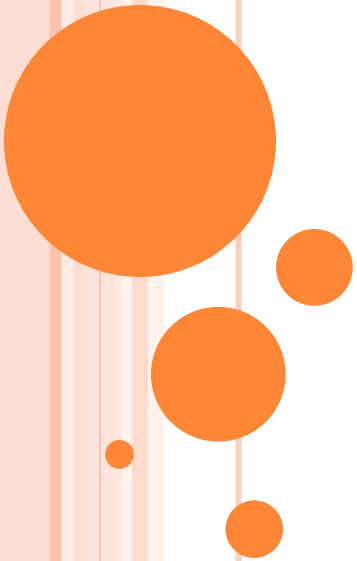


Programovacie jazyky pre vstavané systémy

Cvičenie 4

Náplň cvičenia

1. Smerníky a typové konverzie.
2. Reťazce.
3. Nastavenia v NetBeans.
4. Jednoduché úlohy a úlohy s reťazcami.

Otázky

- Čo je to smerník/ukazovateľ/„pointer“?
- Aký je rozdiel medzi odovzdávaním parametrov hodnotou a referenciou?
- Aký spôsob odovzdávania parametrov podporuje jazyk C?
- Ako sú odovzdávané polia do funkcií? Vysvetlite.
- Ako sú implementované reťazce v jazyku C?
- Môžeme priradiť premennú typu pole do inej premennej typu pole?
- Vysvetlite rozdiel medzi nasledujúcimi zápismi:

```
char ret1[] = "Ahoj";
char *ret2 = "Ahoj";
char *ret3 = ret1;
```

Ukazovateľ (smerník)

- Odvodený dátový typ, ktorý uchováva adresu nejakého objektu alebo funkcie.
- Ukážka práce so smerníkmi:

```
int x;
int *px = &x;
int **p_px;
p_px = &px;

*px = 10;
printf("x = %d\n", x);
printf("x = %d\n", *px);
printf("x = %d\n", **p_px);
printf("&x = %p\n", &x);
printf("&x = %p\n", px);
printf("&x = %p\n", *p_px);
```

Smerníková aritmetika a polia

```c
#include <stdlib.h>
#include <stdio.h>

#define VELKOST_POLA 10

void naplnPole(int n, int pole[]) {
    for (int i = 0; i < n; i++, pole++) {
        *pole = i;
    }
}

void vypisPole2(int* zac, int* kon) {
    int *akt = zac;
    while (akt <= kon) {
        printf("%d ", *akt);
        akt++;
    }
    printf("\n");
}

int main(int argc, char* argv[]) {
    int pole[VELKOST_POLA];
    naplnPole(VELKOST_POLA, pole);
    vypisPole2(pole, pole + VELKOST_POLA - 1);
}
```

Konverzie medzi smerníkmi

- Ľubovoľný smerník na objekt môže byť pretypovaný na ľubovoľný iný typ smerníka na objekt.
- Vo všeobecnosti je však bezpečné pretypovávať len smerníky z väčších dátových typov na menšie (napr. int* na char*).
- Zistite, čo robí nasledujúci kód.

```c
#include <stdlib.h>
#include <stdio.h>
#include <limits.h>

int main(int argc, char* argv[]) {
    int i = INT_MAX, *pi = &i;
    short *ps = (short *)pi;
    int *pi_2 = (int *)ps;

    printf("*pi = %d\n", *pi);
    printf("*ps = %d, *(ps + 1) = %d\n", *ps, *(ps + 1));
    printf("*pi_2 = %d\n", *pi_2);

    printf("pi = %p\n", pi);
    printf("ps = %p, ps + 1 = %p\n", ps, ps + 1);
    printf("pi_2 = %p\n", pi_2);

    return 0;
}
```

Jednoduchý test endianity

- Vysvetlite, ako pracuje nasledujúci kód:

```c
#include <stdlib.h>
#include <stdio.h>

_Bool jeLittleEndian() {
    int cislo = 1;
    return ((char*)&cislo)[0] == (char)1;
}

int main(int argc, char** argv) {
    if (jeLittleEndian())
        printf("Poradie bajtov je little endian.\n");
    else
        printf("Poradie bajtov je big endian.\n");

    return 0;
}
```

