Závěr

- Jazyky z rodiny jazyka C,
- mravní naučení o programování.
- Kam dál?

- Jazyk C,
- C++,
- Java,
- PHP, Javascript, ...

Jazyk C

Na počátku byl Pascal. Pak vznikly jazyky A, B a C.

- Vznikl jako modifikace Pascalu,

Jazyk C

Na počátku byl Pascal. Pak vznikly jazyky A, B a C.

- Vznikl jako modifikace Pascalu,
- používá nám již známou syntax,

Jazyk C

Na počátku byl Pascal. Pak vznikly jazyky A, B a C.

- Vznikl jako modifikace Pascalu,
- používá nám již známou syntax,
- není objektový, funkce jsou identifikovány jménem (ne strukturou parametrů),

Jazyk C

Na počátku byl Pascal. Pak vznikly jazyky A, B a C.

- Vznikl jako modifikace Pascalu,
- používá nám již známou syntax,
- není objektový, funkce jsou identifikovány jménem (ne strukturou parametrů),
- neexistuje typ `string`, vše se řeší pomocí pointerů.

Jazyk C

Na počátku byl Pascal. Pak vznikly jazyky A, B a C.

- Vznikl jako modifikace Pascalu,
- používá nám již známou syntax,
- není objektový, funkce jsou identifikovány jménem (ne strukturou parametrů),
- neexistuje typ `string`, vše se řeší pomocí pointerů.
- Místo Pascalské (postfixové) stříšky se používá prefixová hvězdička pro dereferenci.

Jazyk C

Na počátku byl Pascal. Pak vznikly jazyky A, B a C.

- Vznikl jako modifikace Pascalu,
- používá nám již známou syntax,
- není objektový, funkce jsou identifikovány jménem (ne strukturou parametrů),
- neexistuje typ `string`, vše se řeší pomocí pointerů.
- Místo Pascalské (postfixové) stříšky se používá prefixová hvězdička pro dereferenci.
- Paměť se alokuje pomocí funkce `malloc`, které řekneme, kolik bytů potřebujeme, uvolňujeme pomocí funkce `free`, proměnné se snadno přetypovávají (včetně pointeru do longu).

Jazyk C

podruhé

- Pole je jen převlečený pointer, tedy string se udělá jako pointer na char.

Jazyk C

podruhé

- Pole je jen převlečený pointer, tedy string se udělá jako pointer na char.
- Vše je zakuklený integer (nebo longint). Tedy `if(a=b)...`

Jazyk C

podruhé

- Pole je jen převlečený pointer, tedy string se udělá jako pointer na char.
- Vše je zakuklený integer (nebo longint). Tedy `if(a=b)...`
- Do programu se vstupuje funkcí `main`.

Jazyk C

podruhé

- Pole je jen převlečený pointer, tedy string se udělá jako pointer na char.
- Vše je zakuklený integer (nebo longint). Tedy `if(a=b)...`
- Do programu se vstupuje funkcí `main`.
- Jazyk C předává parametry vždy hodnotou. Chceme-li referenci, použijeme pointer.

Příklady v jazyce C

```
#include <stdio.h>
#include <malloc.h>
int main(int argc, char*argv[])
{     int a=1;
      while(a<=argc) printf("%s\n ",argv[a++]);
}
------------------------------
a=malloc(strlen(b)+1);
while(*a++=*b++); //strcpy
```

Vlastnosti jazyka C

- Někteří lidé považují jazyk C za přežitý,

Vlastnosti jazyka C

- Někteří lidé považují jazyk C za přežitý,
- skutečnost je, že v C se dosud programuje všechno důležité.

Vlastnosti jazyka C

- Někteří lidé považují jazyk C za přežitý,
- skutečnost je, že v C se dosud programuje všechno důležité.
- Člověk má plnou kontrolu nad svěřenými prostředky,

Vlastnosti jazyka C

- Někteří lidé považují jazyk C za přežitý,
- skutečnost je, že v C se dosud programuje všechno důležité.
- Člověk má plnou kontrolu nad svěřenými prostředky,
- z toho plyne i zodpovědnost (nikdo nekontroluje přetypování, nikdo nekontroluje správné argumenty, nikdo nekontroluje střílení do paměti).

