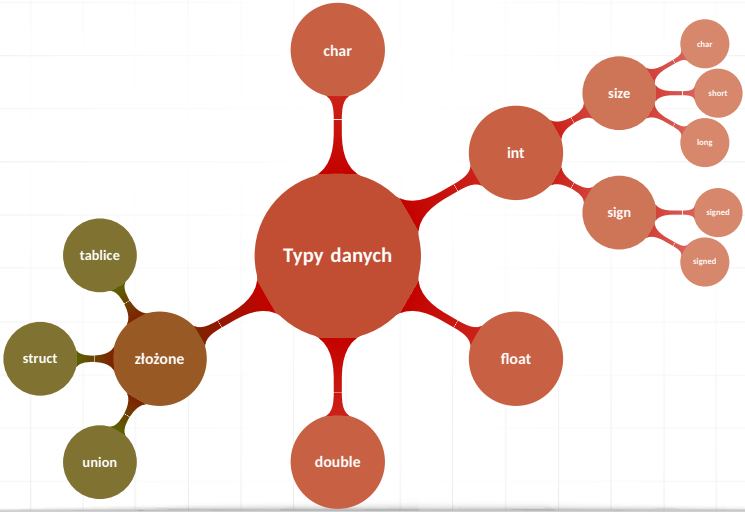
```c
/*
  Hello World in C, Ansi-style
*/
#include <stdio.h>
int main(void)
{
      int z;
      z = z + 1;
      puts("Hello World!"); // druk
}
```

- funkcje

Zmienne

1. Każdy program operuje na pewnych obiektach służących do przechowywania bieżących wartości.
2. Obiekty te nazywa się **zmiennymi**.
3. Każda zmienna musi mieć jakąś nazwę. **Pierwszym znakiem nazwy powinna być litera**, używać można również cyfr i znaków podkreślenia.
4. W odróżnieniu od różnych innych pojemników — te do przechowywania wartości rozróżniają typ przechowywanej wartości.
5. Jak zmienna przechowuje jedną wartość — nazywa się **zmienną prostą**, gdy potrafi przechować więcej wartości — **złożoną**.

Typy danych


Typy danych
char
int
size
char
short
long
sign
signed
signed
float
double
złożone
tablice
struct
union

Reprezentacja binarna: teksty

1. Każdy znak tekstu to jeden bajt.
2. Zmienna tych **char** zajmuje jeden bajt.
3. Reprezentacją binarną znaku ASCII jest liczba całkowita (bez znaku) o wartości równej kodowi ASCII tego znaku.

Reprezentacja binarna: liczby całkowite

1. Każdy liczba całkowita tekstu to jeden, dwa, cztery lub osiem bajtów…
2. Zmienna typu **int** zajmuje cztery bajty (czyli 32 bity).
3. Liczby całkowite (ze znakiem) reprezentowane są w zapisie uzupełnieniowym do dwu.
4. Najbardziej znaczący bit to bit znaku (1 oznacza liczbę ujemną, 0 dodatnią)
5. Liczby całkowite bez znaku reprezentowane są w kodzie binarnym.

Reprezentacja binarna: liczby zmiennoprzecinkowe

1. Każda liczba zmiennoprzecinkowa zajmuje 4, 8 lub 16 bajtów.
2. Dokładnie sposób zapisu liczb definiuje norma IEEE 754.
3. Stosuje się zapis wykładniczy w postaci $znak * mantysa * 2^{cecha}$.
4. Znak to jeden bit, cecha (wykładnik) to 8, 11 albo 15 bitów (binarna liczba całkowita ze znakiem); mantysa (ułamek binarny) — pozostałe bity.
5. Liczby zapisywane są w postaci znormalizowanej (przed kropką dziesiętną musi być 1 — cyfra różna od zera).
6. Specjalne „wartości” $+\infty$, $-\infty$ oraz NaN (Not a Number) mają również swoje reprezentacje binarne; takie wyniki mogą pojawić się w przypadku „dzielena przez zero”.

Nazwy zmiennych

1. Każdy identyfikator musi być unikatowy!
2. Wielkie i małe litery są ważne.
3. Nazwy identyfikatorów powinny zaczynać się od litery i składać z liter i cyfr.
4. Znak _ (podkreślenie) traktowany jest jak litera i może służyć do tworzenia bardziej czytelnych nazw.
5. Znaku _ nie powinno się stosować na początku nazwy — bardzo wiele nazw systemowych zaczyna się od niego.
6. Nie należy używać polskich liter w nazwach.
7. Tradycyjnie zmienne pisze się małymi literami, a różnego rodzaju stałe — wielkimi.
8. Słowa kluczowe (**int**, **void**, **if**, **else**,...) są zarezerwowane i nie mogą być używane jako nazwy.

Słowa kluczowe języka C

auto	break	case	char
const	continue	default	do
double	else	enum	extern
float	for	goto	if
int	long	register	return
short	signed	sizeof	static
struct	switch	typedef	union
unsigned	void	volatile	while

Deklaracje zmiennych I

Język C zna następujące typy zmiennych:

1. **char** znakowe (elementarną jednostką informacji jest jeden znak).
2. **int** całkowite.
3. **float** zmiennoprzecinkowe („rzeczywiste”).
4. **double** zmiennoprzecinkowe, podwójnej precyzji.
5. **void** pusty typ; nie można zadeklarować zmiennej takiego typu, ale może być on wykorzystany do zwrócenia uwagi, że funkcja nie zwraca wartości lub, że nie przyjmuje parametrów.

Przykład deklaracji:
typ nazwa;

Deklaracje zmiennych II

```
int a;
char b;
float c;
double d;
```

Wyprowadzanie wartości zmiennych

printf("Jakiś tekst %*cośtam* tekst", *zmienna*);