Reťazce

- V jazyku C sa pod reťazcom rozumie pole znakov ukončené znakom '\0'.
- Vysvetlite, čo sa stane pri nasledujúcich inicializáciách:

```
char ret[10] = "Ahoj";
char ret[] = "Ahoj";
char ret[2] = "Ahoj";
```

- Vysvetlite, čo robia nasledujúce funkcie:
  - fgets(), puts()
  - strlen(), strncpy, strncat, strcmp
  - atoi(), atol(), atoll(), atof()
  - sprintf(), sscanf()
- Musia dostávať funkcie pracujúce s reťazcami ako parameter veľkosť reťazca? Vysvetlite.

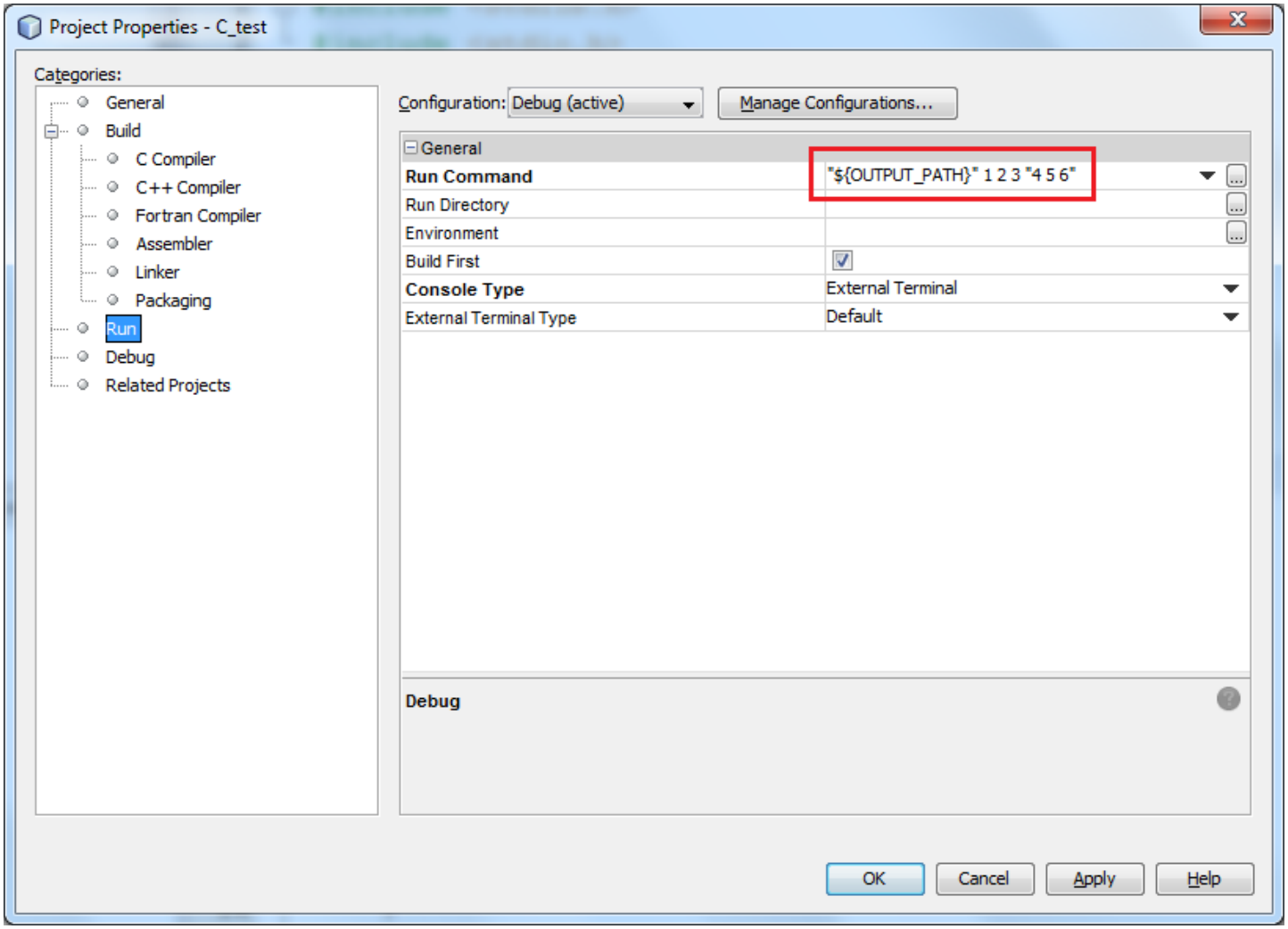
Argumenty programu

- Každý program môžeme spustiť s argumentmi, ktoré môže bližšie špecifikovať, čo má program robiť.
- Argumenty, s ktorými sa program spustí, sa odovzdávajú do funkcie main ako pole reťazcov.

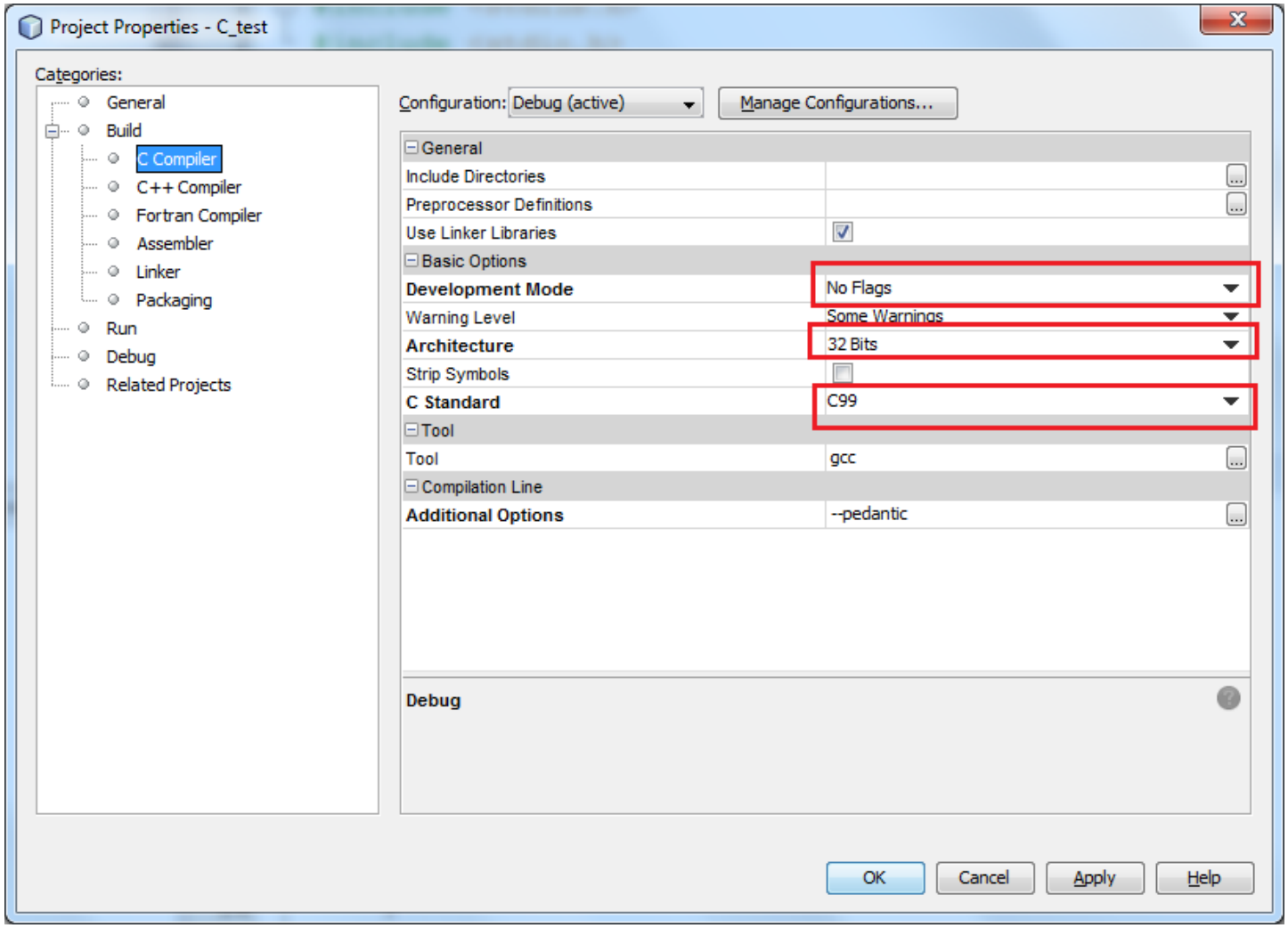
```
1  #include <stdlib.h>
2  #include <stdio.h>
3
4  int main(int argc, char* argv[]) {
5      for (int i = 0; i < argc; i++) {
6          printf("%s\n", argv[i]);
7      }
8  }
```