Vlastnosti jazyka C

- Někteří lidé považují jazyk C za přežitý,
- skutečnost je, že v C se dosud programuje všechno důležité.
- Člověk má plnou kontrolu nad svěřenými prostředky,
- z toho plyne i zodpovědnost (nikdo nekontroluje přetypování, nikdo nekontroluje správné argumenty, nikdo nekontroluje střílení do paměti).
- Jak se řekne pascalské `a^.b`?

Vlastnosti jazyka C

- Někteří lidé považují jazyk C za přežitý,
- skutečnost je, že v C se dosud programuje všechno důležité.
- Člověk má plnou kontrolu nad svěřenými prostředky,
- z toho plyne i zodpovědnost (nikdo nekontroluje přetypování, nikdo nekontroluje správné argumenty, nikdo nekontroluje střílení do paměti).
- Jak se řekne pascalské `a^.b`?
- Buďto `(*a).b`, nebo `a->b`.

Vlastnosti jazyka C

- Někteří lidé považují jazyk C za přežitý,
- skutečnost je, že v C se dosud programuje všechno důležité.
- Člověk má plnou kontrolu nad svěřenými prostředky,
- z toho plyne i zodpovědnost (nikdo nekontroluje přetypování, nikdo nekontroluje správné argumenty, nikdo nekontroluje střílení do paměti).
- Jak se řekne pascalské `a^.b`?
- Buďto `(*a).b`, nebo `a->b`.
- Neexistují třídy a objekty, existují struktury a unie.

Vlastnosti jazyka C

- Někteří lidé považují jazyk C za přežitý,
- skutečnost je, že v C se dosud programuje všechno důležité.
- Člověk má plnou kontrolu nad svěřenými prostředky,
- z toho plyne i zodpovědnost (nikdo nekontroluje přetypování, nikdo nekontroluje správné argumenty, nikdo nekontroluje střílení do paměti).
- Jak se řekne pascalské `a^.b`?
- Buďto `(*a).b`, nebo `a->b`.
- Neexistují třídy a objekty, existují struktury a unie.
- Součástí jména struktury resp. unie může být i klíčové slovo `struct`, resp. `union` (součástí jména typu).

Vlastnosti jazyka C II

- Neexistují generika (ani šablony), existují makra.

Vlastnosti jazyka C II

- Neexistují generika (ani šablony), existují makra.
- Příklad: `#define MAX(a,b) (a>b?a:b)` – makro je před překladem natvdo přepsáno se všemi potenciálně nepříjemnými důsledky:

Vlastnosti jazyka C II

- Neexistují generika (ani šablony), existují makra.
- Příklad: `#define MAX(a,b) (a>b?a:b)` – makro je před překladem natvdo přepsáno se všemi potenciálně nepříjemnými důsledky:
- `MAX(f(x),g(x))` zavolá funkci, která vrátí vyšší výsledek podruhé.

C++

- Rozšíření jazyka C, tedy většina věcí funkčních v C funguje i v C++, jenže...

C++

- Rozšíření jazyka C, tedy většina věcí funkčních v C funguje i v C++, jenže...
- silnější typová kontrola, kontrola parametrů volaných funkcí, objekty.

C++

- Rozšíření jazyka C, tedy většina věcí funkčních v C funguje i v C++, jenže...
- silnější typová kontrola, kontrola parametrů volaných funkcí, objekty.
- Cokoliv existuje, typicky se najde v C++ včetně vícenásobné dědičnosti,

C++

- Rozšíření jazyka C, tedy většina věcí funkčních v C funguje i v C++, jenže...
- silnější typová kontrola, kontrola parametrů volaných funkcí, objekty.
- Cokoliv existuje, typicky se najde v C++ včetně vícenásobné dědičnosti,
- funguje jako C# až na drobné syntaktické odchylky:

C++

- Rozšíření jazyka C, tedy většina věcí funkčních v C funguje i v C++, jenže...
- silnější typová kontrola, kontrola parametrů volaných funkcí, objekty.
- Cokoliv existuje, typicky se najde v C++ včetně vícenásobné dědičnosti,
- funguje jako C# až na drobné syntaktické odchylky:
- `abstract int f();` $\Rightarrow$ `virtual int f()=NULL;`

C++

- Rozšíření jazyka C, tedy většina věcí funkčních v C funguje i v C++, jenže...
- silnější typová kontrola, kontrola parametrů volaných funkcí, objekty.
- Cokoliv existuje, typicky se najde v C++ včetně vícenásobné dědičnosti,
- funguje jako C# až na drobné syntaktické odchylky:
- `abstract int f();` $\Rightarrow$ `virtual int f()=NULL;`
- `override void f()...` $\Rightarrow$ `virtual void f()...`

C++

- Rozšíření jazyka C, tedy většina věcí funkčních v C funguje i v C++, jenže...
- silnější typová kontrola, kontrola parametrů volaných funkcí, objekty.
- Cokoliv existuje, typicky se najde v C++ včetně vícenásobné dědičnosti,
- funguje jako C# až na drobné syntaktické odchylky:
- `abstract int f();` $\Rightarrow$ `virtual int f()=NULL;`
- `override void f()...` $\Rightarrow$ `virtual void f()...`
- `abstract class c...` $\Rightarrow$ `class c` (překladač není úplný pitomec, aby přehlédl, že ve třídě je čistě virtuální funkce).

Vlastnosti C++

- Member- a friend-funkce: O vně sedící funkci můžeme v definici třídy říct, že je kamarád a že tudíž může k privátním položkám.

Vlastnosti C++

- Member- a friend-funkce: O vně sedící funkci můžeme v definici třídy říct, že je kamarád a že tudíž může k privátním položkám.
- Member-funkce se definují za interfacem pomocí operátoru čtyřtečky:
  `int trida::funkce(int parametr)...`

Vlastnosti C++

- Member- a friend-funkce: O vně sedící funkci můžeme v definici třídy říct, že je kamarád a že tudíž může k privátním položkám.
- Member-funkce se definují za interfacem pomocí operátoru čtyřtečky:
  `int trida::funkce(int parametr)...`
- Místo interfaců se používají abstraktní třídy a vícenásobná dědičnost (neříká se tudíž `implements`).

Vlastnosti C++

- Member- a friend-funkce: O vně sedící funkci můžeme v definici třídy říct, že je kamarád a že tudíž může k privátním položkám.
- Member-funkce se definují za interfacem pomocí operátoru čtyřtečky:
  `int trida::funkce(int parametr)...`
- Místo interfaců se používají abstraktní třídy a vícenásobná dědičnost (neříká se tudíž `implements`).
- Do šablony (předchůdce C# generik) lze strčit nejen typ, ale i hodnotu.

Vlastnosti C++

- Member- a friend-funkce: O vně sedící funkci můžeme v definici třídy říct, že je kamarád a že tudíž může k privátním položkám.
- Member-funkce se definují za interfacem pomocí operátoru čtyřtečky:
  `int trida::funkce(int parametr)...`
- Místo interfaců se používají abstraktní třídy a vícenásobná dědičnost (neříká se tudíž `implements`).
- Do šablony (předchůdce C# generik) lze strčit nejen typ, ale i hodnotu.
- C++ není vybaveno garbage collectorem, objekty se tvoří operátorem `new` a ruší operátorem `delete`, proto má dobrý smysl definovat destruktory.

Java

- Jazyku C# nejpodobnější (i povahou, tedy jde o interpretovaný jazyk).

Java

- Jazyku C# nejpodobnější (i povahou, tedy jde o interpretovaný jazyk).
- Javu umí několik miliard zařízení po světě (zpravidla programovaných v C).

Java

- Jazyku C# nejpodobnější (i povahou, tedy jde o interpretovaný jazyk).
- Javu umí několik miliard zařízení po světě (zpravidla programovaných v C).
- Memory-management vpodstatě totožný (Garbage-collector).

Java

- Jazyku C# nejpodobnější (i povahou, tedy jde o interpretovaný jazyk).
- Javu umí několik miliard zařízení po světě (zpravidla programovaných v C).
- Memory-management vpodstatě totožný (Garbage-collector).
- Syntakticky velmi podobná, ovšem bez namespaců.