- Prepíšte predchádzajúci kód pomocou smerníkovej aritmetiky.

NetBeans – spustenie programu s argumentmi



NetBeans – nastavenie typu kompilácie, štandardu C a architektúry



Jednoduché úlohy

- Vytvorte jednoduché funkcie (navrhnite parametre a návratový typ), ktoré umožnia:
  - vymeniť obsah dvoch premenných typu int;
  - vypočítať korene kvadratickej rovnice nad oborom komplexných čísel (korene vráťte cez parametre);
  - vypočítať súčet dvoch komplexných polynómov (súčet vráťte cez parameter);
  - vrátiť minimum a jeho index v poli celých čísel odovzdanom ako parameter.

Úlohy – práca s reťazcami

- Vytvorte nasledujúce funkcie:
  - char* trim(char* str) – odstráni všetky biele znaky zo začiatku a z konca reťazca odovzdaného ako parameter, funkcia vráti smerník na prvý znak orezaného reťazca;
  - char* caesar(char* src, char* dest, int shift) – funkcia zašifruje zdrojový reťazec (src) pomocou cézarovej šifry s posunom shift a uloží ho do reťazca dest, funkcia vráti smerník na prvý znak zašifrovaného reťazca;
  - char* strDel(char* str, int pos, int count) – funkcia odstráni od pozíce pos v reťazci str count znakov, funkcia vráti smerník na prvý znak modifikovaného reťazca;
  - char* strIns(char* dest, int pos, char *src) – funkcia vloží do reťazca dest na pozíciu pos reťazec src, funkcia vráti smerník na prvý znak reťazca dest (??);
  - char* substitute(char* src, char* pattern, char* sub) – funkcia nahradí v reťazci src všetky výskyty reťazca pattern reťazcom sub, funkcia vráti smerník na prvý znak modifikovaného reťazca (??);
  - _Bool isPalindrome(char* str) – funkcia zistí, či reťazec str je palindróm;
  - char* reverse(char* str) – funkcia preklopí reťazec ("Ahoj" -> " johA") a vráti smerník na prvý znak preklopeného reťazca;
  - char* toLowerStr(char* str) – funkcia prevedie všetky písmená v reťazci na malé a vráti smerník na prvý znak modifikovaného reťazca;
  - char* toUpperStr(char* str) – funkcia prevedie všetky písmená v reťazci na veľké a vráti smerník na prvý znak modifikovaného reťazca;
- Vytvorte program, ktorý umožní:
  a) sčítať 2 celé čísla, ktoré môžu mať najviac *n* cifier (+ znamienko);
  b) odčítať 2 celé čísla, ktoré môžu mať najviac *n* cifier (+ znamienko);
  - parameter *n* bude predstavovať argument odovzdaný programu pri jeho spustení, pričom:
    - ak *n* je reťazec alebo číslo menšie ako 1, program vypíše chybové hlásenie
    - ak *n* nie je definované, implicitne sa bude predpokladať hodnota 50

Úlohy – bonusové

- Do programu pre sčítavanie a odčítavanie celých čísel dorobte funkcie, pomocou ktorých bude možné:
  a) vynásobiť 2 celé čísla, ktoré môžu mať najviac *n* cifier (+ znamienko);
  b) celočíselne vydeliť 2 celé čísla, ktoré môžu mať najviac *n* cifier (+ znamienko);
  c) vypočítať zvyšok po celočíselnom delení 2 celých čísel, ktoré môžu mať najviac *n* cifier (+ znamienko).
- Vytvorte program, ktorý z klávesnice načíta reťazec a zistí, či sa jedná o platné rodné číslo.