Java

- Jazyku C# nejpodobnější (i povahou, tedy jde o interpretovaný jazyk).
- Javu umí několik miliard zařízení po světě (zpravidla programovaných v C).
- Memory-management vpodstatě totožný (Garbage-collector).
- Syntakticky velmi podobná, ovšem bez namespaců.
- Veřejnou třídu je třeba definovat v souboru jmenujícím se stejně jako tato třída.

Java

- Jazyku C# nejpodobnější (i povahou, tedy jde o interpretovaný jazyk).
- Javu umí několik miliard zařízení po světě (zpravidla programovaných v C).
- Memory-management vpodstatě totožný (Garbage-collector).
- Syntakticky velmi podobná, ovšem bez namespaců.
- Veřejnou třídu je třeba definovat v souboru jmenujícím se stejně jako tato třída.
- Ne vždy se do programu vstupuje statickou metodou `main`,

Java

- Jazyku C# nejpodobnější (i povahou, tedy jde o interpretovaný jazyk).
- Javu umí několik miliard zařízení po světě (zpravidla programovaných v C).
- Memory-management vpodstatě totožný (Garbage-collector).
- Syntakticky velmi podobná, ovšem bez namespaců.
- Veřejnou třídu je třeba definovat v souboru jmenujícím se stejně jako tato třída.
- Ne vždy se do programu vstupuje statickou metodou `main`,
- některé funkce se jmenují jinak (`System.Console.WriteLine` -> `System.out.println`)

Java

- Jazyku C# nejpodobnější (i povahou, tedy jde o interpretovaný jazyk).
- Javu umí několik miliard zařízení po světě (zpravidla programovaných v C).
- Memory-management vpodstatě totožný (Garbage-collector).
- Syntakticky velmi podobná, ovšem bez namespaců.
- Veřejnou třídu je třeba definovat v souboru jmenujícím se stejně jako tato třída.
- Ne vždy se do programu vstupuje statickou metodou `main`,
- některé funkce se jmenují jinak (`System.Console.WriteLine` -> `System.out.println`)
- Dědičnost se neřekne operátorem dvojtečky, ale slovem `extends`,

Java

- Jazyku C# nejpodobnější (i povahou, tedy jde o interpretovaný jazyk).
- Javu umí několik miliard zařízení po světě (zpravidla programovaných v C).
- Memory-management vpodstatě totožný (Garbage-collector).
- Syntakticky velmi podobná, ovšem bez namespaců.
- Veřejnou třídu je třeba definovat v souboru jmenujícím se stejně jako tato třída.
- Ne vždy se do programu vstupuje statickou metodou `main`,
- některé funkce se jmenují jinak (`System.Console.WriteLine` -> `System.out.println`)
- Dědičnost se neřekne operátorem dvojtečky, ale slovem `extends`,
- `using` $\Rightarrow$ `import`.

Java příklad aplikace

```
class apple{
      public static void main(String args[])
      {    System.out.println("Nic!");
      }
}
```

Další typy kódu v Javě

nejen aplikacemi živ je člověk

- Applet – kód vyskytující se na webových stránkách,

Další typy kódu v Javě

nejen aplikacemi živ je člověk

- Applet – kód vyskytující se na webových stránkách,
- MIDlet – kód provozovaný v rámci aplikací na mobilních telefonech,

Další typy kódu v Javě

nejen aplikacemi živ je člověk

- Applet – kód vyskytující se na webových stránkách,
- MIDlet – kód provozovaný v rámci aplikací na mobilních telefonech,
- aplikace pro Android – více typů.

Další typy kódu v Javě

nejen aplikacemi živ je člověk

- Applet – kód vyskytující se na webových stránkách,
- MIDlet – kód provozovaný v rámci aplikací na mobilních telefonech,
- aplikace pro Android – více typů.
- Mají společné rysy: Vždy je třeba definovat potomka správné třídy (v prvních dvou případech jasné, ve třetím je více možností),

Další typy kódu v Javě

nejen aplikacemi živ je člověk

- Applet – kód vyskytující se na webových stránkách,
- MIDlet – kód provozovaný v rámci aplikací na mobilních telefonech,
- aplikace pro Android – více typů.
- Mají společné rysy: Vždy je třeba definovat potomka správné třídy (v prvních dvou případech jasné, ve třetím je více možností),
- tajuplný interface plný různých pastí (například MIDlet nesmí mít statický kód přes 32 kB, paměti je většinou k dispozici kolem 2 MB, obrazovka se aktualizuje tajuplným způsobem, v různých implementacích Javy jsou různé chyby).

Další typy kódu v Javě

nejen aplikacemi živ je člověk

- Applet – kód vyskytující se na webových stránkách,
- MIDlet – kód provozovaný v rámci aplikací na mobilních telefonech,
- aplikace pro Android – více typů.
- Mají společné rysy: Vždy je třeba definovat potomka správné třídy (v prvních dvou případech jasné, ve třetím je více možností),
- tajuplný interface plný různých pastí (například MIDlet nesmí mít statický kód přes 32 kB, paměti je většinou k dispozici kolem 2 MB, obrazovka se aktualizuje tajuplným způsobem, v různých implementacích Javy jsou různé chyby).
- Obvykle pro 2. a 3. je potřeba zvláštní překladač.

Applet

```
import java.applet.*; //Kazdy musi includovat abstr.
Applet
import java.awt.*; //Kdo chce kreslit na obrazovku?
public class apple extends Applet{
     public void init(){}
     public void stop(){}
     public void paint(Graphics g)
     {    g.drawString("Ha!",20,20);
          g.drawString("Co znamena to vase
'Ha!'?",20,40);
     }
}  //Musí být v souboru apple.java (protože jde o veřejnou třídu)!
//Abstraktní třídě Applet je třeba definovat metodu init.
//Metoda paint se volá při překreslení okna.
//Okno se za běhu metody nepřekresluje
```

MIDlet I/II

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.util.*;
public class podivna_hra extends MIDlet {
     protected void startApp() throws
                    MIDletStateChangeException {
          Display display = Display.getDisplay(this);
          display.setCurrent((
                    new GameMidletScreen(this)) );
     }
...
```

MIDlet II/II

```
public class podivna_hra extends MIDlet {
     protected void startApp() throws
                    MIDletStateChangeException {
          Display display = Display.getDisplay(this);
          display.setCurrent((
                    new GameMidletScreen(this)) );
     }
     protected void destroyApp(boolean unconditional)
               throws MIDletStateChangeException {}
     public void exit()
     {    try{destroyApp(false);}catch(Exception e){}
          notifyDestroyed();
     }
     protected void pauseApp(){}
}
```

Aplikace pro Android

```
package pytel.com;

import android.app.Activity;
import android.os.Bundle;

public class AnthropoidActivity extends Activity {
     /** Called when the activity first created. */
     @Override
     public void onCreate(Bundle savedInstanceState){
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
     }
}
```

Mravní naučení

- Při obecné znalosti rodiny programovacích jazyků programování obnáší:

Mravní naučení

- Při obecné znalosti rodiny programovacích jazyků programování obnáší:
- Návrh algoritmu (teoretická práce s jasným cílem – matematika).

Mravní naučení

- Při obecné znalosti rodiny programovacích jazyků programování obnáší:
- Návrh algoritmu (teoretická práce s jasným cílem – matematika).
- Návrh a implementace interfacu (rituály s nejasnými vlastnostmi).

Mravní naučení

- Při obecné znalosti rodiny programovacích jazyků programování obnáší:
- Návrh algoritmu (teoretická práce s jasným cílem – matematika).
- Návrh a implementace interfacu (rituály s nejasnými vlastnostmi).
- Implementace jádra aplikace (praktická práce s jasnými pravidly).

Mravní naučení

- Při obecné znalosti rodiny programovacích jazyků programování obnáší:
- Návrh algoritmu (teoretická práce s jasným cílem – matematika).
- Návrh a implementace interfacu (rituály s nejasnými vlastnostmi).
- Implementace jádra aplikace (praktická práce s jasnými pravidly).
- Nepříjemný je návrh vstupů a výstupů (kde se projevují specifika jazyků),
  pokud člověk zvládne navrhnout vstup a výstup (a umí programovat), je téměř vyhráno.

Kam dál?

- Kamkoliv dle studijních plánů,

Kam dál?

- Kamkoliv dle studijních plánů,
- kamkoliv dle informatických studijních plánů, konkrétně třebas Programování:

Kam dál?

- Kamkoliv dle studijních plánů,
- kamkoliv dle informatických studijních plánů, konkrétně třebas Programování:
  - Jazyk C# a platforma .NET (Pavel Ježek – v zimě),

Kam dál?

- Kamkoliv dle studijních plánů,
- kamkoliv dle informatických studijních plánů, konkrétně třebas Programování:
  - Jazyk C# a platforma .NET (Pavel Ježek – v zimě),
  - Programování v C++ (Bednárek, Zavoral – v zimě),

Kam dál?

- Kamkoliv dle studijních plánů,
- kamkoliv dle informatických studijních plánů, konkrétně třebas Programování:
  - Jazyk C# a platforma .NET (Pavel Ježek – v zimě),
  - Programování v C++ (Bednárek, Zavoral – v zimě),
  - Java (Hnětynka – v zimě),

Kam dál?

- Kamkoliv dle studijních plánů,
- kamkoliv dle informatických studijních plánů, konkrétně třebas Programování:
  - Jazyk C# a platforma .NET (Pavel Ježek – v zimě),
  - Programování v C++ (Bednárek, Zavoral – v zimě),
  - Java (Hnětynka – v zimě),
  - Neprocedurální programování (pro neinformatiky) (Kryl, Kryl/Hric/Dvořák – v zimě)

Kam dál?

- Kamkoliv dle studijních plánů,
- kamkoliv dle informatických studijních plánů, konkrétně třebas Programování:
  - Jazyk C# a platforma .NET (Pavel Ježek – v zimě),
  - Programování v C++ (Bednárek, Zavoral – v zimě),
  - Java (Hnětynka – v zimě),
  - Neprocedurální programování (pro neinformatiky) (Kryl, Kryl/Hric/Dvořák – v zimě)
- Teoretická informatika: Automaty a gramatiky (Barták), Základy složitosti a vyčíslitelnosti (P. Kučera),

Kam dál?

- Kamkoliv dle studijních plánů,
- kamkoliv dle informatických studijních plánů, konkrétně třebas Programování:
  - Jazyk C# a platforma .NET (Pavel Ježek – v zimě),
  - Programování v C++ (Bednárek, Zavoral – v zimě),
  - Java (Hnětynka – v zimě),
  - Neprocedurální programování (pro neinformatiky) (Kryl, Kryl/Hric/Dvořák – v zimě)
- Teoretická informatika: Automaty a gramatiky (Barták), Základy složitosti a vyčíslitelnosti (P. Kučera),
- TI (algoritmy): Pravděpodobnostní-, Aproximační a on-line algoritmy (Sgall),

Kam dál?

- Kamkoliv dle studijních plánů,
- kamkoliv dle informatických studijních plánů, konkrétně třebas Programování:
  - Jazyk C# a platforma .NET (Pavel Ježek – v zimě),
  - Programování v C++ (Bednárek, Zavoral – v zimě),
  - Java (Hnětynka – v zimě),
  - Neprocedurální programování (pro neinformatiky) (Kryl, Kryl/Hric/Dvořák – v zimě)
- Teoretická informatika: Automaty a gramatiky (Barták), Základy složitosti a vyčíslitelnosti (P. Kučera),
- TI (algoritmy): Pravděpodobnostní-, Aproximační a on-line algoritmy (Sgall),
- cokoliv dle vlastního uvážení (nejlépe po dohodě s přednášejícím).

Závěr

- Informatika není jen programování,

Závěr

- Informatika není jen programování,
- ale programování je předpokladem pochopení informatických problémů,

Závěr

- Informatika není jen programování,
- ale programování je předpokladem pochopení informatických problémů,
- a kromě toho programování je v současné době výhodnou schopností (mimo jiné finančně).

Závěr

- Informatika není jen programování,
- ale programování je předpokladem pochopení informatických problémů,
- a kromě toho programování je v současné době výhodnou schopností (mimo jiné finančně).
- Děkuji za pozornost...